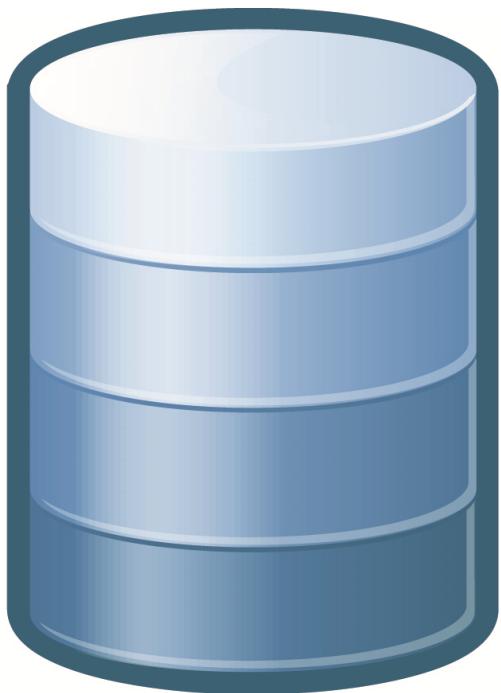


NexusDB Manual

© 2019 NexusQA Pty Ltd



Your guide to NexusDB

by NexusQA Pty Ltd

NexusDB Manual

© 2019 NexusQA Pty Ltd

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: June 2019

Table of Contents

Foreword	53
Part I Hardware and Software Requirements	54
Part II NexusDB Architecture	55
1 Historical Context.....	55
2 NexusDB V2 Features.....	56
3 NexusDB Architecture.....	58
Client	60
Server	61
Remoting	62
4 Deployment Options.....	63
Part III NexusDB Concepts	65
1 Aliases.....	65
2 Autolnc Fields.....	65
3 Blob Handling.....	66
Normal Blob Handling Server Side	66
Normal Blob Handling Client Side	67
Blob Caching	68
4 Data Dictionary.....	69
Descriptors	70
Overview of Sub-Engines	71
5 Data Security.....	72
Security Issues	73
6 Fulltext Indexing.....	74
7 In-Memory Tables.....	75
8 Monitors and Extenders.....	75
9 Plug-ins.....	76
10 Real-time Backups.....	76
11 SMP Scalability.....	77
12 SQL Joins.....	78
13 Transactions and Data Integrity.....	85
Locks and Locking	85
Deadlocks	88
Optimistic Record Locks	89
Transaction Types	89
Nested Transactions	91
Snapshot Transactions.....	91
Failsafe Transactions.....	91
Implications for Developers	93
Advice for Developers	93
Referential Integrity	94

14 Transports.....	94
Selecting the correct Transport	95
Why use compression and batch modes?.....	96
Which compression and batch size should I use?.....	96
Part IV Appendices	98
1 System Capacities.....	98
2 Data Types.....	99
3 Error Messages.....	100
Part V Data Management Tools Home	102
Part VI The NexusDB Server	103
1 Introduction.....	103
2 The Server User Interface.....	103
Settings Panel	105
Aliases	106
Transports.....	107
Blow fish/RC4 Secured Transport.....	107
COM Transport.....	108
Direct TCP/.NET Transport.....	109
Named Pipe Transport.....	110
TCP/IPv4 Transport.....	111
Event Log.....	113
GUI Settings.....	115
Server Engine.....	115
Plugins	117
SQL Engine.....	118
Users	120
Statistics Panel	122
Transports.....	122
Server Engine.....	123
Summary.....	124
3 Configuring and Testing the NexusDB Server.....	124
4 Running the Server as a WinNT Service.....	125
5 The Server Settings File	127
How are settings stored?	128
Example of configuration file	130
Part VII The Enterprise Manager	147
1 Introduction.....	147
2 Global Options.....	148
3 Using the Servers Window.....	150
Server Context Menu	152
The Database Context Menu	155
The Table Context Menu	157
4 Creating and Restructuring Tables.....	160
Regular Identifiers	160

Field Types	162
Table Structure Window	163
Locale Descriptor.....	164
File Descriptors.....	164
Encryption Engine.....	165
Autolnc Engine.....	166
Field Descriptors.....	167
Default Values	168
Index Descriptors.....	169
Record Engine.....	171
5 Browsing Tables and Queries.....	172
Using the Data Grid	172
Browsing a Table	173
The Table Menu.....	176
The Table View Menu.....	177
The Table Options Menu.....	178
Using SQL	178
The Query Menu.....	180
The Query View Menu.....	181
The Query Connections Menu.....	182
The Query Options Menu.....	183
6 Importing CSV Files.....	185
Step One - File Type	185
Step Two - Delimiters	186
Step Three - Field Formats and Advanced Options	187
7 Printing Reports.....	189
8 Runtime Loaded Packages.....	190
Part VIII The Importer	191
1 Introduction.....	191
2 Design.....	191
Known Issues	192
3 Command Line Usage.....	192
Configuration file example	193
Known Issue	195
4 Interactive Usage.....	195
Engine Selection	195
Configuration of the Source Engines	196
Borland Database Engine.....	196
FlashFiler 2.....	199
ADO	201
DAO	202
Configuration of the Destination Engines	203
NexusDB.....	203
Configuration of the Conversion Process	206
Transfer Process	208
5 Gotcha's.....	209
Side Effects of the "Directory Mode"	209
Misinterpretation of the Table Types	210
Long Import Durations	210

Long File Problems	210
6 Implementation Details.....	210
User Interface	211
Command Line Parameters.....	211
Graphical User Interface.....	212
Importer Classes	212
Transfer Manager.....	212
Data Engines.....	212
Structure Converters.....	213
Data Movers.....	213
How to add a new importer	213
Using the Importer Classes in own applications	214
Part IX Introduction	216
Part X Installation Guidelines	217
1 Embedded Server Version.....	217
2 (Delphi) Developer Version.....	217
3 (Delphi) Enterprise Version.....	217
4 Trial Versions.....	217
5 Upgrading from V1.....	218
6 Recompiling the Packages and Binaries.....	218
Part XI VCL Component Overview	220
1 Components on the VCL Palette.....	220
2 Common Properties and Events.....	227
3 Server Engines and Monitors.....	228
TnxSQLEngine	228
TnxServerEngine	229
TnxServerManager	231
Tnx1xAllEngines and TnxseAllEngines	232
TnxPascalScriptEngine	233
TnxServerScriptEngine	233
TnxSimpleMonitor	234
TnxSQLTriggerMonitor	236
TnxSecurity Monitor	237
4 Transport Group.....	238
Common Transport Properties and Events	238
TnxCustomCOMTransport	239
TnxRegisteredCOMTrasnsport	241
Common to Winsock and Named Pipe Transports	242
TnxSharedMemoryTransport	243
TnxWinsockTransport	245
TnxNamedPipeTransport	247
TnxBlowfishRC4SecuredTransport	249
5 Servers and Command Handlers.....	250
TnxServerCommandHandler	250
TnxSecuredCommandHandler	251
TnxSimpleCommandHandler	252

TnxCustomConnectionFilter	253
TnxRemoteServerEngine	255
6 Sessions and Databases.....	256
TnxSessionPool.....	256
TnxSession	257
TnxDatabase	259
TnxBackupController	260
TnxSimpleSession	262
TnxTransContext	264
TnxDatabaseMapper	265
7 Plugins.....	266
TnxServerInfoPlugin	266
TnxRemoteServerInfoPlugin	267
TnxServerInfoPluginCommandHandler	268
TnxRemotingServer	269
TnxRemotingClient	270
TnxRemotingCommandHandler	271
TnxRemoteRemoteCommandsPlugin	272
TnxServerRemoteCommandsPlugin	274
TnxRemoteCommandsPluginCommandHandler	275
8 TnxDataset Group.....	277
Common Properties and Events	277
TnxTable	278
TnxQuery	280
TnxCachedDataset	282
TnxStoredProc	283
TnxSQLUpdateObject	285
TnxMemTable	286
9 Logging Components.....	287
TnxEventBasedLog	287
TnxWinEventLog	288
TnxEventLogDispatcher	289
TnxEventLog	290

Part XII Converting from other Versions or Database Engines 293

1 Upgrading V1 Applications.....	293
2 Delphi Interfaces.....	293
3 Upgrading FF2 Projects to NexusDB.....	297
Differences between FF and Nexus DB	298

Part XIII Code Examples & Fragments 303

1 Creating the Northwind sample database.....	303
2 Applications with the database embedded.....	304
Setting up a general NexusDB Embedded Server Data Module	304
Creating a simple databound form	307
Making the embedded application path independent (copy)	310
3 Client/Server Applications - Client App Separate from the Server.....	311
Starting and setting up the server	312
Setting up a generic NexusDB Data Module for C/S	312

Creating a simple databound form	316
Making the application server independent	319
4 Mixing Embedded and C/S Applications.....	320
5 Converting Paradox Tables to NexusDB Tables.....	325
6 How to create Tables?.....	330
Creating Tables Using the Data Dictionary	331
Creating Tables Using FieldDefs	333
Creating Tables Using SQL DDL	333
Creating Tables Using the Enterprise Manager	335
Thoughts about In-Memory Tables	335
7 How to create Indices?.....	335
Index Descriptors	336
Key Descriptors	336
Key Field Descriptors	336
How to create Indices in Code	337
8 How to create a monitor/extender?.....	338
9 How to optimize read/write access?.....	345
10 How to restructure Tables?.....	352
11 How do handle Autoinc manually?.....	353
12 How to handle locking errors?.....	354
13 How to use Memory Tables?.....	355
Part XIV Frequently asked Questions	364
Part XV Introduction	381
Part XVI About NexusDB SQL Reference	383
Part XVII SQL Language Elements	386
1 Keywords.....	386
2 Identifiers.....	391
3 Comments.....	394
4 Operators.....	395
5 Null Values.....	397
6 Literals.....	398
Numeric Literals	399
String Literals	401
DateTime Literals	403
Interval Literals	404
Boolean Literals	406
7 Value Expressions.....	407
Numeric Value Expressions	408
String Value Expressions	410
DateTime Value Expressions	412
Boolean Value Expressions	413
Row Value Expressions	416
Subqueries	417

Case Expressions	419
Cast Specification	422
8 Parameters.....	424

Part XVIII SQL Data Types 426

1 Overview of Predefined Data Types.....	429
2 String Types.....	431
3 Numeric Types.....	435
4 Boolean Types.....	438
5 DateTime Types.....	439

Part XIX Functions 441

1 Aggregate Functions.....	441
2 Numeric Value Functions.....	445
Position Expression	446
Extract Expression	448
Length Expression	449
Absolute Value Expression	451
Modulus Expression	451
Natural Logarithm	452
General Logarithm	453
Exponential Function	454
Power Function	455
Square Root Function	456
Floor Function	457
Ceiling Function	457
Arctangent Function	458
Cosine Function	459
Sine Function	460
Ordinal Function	461
PI Function	462
Round Function	462
Random Function	463
LastAutoinc Function	464
3 String Value Functions.....	465
Substring Function	465
OVERLAY Function	469
Fold Function	470
Trim Function	470
Char Function	472
ToString Function	472
ToStringLen Function	473
4 DateTime Value Functions.....	474
Current Date Function	475
Current Time Function	475
Current Timestamp Function	476
OVERLAY Function	477
5 System Functions.....	478

Part XX Predicates	481
1 Comparison Predicate.....	482
2 Between Predicate.....	483
3 In Predicate.....	484
4 Like Predicate.....	486
5 Null Predicate.....	487
6 Quantified Comparison Predicate.....	488
7 Exists Predicate.....	491
8 Unique Predicate.....	492
9 Match Predicate.....	493
10 Equivalent Predicate.....	494
11 Odd Predicate.....	495
12 Inserting Predicate.....	496
13 Updating Predicate.....	497
14 Deleting Predicate.....	498
15 SIMILAR Predicate.....	499
16 CONTAINS Predicate.....	500
Part XXI Stored Procedures and Functions	501
1 Procedure Language.....	503
2 SQL-Invoked Routines.....	503
3 SQL Procedure Statements.....	509
Part XXII SQL Statements	512
1 Prepared Statements.....	512
2 Statement Switches.....	513
3 Schema Statements.....	514
CREATE TABLE statement	515
CREATE INDEX statement	526
CREATE VIEW statement	528
CREATE TRIGGER statement	529
CREATE PROCEDURE statement	532
CREATE FUNCTION statement	536
CREATE ASSEMBLY statement	539
ALTER TABLE statement	541
DROP TABLE statement	544
DROP INDEX statement	545
DROP VIEW statement	546
DROP TRIGGER statement	547
DROP ROUTINE statement	548
DROP ASSEMBLY statement	550
4 Data Statements.....	551
SELECT statement	551
INSERT statement	560

UPDATE Statement	562
DELETE Statement	564
EXECUTE IMMEDIATE Statement	566
5 CURSOR Statements.....	566
CLOSE Statement	566
DECLARE CURSOR Statement	566
FETCH Statement	568
OPEN Statement	568
6 Transaction Statements.....	568
START TRANSACTION statement	568
COMMIT statement	570
ROLLBACK statement	571
7 Session Statements.....	572
SET PASSWORDS statement	572
ALTER SYSTEMPASSWORD statement	574
CREATE ALIAS statement	574
ALTER ALIAS statement	574
DROP ALIAS Statement	575
USE Statement	575
8 Control Statements.....	575
CALL statement	576
RETURN statement	577
Compound statement	578
DECLARE variable statement	579
SET statement	581
IF statement	582
ITERATE statement	583
LEAVE statement	585
REPEAT statement	586
WHILE statement	587
SIGNAL statement	588
TRY statement	589
9 Special Statements.....	591
ASSERT TABLE statement	591
Part XXIII System Tables	593
Part XXIV Appendices	595
1 Core SQL:2003 Features.....	595
2 Additional SQL:2003 Features.....	600
Part XXV Symbol Reference	603
1 Classes.....	603
EnxAbortStateChange Class	603
EnxAdvAutoIncDescriptorException Class	603
EnxBaseAutoIncDescriptorException Class	603
EnxBaseBlobDescriptorException Class	604
EnxBaseBlockHeapDescriptorException Class	604
EnxBaseCustomDescriptorException Class	605
EnxBaseDefaultValueDescriptorException Class	605

EnxBaseDictionaryException Class	605
EnxBaseFieldValidationDescriptorException Class	606
EnxBaseHeapDescriptorException Class	606
EnxBaseIndicesDescriptorException Class	606
EnxBaseKeyDescriptorException Class	607
EnxBaseLogException Class	607
EnxBaseRecordCompressionDescriptorException Class	608
EnxBaseRecordDescriptorException Class	608
EnxBaseStreamDescriptorException Class	608
EnxBaseTableDescriptorException Class	609
EnxCommandHandlerException Class	609
EnxCompKeyDescriptorException Class	609
EnxComponentException Class	610
Constructors.....	610
nxcCreate	610
nxcCreate Constructor (TComponent, PResStringRec).....	610
nxcCreate Constructor (TComponent, PResStringRec, array of const)	611
nxcCreate Constructor (TComponent, TnxResult).....	611
nxcCreate Constructor (TComponent, TnxResult, PResStringRec).....	611
nxcCreate Constructor (TComponent, TnxResult, PResStringRec, array of const).....	612
nxcCreate Constructor (TComponent, TnxResult, string).....	612
nxcCreate Constructor (TComponent, TnxResult, string, array of const)	613
nxcCreate Constructor (TComponent, string).....	613
nxcCreate Constructor (TComponent, string, array of const).....	613
EnxConstDefaultValueDescriptorException Class	614
EnxCustomDescsDescriptorException Class	614
EnxDataDictionaryException Class	615
EnxDictionaryItemException Class	615
EnxExtTextKeyFieldDescriptorException Class	615
EnxFieldDescriptorException Class	616
EnxFieldsDescriptorException Class	616
EnxFieldValidationsDescriptorException Class	616
EnxFileDescriptorException Class	617
EnxFilesDescriptorException Class	617
EnxHeapBlobDescriptorException Class	618
EnxHeapRecordDescriptorException Class	618
EnxIndexDescriptorException Class	618
EnxKeyFieldDescriptorException Class	619
EnxLocaleDescriptorException Class	619
EnxLocalizedDictionaryItemException Class	619
EnxLoggableComponentException Class	620
EnxMainIndicesDescriptorException Class	620
EnxMappedKeyDescriptorException Class	621
EnxMinMaxValidationDescriptorException Class	621
EnxNestedTableDescriptorException Class	621
EnxNoChangeValidationDescriptorException Class	622
EnxNotNullCompKeyDescriptorException Class	622
EnxPluginCommandHandlerException Class	622
EnxRefKeyDescriptorException Class	623
EnxRegisterableComponentException Class	623
EnxSqlCheckValidationDescriptorException Class	623
EnxStateComponentException Class	624

EnxTablesDescriptorException Class	624
EnxTextKeyFieldDescriptorException Class	625
EnxTransportException Class	625
TnxAbstractCursor Class	625
Constructors.....	626
Create Constructor.....	626
Destructors.....	626
Destroy Destructor.....	626
Methods	627
AfterConstruction Method.....	627
AutoIncGet Method.....	627
AutoIncSet Method.....	627
BeforeDestruction Method.....	628
BlobCreate Method.....	628
BlobCreateFile Method.....	628
BlobDelete Method.....	629
BlobFree Method.....	629
BlobGetLength Method.....	630
BlobModified Method.....	630
BlobRead Method.....	630
BlobTruncate Method.....	631
BlobWrite Method.....	631
BookmarkAsVariant Method.....	632
ChangePassword Method.....	632
CompareBookmarks Method.....	633
CompareKeys Method.....	633
CopyRecords Method.....	634
CopyRecordsEx Method.....	635
DeleteRecords Method.....	635
Duplicate Method.....	635
FilterActivate Method.....	636
FilterAddCallback Method.....	636
FilterAddCustom Method.....	636
FilterAddExpression Method.....	637
FilterDeactivate Method.....	637
FilterRemove Method.....	638
FilterSetTimeout Method.....	638
GetBookmark Method.....	638
GetIndexID Method.....	639
GetRecordCount Method.....	639
GetRecordCountAsync Method.....	639
GetRecordCountEx Method.....	640
GetRecordCountExAsync Method.....	640
KeyFilterActivate Method.....	640
KeyFilterAddCallback Method.....	641
KeyFilterAddCustom Method.....	641
KeyFilterAddExpression Method.....	642
KeyFilterDeactivate Method.....	642
KeyFilterRemove Method.....	642
LookupByID Method.....	643
RangeReset Method.....	643
RangeSet Method.....	643
RecNoGet Method.....	644
RecNoSet Method.....	645

RecNoSupported Method.....	645
RecordDelete Method.....	646
RecordGet Method.....	646
RecordGetBatch Method.....	646
RecordGetForKey Method.....	647
RecordGetNext Method.....	648
RecordGetPrior Method.....	648
RecordInsert Method.....	648
RecordInsertBatch Method.....	649
RecordIsLocked Method.....	649
RecordLockRelease Method.....	650
RecordModify Method.....	650
RequestNestedTransactions Method.....	650
SetToBegin Method.....	651
SetToBookmark Method.....	651
SetToCursor Method.....	651
SetToEnd Method.....	652
SetToFilter Method.....	652
SetToKey Method.....	652
Sw itchToIndex Method.....	653
TableIsLocked Method.....	654
TableLockAcquire Method.....	654
TableLockRelease Method.....	655
TableStreamDelete Method.....	655
TableStreamGetList Method.....	655
TableStreamRead Method.....	656
TableStreamWrite Method.....	656
Properties	656
(Dictionary Property.....	656
Database Property	657
DatabaseNext Property.....	657
DatabasePrev Property.....	657
FullName Property.....	658
TableDescriptor Property.....	658
TnxAbstractDatabase Class	658
Constructors.....	659
Create Constructor.....	659
Destructors.....	659
Destroy Destructor.....	659
Methods	660
AfterConstruction Method.....	660
BeforeDestruction Method.....	660
CursorOpen Method.....	660
GetChangedTables Method.....	661
GetFreespace Method.....	661
GetPath Method.....	662
LookupByID Method.....	662
StatementAlloc Method.....	663
StatementExecDirect Method.....	663
TableAddIndex Method.....	664
TableAutolncGet Method.....	664
TableBackup Method.....	665
TableBuild Method.....	665
TableChangePassword Method.....	666

TableDelete Method.....	666
TableDropIndex Method.....	667
TableEmpty Method.....	667
TableExists Method.....	668
TableGetDictionary Method.....	668
TableGetList Method.....	669
TablePack Method.....	669
TableRebuildIndex Method.....	670
TableRecover Method.....	670
TableRename Method.....	671
TableRestructure Method.....	671
TransactionCommit Method.....	672
TransactionCorrupted Method.....	672
TransactionGetLevel Method.....	672
TransactionRollback Method.....	673
TransactionStart Method.....	673
TransactionStartWith Method.....	673
Properties.....	674
Session Property.....	674
TransContext Property.....	674
TnxAbstractSecurityMonitor Class	675
Methods	675
HasAdmins Method.....	675
HasUsers Method.....	675
IsAdmin Method.....	676
UiStateVisible Method.....	676
TnxAbstractServerObject Class	676
Methods	677
BeforeDestruction Method.....	677
Free Method	677
Properties.....	677
RemoteThreadPriority Property.....	677
TnxAbstractSession Class	678
Constructors.....	678
Create Constructor.....	678
Destructors.....	679
Destroy Destructor.....	679
Methods	679
AfterConstruction Method.....	679
AliasAdd Method.....	679
AliasDelete Method.....	680
AliasGetList Method.....	680
AliasGetPath Method.....	681
AliasModify Method.....	681
BeforeDestruction Method.....	682
CloseInactiveFolders Method.....	682
CloseInactiveTables Method.....	682
DatabaseOpen Method.....	683
Failed Method	684
GetRegisteredClassList Method.....	684
GetUserInfo Method.....	684
LookupByID Method.....	685
PasswordAdd Method.....	685
PasswordRemove Method.....	685

PasswordRemoveAll Method.....	686
TransContextCreate Method.....	686
Properties.....	687
Authenticated Property	687
CleanedUp Property	687
ClientVersion Property	687
ConnectedFrom Property	688
Password Property.....	688
ServerEngine Property.....	688
ServerVersion Property.....	689
UserName Property.....	689
TnxAbstractStatement Class	689
Constructors.....	690
Create Constructor.....	690
Destructors.....	690
Destroy Destructor.....	690
Methods	690
AfterConstruction Method.....	690
BeforeDestruction Method.....	691
Exec Method	691
GetParams Method.....	692
LookupByID Method.....	692
Prepare Method.....	692
SetParams Method.....	693
Properties.....	693
Database Property	693
TnxAbstractTaskInfo Class	694
Constructors.....	694
Create Constructor.....	694
Destructors.....	695
Destroy Destructor.....	695
Methods	695
AfterConstruction Method.....	695
BeforeDestruction Method.....	695
Cancel Method	696
GetStatus Method.....	696
LookupByID Method.....	696
Properties.....	697
Session Property.....	697
TnxAbstractTimeoutObject Class	697
Constructors.....	698
Create Constructor.....	698
Methods	698
OptionClear Method.....	698
OptionGet Method.....	698
OptionGetEffective Method.....	699
OptionList Method.....	699
OptionListEffective Method.....	699
OptionSet Method.....	700
Properties	700
Timeout Property	700
TnxAbstractTransContext Class	701
Constructors.....	701
Create Constructor.....	701

Destructors.....	702
Destroy Destructor.....	702
Methods	702
AfterConstruction Method.....	702
BeforeDestruction Method.....	702
Failed Method	703
LookupByID Method.....	703
TransactionCommit Method.....	703
TransactionCorrupted Method.....	704
TransactionGetLevel Method.....	704
TransactionRollback Method.....	704
TransactionStart Method.....	705
TransactionStartWith Method.....	705
Properties.....	706
Session Property.....	706
SessionNext Property.....	706
SessionPrev Property.....	706
TnxAdvAutoIncDescriptor Class	707
Methods	707
isEqual Method.....	707
LoadFromReader Method.....	707
SaveToWriter Method.....	708
Properties.....	708
InitialValue Property.....	708
Step Property	709
TnxAliasDescriptor Class	709
TnxAutoGuidDefaultValueDescriptor Class	709
TnxBaseAutoIncDescriptor Class	710
Destructors.....	710
Destroy Destructor.....	710
Methods	710
CheckValid Method.....	710
isEqual Method.....	711
LoadFromReader Method.....	711
SaveToWriter Method.....	711
Properties.....	712
AutoIncEngine Property.....	712
FileNumber Property.....	712
HeaderSection Property.....	712
TnxBaseBlobDescriptor Class	713
Constructors.....	713
Create Constructor.....	713
Destructors.....	714
Destroy Destructor.....	714
Methods	714
CheckValid Method.....	714
GetList Method.....	714
isEqual Method	715
LoadFromReader Method.....	715
SaveToWriter Method.....	715
Properties.....	716
BlobEngine Property.....	716
FileNumber Property.....	716
HeaderSection Property.....	716

TnxBaseBlobStream Class	717
Destructors.....	717
Destroy Destructor.....	717
Methods	718
Truncate Method.....	718
TnxBaseBlockHeapDescriptor Class	718
Methods	718
IsEqual Method.....	718
LoadFromReader Method.....	719
SaveToWriter Method.....	719
Properties	720
BlockHeapEngine Property.....	720
TnxBaseCommandHandler Class	720
Constructors.....	720
Create Constructor.....	720
Destructors.....	721
Destroy Destructor.....	721
Methods	721
UISettingsVisible Method.....	721
TnxBaseComponentConfiguration Class	721
Methods	722
GetValue Method.....	722
GetValueStream Method.....	722
SetValue Method.....	723
SetValueStream Method.....	723
TnxBaseConnectionFilter Class	723
TnxBaseCustom Descriptor Class	724
Methods	724
CheckValid Method.....	724
IsEqual Method.....	724
Properties	725
Name Property	725
TnxBaseDefaultValueDescriptor Class	725
Methods	726
CheckValid Method.....	726
GetList Method.....	726
IsEqual Method.....	726
LoadFromReader Method.....	727
SaveToWriter Method.....	727
Properties	727
ApplyAt Property.....	727
ApplyOnInsert Property.....	728
ApplyOnModify Property.....	728
OverwriteNonNull Property.....	728
TnxBaseDirectTransport Class	729
TnxBaseEngineExtender Class	729
Constructors.....	729
Create Constructor.....	729
Destructors.....	730
Destroy Destructor.....	730
Methods	730
Notify Method	730
Properties	731
ExtendableObject Property.....	731

Monitor Property.....	731
TnxBaseEngineMonitor Class	731
Constructors.....	732
Create Constructor.....	732
Destructors.....	732
Destroy Destructor.....	732
Methods	733
ExtendableObjectCreated Method.....	733
ExtendableObjectDestroyed Method.....	733
Properties	733
ServerEngine Property.....	733
TnxBaseFieldsValidationDescriptor Class	734
TnxBaseFieldValidationDescriptor Class	734
Methods	735
CheckValid Method.....	735
GetList Method.....	735
IsEqual Method.....	735
LoadFromReader Method.....	736
SaveToWriter Method.....	736
Properties	736
ApplyAt Property.....	736
Name Property	737
TnxBaseHeapDescriptor Class	737
Destructors.....	737
Destroy Destructor.....	737
Methods	738
AddBlockHeapDescriptor Method.....	738
CheckValid Method.....	738
IsEqual Method.....	739
LoadFromReader Method.....	739
RemoveBlockHeapDescriptor Method.....	739
SaveToWriter Method.....	740
Properties	740
BlockHeapDescriptor Property.....	740
HeapEngine Property.....	740
TnxBaseIndicesDescriptor Class	741
Constructors.....	741
Create Constructor.....	741
Destructors.....	742
Destroy Destructor.....	742
Methods	742
AddIndex	742
AddIndex Method (TnxIndexDescriptor).....	742
AddIndex Method (string, Integer, Boolean, string, TnxKeyDescriptorClass, TnxIndexDescriptorClass).....	742
CheckValid Method.....	743
Clear Method	743
GetDescriptorFromName Method.....	744
GetIndexFromName Method.....	744
InsertIndexAt Method.....	744
IsEqual Method.....	745
LoadFromReader Method.....	746
MoveIndex Method.....	746
RemoveIndex	746

RemoveIndex Method (Integer).....	746
RemoveIndex Method (string).....	747
SaveToWriter Method.....	747
Properties	748
HeaderSection Property.....	748
TnxBaseKeyDescriptor Class	748
Destructors.....	748
Destroy Destructor.....	748
Methods	749
LoadFromReader Method.....	749
SaveToWriter Method.....	749
Properties	749
KeyLen Property.....	749
TnxBaseLog Class	750
Methods	750
Clear Method	750
Flush Method	751
WriteBlock Method.....	751
WriteLogData Method.....	751
WriteString	752
WriteString Method (string).....	752
WriteString Method (string, array of const).....	752
WriteStrings Method.....	752
Properties	753
Enabled Property.....	753
TnxBasePluginCommandHandler Class	753
Methods	753
UISettingsVisible Method.....	753
Properties	754
CommandHandler Property.....	754
TnxBasePooledTransport Class	754
Constructors.....	755
Create Constructor.....	755
Destructors.....	755
Destroy Destructor.....	755
Methods	755
Broadcast Method.....	755
ConnectionCount Method.....	756
EstablishConnection Method.....	756
GetConfigSettings Method.....	756
GetNetIdle Method.....	757
GetStatsCaptions Method.....	757
GetStatsValues Method.....	758
IsConnected Method.....	758
LoadConfig Method.....	758
LoadSettingsFromStream Method.....	759
Post Method	759
Request Method.....	759
SaveConfig Method.....	760
SaveSettingsToStream Method.....	760
SetNetIdle Method.....	761
TerminateConnection Method.....	761
Write Method	761
Properties	762

CallbackThreadCount Property.....	762
CallbackThreadPriority Property.....	762
CompressLimit Property	762
CompressType Property	763
ConcurrentIOPThreads Property.....	763
HeartbeatInterval Property	763
HeartbeatThreadPriority Property	764
OverlappedClient Property	764
Port Property	764
ServerThreadPriority Property	765
WatchdogInterval Property.....	765
WatchdogThreadPriority Property	765
TnxBaseRecordCompressionDescriptor Class	766
Methods	766
IsEqual Method.....	766
LoadFromReader Method.....	766
SaveToWriter Method.....	767
Properties	767
RecordCompressionEngine Property	767
TnxBaseRecordDescriptor Class	768
Destructors.....	768
Destroy Destructor.....	768
Methods	768
CheckValid Method.....	768
IsEqual Method.....	769
LoadFromReader Method.....	769
SaveToWriter Method.....	769
Properties	770
FileNumber Property	770
HeaderSection Property.....	770
RecordEngine Property	770
TnxBaseServer Class	771
Constructors.....	771
Create Constructor.....	771
Destructors.....	772
Destroy Destructor.....	772
Methods	772
ActivateAll Method.....	772
Authenticate Method.....	772
ClearAllStats Method.....	773
DeActivateAll Method.....	773
LoadSettings Method.....	773
SaveSettings Method.....	774
StartStoppedModules Method.....	774
StopStartedModules Method.....	775
Events	775
OnError Property.....	775
OnShow Login Property.....	775
Properties	776
AutoSaveConfig Property.....	776
IsServerActive Property.....	776
LoginTries Property.....	777
MaxUserCount Property.....	777
OnLoginError Property.....	777

Password Property.....	778
SecureServer Property.....	778
SecurityMonitor Property.....	779
ServerConfiguration Property.....	779
Username Property.....	779
TnxBaseServerConfiguration Class	780
Constructors.....	780
Create Constructor.....	780
Methods	781
EnsureComponentConfiguration Method.....	781
GetComponentConfiguration Method.....	781
TnxBaseServerEngine Class	781
Constructors.....	782
Create Constructor.....	782
Destructors.....	782
Destroy Destructor.....	782
Methods	783
ClearStats Method.....	783
GetStatsCaptions Method.....	783
GetStatsValues Method.....	783
SessionOpen Method.....	784
SessionOpenEx Method.....	784
Properties.....	785
CursorCount Property.....	785
DatabaseCount Property.....	785
ServerGuid Property.....	786
SessionCount Property.....	786
StatementCount Property.....	786
TransContextCount Property.....	787
TnxBaseSession Class	787
Constructors.....	787
Create Constructor.....	787
Destructors.....	788
Destroy Destructor.....	788
Methods	788
AddAlias Method.....	788
AddAliasEx Method.....	789
CancelProcessing Method.....	789
CloseInactiveFolders Method.....	790
CloseInactiveTables Method.....	790
DefaultDatabase Method.....	790
DefaultSession Method.....	791
DefaultTransContext Method.....	791
DeleteAlias Method.....	791
DeleteAliasEx Method.....	792
FindDatabase Method.....	792
GetAliasNames Method.....	793
GetAliasNamesEx Method.....	793
GetAliasPath Method.....	794
GetAliasPathEx Method.....	794
GetChangedTables Method.....	795
GetRegisteredClassList Method.....	795
GetStoredProcNames Method.....	796
GetStoredProcParams Method.....	796

GetTableNames Method.....	797
IsAlias Method	797
IsConnected Method.....	798
ModifyAlias Method.....	798
ModifyAliasEx Method.....	799
OpenDatabase Method.....	799
PasswordAdd Method.....	800
PasswordRemove Method.....	800
PasswordRemoveAll Method.....	800
TableExists Method.....	801
Events	801
OnLogin Property.....	801
Properties.....	802
AbstractSession Property.....	802
BeepOnLoginError Property.....	802
CurrentUserName Property.....	802
DatabaseCount Property.....	803
Databases Property.....	803
Default Property.....	804
Password Property.....	804
PasswordRetries Property.....	804
ServerEngine Property.....	805
ServerVersion Property.....	805
Timeout Property.....	805
TransContextCount Property.....	806
TransContexts Property.....	806
UserName Property.....	806
TnxBaseSessionPool Class	807
Constructors.....	807
Create Constructor.....	807
Destructors.....	808
Destroy Destructor.....	808
Methods	808
AcquireSession Method.....	808
Properties.....	808
Password Property.....	808
ServerEngine Property.....	809
Timeout Property.....	809
UserName Property.....	809
TnxBaseSetting Class	810
Properties.....	810
DefaultValue Property.....	810
EnforceValues Property.....	811
Hint Property	811
Max Property	811
Min Property	811
Name Property	812
PropertyName Property.....	812
SettingType Property.....	812
ValueList Property.....	813
TnxBaseSettings Class	813
Methods	813
AddSetting Method.....	813
Properties	814

Count Property	814
Setting Property	814
TnxBaseStreamDescriptor Class	814
Methods	815
IsEqual Method	815
LoadFromReader Method	815
SaveToWriter Method	816
Properties	816
HeaderSection Property	816
StreamEngine Property	816
TnxBaseTableDescriptor Class	817
Constructors	817
Create Constructor	817
Destructors	817
Destroy Destructor	817
Methods	818
AddAutoIncDescriptor Method	818
AddBlobDescriptor Method	818
AddIndicesDescriptor Method	819
AddRecordDescriptor Method	819
AssignOnlyFields Method	820
CheckValid Method	820
Clear Method	820
EnsureIndicesDescriptor Method	821
FindRelatedDescriptorOfType Method	821
GetChild Method	821
GetConstraintByName Method	822
GetConstraints Method	822
GetFieldFromName Method	822
GetIndexFromName Method	823
IsEqual Method	823
IsIndexDescValid Method	824
LoadFromReader Method	824
RemoveAutoIncDescriptor Method	824
RemoveBlobDescriptor Method	825
RemoveIndicesDescriptor Method	825
RemoveRecordDescriptor Method	825
SaveToWriter Method	826
Properties	826
AutoIncDescriptor Property	826
BlobDescriptor Property	827
BookmarkSize Property	827
CustomDescsDescriptor Property	827
FieldsDescriptor Property	828
IndicesDescriptor Property	828
KeyLen Property	828
RecordDescriptor Property	829
TablesDescriptor Property	829
TnxBaseTransport Class	829
Constructors	830
Create Constructor	830
Destructors	830
Destroy Destructor	830
Methods	831

BeginBroadcast Method.....	831
Broadcast Method.....	831
btDoCallBack Method.....	832
ClearStats Method.....	832
ConnectionCount Method.....	833
CurrentTransport Method.....	833
EstablishConnection Method.....	833
GetBoundAddresses Method.....	834
GetConfigSettings Method.....	834
GetName Method.....	835
GetServerNames Method.....	835
GetSessionCallback Method.....	835
GetStatsCaptions Method.....	836
GetStatsValues Method.....	836
IsConnected Method.....	836
LoadConfig Method.....	837
LoadSettingsFromStream Method.....	837
Post Method	837
ProtocolName Method.....	838
Reply Method	838
Request Method.....	839
ResetMsgCount Method.....	839
SaveConfig Method.....	840
SaveSettingsToStream Method.....	840
Sleep Method	840
TerminateConnection Method.....	841
Write Method	841
Events	842
OnAddSession Property	842
OnChooseServer Property.....	843
OnConnectionLost Property.....	843
OnFindServers Property	843
OnRemoveSession Property.....	844
Properties.....	844
ActualBytesReceived Property.....	844
ActualBytesSent Property.....	844
CommandHandler Property.....	845
CompressTime Property.....	845
EventLogOptions Property	845
KeepStats Property.....	846
MaxBytesPerSecond Property.....	846
MessagesReceived Property.....	846
MessagesSent Property.....	847
Mode Property	847
MsgCount Property.....	847
PingTime Property.....	847
RespondToBroadcasts Property.....	848
ServerGUID Property.....	848
ServerName Property.....	848
ServerNameDesigntime Property	849
ServerNameRuntime Property.....	849
ThreadPriorityMax Property.....	849
ThreadPriorityMin Property.....	850
Timeout Property.....	850

UncompressedBytesReceived Property.....	850
UncompressedBytesSent Property.....	851
UncompressTime Property.....	851
TnxBaseTransportWrapper Class	851
TnxBaseUpdateObject Class	852
TnxBatchAppendBlobStream Class	852
Destructors.....	853
Destroy Destructor.....	853
Methods	853
Read Method	853
Seek Method	853
Truncate Method.....	854
Write Method	854
TnxBlobStream Class	855
Destructors.....	855
Destroy Destructor.....	855
Methods	856
Read Method	856
Seek	856
Seek Method (Int64, TSeekOrigin).....	856
Seek Method (Integer, Word).....	857
Truncate Method.....	857
Write Method	857
Properties	858
ChunkSize Property.....	858
TnxBlockModeBlobStream Class	858
Methods	859
Read Method	859
Seek Method	859
Truncate Method.....	860
Write Method	860
TnxBroadcastReply Class	861
Constructors.....	861
Create	861
Create Constructor (string, TnxGuid, Integer).....	861
Create Constructor (string, string, string).....	861
TnxCachedDataSet Class	862
Constructors.....	862
Create Constructor.....	862
Destructors.....	863
Destroy Destructor.....	863
Events	863
OnCreateTable Property.....	863
Properties	863
AddIndexDefs Property.....	863
BatchSize Property.....	864
FieldDefs Property.....	864
IndexDefs Property	864
IndexFieldCount Property	865
IndexFieldNames Property.....	865
IndexFields Property.....	865
IndexName Property	866
KeyExclusive Property.....	866
KeyFieldCount Property.....	866

KeyPartialLen Property.....	867
MasterFields Property.....	867
MasterSource Property.....	867
Options Property.....	868
ReadOnly Property.....	868
SourceDataSet Property.....	868
TableName Property.....	869
UpdateObject Property.....	869
UsedTableName Property.....	869
TnxCompKeyDescriptor Class	870
Constructors.....	870
Destroy Destructor.....	870
Methods	870
Add Method	870
CheckValid Method.....	871
Clear Method	871
Delete Method	871
IsEqual Method.....	872
LoadFromReader Method.....	872
SaveToWriter Method.....	873
Properties	873
KeyFieldCount Property.....	873
KeyFields Property.....	873
TnxComponent Class	874
Constructors.....	874
Create Constructor.....	874
Destructors.....	875
Destroy Destructor.....	875
Methods	875
BeforeDestruction Method.....	875
ClearStats Method.....	875
FindClassRecursive Method.....	876
FreeInstance Method.....	876
GetConfigSettings Method.....	876
GetStatsCaptions Method.....	877
GetStatsValues Method.....	877
IterateDependents Method.....	877
LoadConfig Method.....	878
LoadSettingsFromStream Method.....	878
New Instance Method.....	878
SaveConfig Method.....	879
SaveSettingsToStream Method.....	879
Supported Method.....	879
UISettingsVisible Method.....	880
UIStatsVisible Method.....	880
Properties	880
DisplayCategory Property.....	880
DisplayName Property.....	881
Version Property.....	881
TnxConstDefaultValueDescriptor Class	881
Methods	882
IsEqual Method.....	882
LoadFromReader Method.....	882
SaveToWriter Method.....	882

Properties.....	883
AsVariant Property.....	883
TnxCurrentDateTimeDefaultValueDescriptor Class	883
TnxCurrentUserDefaultValueDescriptor Class	883
TnxCursor Class	884
Constructors.....	884
Create Constructor.....	884
Properties.....	885
Database Property	885
TnxCustomConnectionFilter Class	885
Events	885
OnAcceptConnection Property.....	885
TnxCustomDescsDescriptor Class	886
Constructors.....	886
Create Constructor.....	886
Destructors.....	887
Destroy Destructor.....	887
Methods	887
AddCustom Method.....	887
CheckValid Method.....	887
Clear Method	888
GetCustomDescriptorFromName Method.....	888
IsEqual Method.....	888
LoadFromReader Method.....	889
RemoveCustom.....	889
RemoveCustom Method (Integer).....	889
RemoveCustom Method (string).....	890
SaveToWriter Method.....	890
TnxDataAccessState Component Class	890
TnxDatabase Class	891
Constructors.....	891
Create Constructor.....	891
Destructors.....	892
Destroy Destructor.....	892
Methods	892
BackupTable Method.....	892
BackupTableEx Method.....	892
ChangePassw ord Method.....	893
ChangePassw ordEx Method.....	893
Commit Method.....	894
CreateTable Method.....	894
CreateTableEx Method.....	895
DeleteTable Method.....	896
EmptyTable Method.....	896
ExecQuery Method.....	897
ExecStoredProc Method.....	897
Exists Method	897
GetAutolncValue Method.....	898
GetChangedTables Method.....	898
GetDataDictionary Method.....	899
GetDataDictionaryEx Method.....	899
GetFreeDiskSpace Method.....	900
GetFreeDiskSpaceEx Method.....	900
GetPath Method.....	900

GetStoredProcNames Method.....	901
GetStoredProcParams Method.....	901
GetTableNames Method.....	902
OpenQuery Method.....	902
OpenStoredProc Method.....	903
OpenTable Method.....	903
PackTable Method.....	904
PackTableEx Method.....	904
RecoverTable Method.....	905
RecoverTableEx Method.....	905
RelIndexTable	906
RelIndexTable Method (string, string, Integer).....	906
RelIndexTable Method (string, string, string).....	906
RelIndexTableEx.....	907
RelIndexTableEx Method (string, string, Integer,	907
TnxAbstractTaskInfo).....	907
RelIndexTableEx Method (string, string, string,	908
TnxAbstractTaskInfo).....	908
RenameTable Method.....	908
RestructureTable Method.....	909
RestructureTableEx Method.....	910
Rollback Method.....	910
StartTransaction Method.....	911
StartTransactionWith Method.....	911
StartTransactionWithEx Method.....	912
TableExists Method.....	913
TransactionCorrupted Method.....	913
TryStartTransaction Method.....	913
Properties.....	914
AliasName Property.....	914
AliasPath Property.....	914
DataSetCount Property.....	915
DataSets Property.....	915
Default Property.....	915
Exclusive Property.....	916
FailSafe Property.....	916
Implicit Property	916
InTransaction Property.....	917
Path Property	917
Queries Property.....	917
QueryCount Property.....	918
ReadOnly Property.....	918
TableCount Property.....	918
Tables Property.....	919
TransContext Property.....	919
TnxDataDictionary Class	920
Constructors.....	920
Create Constructor.....	920
Destructors.....	920
Destroy Destructor.....	920
Methods	921
AddStreamDescriptor Method.....	921
Assign Method.....	921
AssignOnlyFields Method.....	922
CheckReadOnly Method.....	922

CheckValid Method.....	922
FindRelatedDescriptorOfType Method.....	923
GetFileFromExt Method.....	923
IsEqual Method.....	923
MakeReadOnly Method.....	924
ReadFromReader Method.....	924
ReadFromStream Method.....	925
RemoveStreamDescriptor Method.....	925
UsableBlockSize Method.....	925
WriteToStream Method.....	926
WriteToWriter Method.....	926
Properties	927
FilesDescriptor Property.....	927
StreamDescriptor Property.....	927
Version Property.....	927
TnxDataMessageReader Class	928
Constructors.....	928
Create Constructor.....	928
Destructors.....	928
Destroy Destructor.....	928
Properties	929
DataMessageStream Property.....	929
TnxDataMessageStream Class	929
Constructors.....	930
Create Constructor.....	930
TnxDataset Class	930
Constructors.....	930
Create Constructor.....	930
Destructors.....	931
Destroy Destructor.....	931
Methods	931
AddFileBlob Method.....	931
AddFileBlobEx Method.....	932
AfterConstruction Method.....	932
BatchPost Method.....	932
BeginBatchAppend Method.....	933
BookmarkValid Method.....	933
CompareBookmarks Method.....	934
CopyRecords Method.....	934
CopyRecordsEx Method.....	935
CreateBlobStream Method.....	935
DeleteRecords Method.....	936
DeleteStream Method.....	936
EndBatchAppend Method.....	937
Exists Method	937
FlushBatchAppend Method.....	937
GetAutoIncValue Method.....	938
GetBookmark Method.....	938
GetCurrentRecord Method.....	939
GetFieldData Method.....	939
GetStreamList Method.....	939
GotoCurrent Method.....	940
InBatchAppend Method.....	940
IsRecordLocked Method.....	941

IsSequenced Method.....	941
IsTableLocked Method.....	941
KeyAsVariant Method.....	942
Locate Method	942
LockTable Method.....	942
Lookup Method.....	943
Post Method	943
PSEndTransaction Method.....	944
PSExecuteStatement Method.....	944
PSGetQuoteChar Method.....	944
PSInTransaction Method.....	945
PSIsSqlBased Method.....	945
PSIsSqlSupported Method.....	945
PSReset Method.....	945
PSStartTransaction Method.....	946
ReadStream Method.....	946
ReadStreamEx Method.....	947
RecordCountAsync Method.....	947
SetAutoIncValue Method.....	948
UnlockTable Method	948
UnlockTableAll Method.....	949
WriteStream Method.....	949
Events	950
OnServerFilterTimeout Property.....	950
Properties	950
_Dictionary Property.....	950
AbstractCursor Property.....	950
ActiveDesigntime Property.....	951
ActiveRuntime Property.....	951
AfterCancel Property.....	951
AfterClose Property	952
AfterDelete Property	952
AfterEdit Property.....	952
AfterInsert Property	953
AfterOpen Property.....	953
AfterPost Property	953
AfterRefresh Property	954
AfterScroll Property	954
AliasName Property.....	954
AutoCalcFields Property.....	955
BeforeCancel Property.....	955
BeforeClose Property.....	955
BeforeDelete Property	955
BeforeEdit Property.....	956
BeforeInsert Property	956
BeforeOpen Property.....	956
BeforePost Property	957
BeforeRefresh Property	957
BeforeScroll Property	957
BlockReadOptions Property.....	958
Database Property	958
FieldsDescriptor Property.....	958
Filter Property	959
Filtered Property.....	959

FilterOptions Property.....	959
FilterResync Property.....	960
FilterTimeout Property.....	960
FilterType Property.....	960
FlipOrder Property.....	961
OnCalcFields Property.....	961
OnDeleteError Property.....	961
OnEditError Property.....	962
OnFilterRecord Property.....	962
OnNew Record Property.....	962
OnPostError Property.....	962
Options Property.....	963
Session Property.....	963
SimpleExpressionFilterClass Property.....	963
SqlFilterClass Property.....	964
TableDescriptor Property.....	964
Timeout Property.....	964
Version Property.....	965
TnxDictionaryItem Class	965
Constructors.....	965
Destroy Destructor.....	965
Methods	966
Assign Method.....	966
CheckValid Method.....	966
FindDescriptorClass Method.....	966
FindParentOfType Method.....	967
FindRelatedDescriptorOfType Method.....	967
GetParentOfType Method.....	968
GetRelatedDescriptorOfType Method.....	968
IsEqual Method.....	968
LoadFromReader Method.....	969
SaveToWriter Method.....	969
Properties	970
ConstraintName Property.....	970
CustomStrings Property.....	970
HasCustomStrings Property.....	970
ID Property	971
TnxEmptyDefaultValueDescriptor Class	971
TnxExtendableServerObject Class	971
Constructors.....	972
Create Constructor.....	972
Destructors.....	972
Destroy Destructor.....	972
Methods	973
AfterConstruction Method.....	973
BeforeDestruction Method.....	973
esoOptionClear Method.....	973
esoOptionGetEffective Method.....	974
esoOptionSet Method.....	974
GetExtenderOfType Method.....	974
NotifyExtenders Method.....	975
OptionClear Method.....	975
OptionGet Method.....	976
OptionGetEffective Method.....	976

OptionList Method.....	976
OptionListEffective Method.....	977
OptionSet Method.....	977
TnxExtTextKeyFieldDescriptor Class	977
Methods	978
LoadFromReader Method.....	978
SaveToWriter Method.....	978
TnxFieldDescriptor Class	979
Destructors.....	979
Destroy Destructor.....	979
Methods	979
AddDefaultValue Method.....	979
AddValidations Method.....	980
CheckValid Method.....	980
EnsureValidations Method.....	980
FindRelatedDescriptorOfType Method.....	981
IsEqual Method.....	981
LoadFromReader Method.....	981
RemoveDefaultValue Method.....	982
RemoveValidations Method.....	982
SaveToWriter Method.....	982
SetupField Method.....	983
UpdateSetup Method.....	983
Properties	984
BufferAsVariant Property.....	984
Name Property	984
Number Property	984
TnxFieldsDescriptor Class	985
Constructors.....	985
Create Constructor.....	985
Destructors.....	986
Destroy Destructor.....	986
Methods	986
AddField Method.....	986
AddValidations Method.....	987
CheckValid Method.....	987
Clear Method	987
EnsureValidations Method.....	988
fsdFieldChangeNotification Method.....	988
GetFieldForFilter Method.....	988
GetFieldFromName Method.....	989
GetFields Method.....	989
GetRecordField Method.....	990
GetRecordFieldForFilter.....	990
GetRecordFieldForFilter Method (Integer, PnxByteArray).....	990
GetRecordFieldForFilter Method (Integer, PnxByteArray, TnxFieldType, TnxWord16, Pointer).....	991
HasAutolncField Method.....	991
HasBlobFields Method.....	992
HasRecRevField Method.....	992
HasSameFields Method.....	992
InitRecord Method.....	993
InsertFieldAt Method.....	993
IsEqual Method.....	994

IsRecordFieldNull Method.....	994
LoadFromReader Method.....	995
MoveField Method.....	995
RemoveField Method.....	995
RemoveValidations Method.....	996
SaveToWriter Method.....	996
SetRecordField Method.....	996
SetRecordFieldNull Method.....	997
UpdateLogRecLenAlign Method.....	997
UpdateSetupAndOffsets Method.....	998
Properties.....	998
FieldByIndexAsVariant Property.....	998
FieldNameAsVariant Property.....	998
LogicalRecordLength Property.....	999
RecordLength Property.....	999
Validations Property.....	999
TnxFieldsValidationsDescriptor Class	1000
Methods.....	1000
AddValidation Method.....	1000
TnxFieldValidationsDescriptor Class	1000
Constructors.....	1001
Create Constructor.....	1001
Destructors.....	1001
Destroy Destructor.....	1001
Methods.....	1002
AddValidation Method.....	1002
CheckValid Method.....	1002
Clear Method	1002
GetValidationDescriptorFromName Method.....	1003
isEqual Method.....	1003
LoadFromReader Method.....	1003
RemoveValidation.....	1004
RemoveValidation Method (Integer).....	1004
RemoveValidation Method (string).....	1004
SaveToWriter Method.....	1005
TnxFileDescriptor Class	1005
Methods.....	1005
CheckValid Method.....	1005
isEqual Method.....	1006
LoadFromReader Method.....	1006
SaveToWriter Method.....	1006
Properties.....	1007
BlockSize Property.....	1007
BlockSizeBytes Property.....	1007
Desc Property	1008
Extension Property.....	1008
Grow Size Property.....	1008
InitialSize Property.....	1009
Number Property	1009
UsableBlockSize Property.....	1009
TnxFilesDescriptor Class	1010
Constructors.....	1010
Create Constructor.....	1010
Destructors.....	1010

Destroy Destructor.....	1010
Methods.....	1011
AddFile Method.....	1011
CheckValid Method.....	1011
Clear Method	1012
GetFileFromExt Method.....	1012
InsertFileAt Method.....	1012
IsEqual Method.....	1013
LoadFromReader Method.....	1013
MoveFile Method.....	1014
RemoveFile Method.....	1014
SaveToWriter Method.....	1015
TnxGetServerNamesHandler Class	1015
Constructors.....	1015
Create Constructor.....	1015
Destructors.....	1016
Destroy Destructor.....	1016
Methods.....	1016
GetServerNames Method.....	1016
TnxHeapBlobDescriptor Class	1017
Constructors.....	1017
Create Constructor.....	1017
Destructors.....	1017
Destroy Destructor.....	1017
Methods.....	1018
AddHeapDescriptor Method.....	1018
CheckValid Method.....	1018
IsEqual Method	1018
LoadFromReader Method.....	1019
RemoveHeapDescriptor Method.....	1019
SaveToWriter Method.....	1020
Properties	1020
HeapDescriptor Property.....	1020
TnxHeapRecordDescriptor Class	1020
Destructors.....	1021
Destroy Destructor.....	1021
Methods.....	1021
AddHeapDescriptor Method.....	1021
AddRecordCompressionDescriptor Method.....	1022
CheckValid Method.....	1022
IsEqual Method	1022
LoadFromReader Method.....	1023
RemoveHeapDescriptor Method.....	1023
RemoveRecordCompressionDescriptor Method.....	1023
SaveToWriter Method.....	1024
Properties	1024
HeapDescriptor Property.....	1024
RecordCompressionDescriptor Property.....	1025
TnxIndexDataSet Class	1025
Constructors.....	1025
Create Constructor.....	1025
Destructors.....	1026
Destroy Destructor.....	1026
Methods.....	1026

ApplyRange Method.....	1026
Cancel Method.....	1026
CancelRange Method.....	1027
EditKey Method.....	1027
EditRangeEnd Method.....	1027
EditRangeStart Method.....	1028
FindKey	1028
FindKey Method (array of const).....	1028
FindKey Method (array of const, TnxSetKeyOptions).....	1028
FindNearest	1029
FindNearest Method (array of const).....	1029
FindNearest Method (array of const, TnxSetKeyOptions).....	1029
GetIndexNames Method.....	1030
GotoKey Method.....	1030
GotoNearest Method.....	1030
GotoNearestBackward Method.....	1031
Post Method	1031
SetKey Method.....	1031
SetRange	1032
SetRange Method (array of const, array of const).....	1032
SetRange Method (array of const, array of const, TnxSetKeyOptions).....	1032
SetRange Method (array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions).....	1032
SetRangeEnd Method.....	1033
SetRangeShared.....	1033
SetRangeShared Method (array of const, array of const, array of const, TnxSetKeyOptions).....	1033
SetRangeShared Method (array of const, array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions).....	1034
SetRangeSimple Method.....	1034
SetRangeStart Method.....	1034
TnxIndexDescriptor Class	1035
Constructors.....	1035
CreateStandalone Constructor.....	1035
Destructors.....	1036
Destroy Destructor.....	1036
Methods.....	1036
CheckValid Method.....	1036
CreateStandaloneFromReader Method.....	1036
IsEqual Method.....	1037
LoadFromReader Method.....	1037
SaveStandalone Method.....	1037
SaveToWriter Method.....	1038
Properties.....	1038
Desc Property	1038
Dups Property	1038
FileNumber Property.....	1039
IndexEngine Property.....	1039
IndexFile Property.....	1039
KeyDescriptor Property.....	1040
Name Property.....	1040
Number Property	1040
TnxINIComponentConfiguration Class	1041
Constructors.....	1041

Create Constructor.....	1041
Destructors.....	1041
Destroy Destructor.....	1041
Methods.....	1042
GetValue Method.....	1042
GetValueStream Method.....	1042
SetValue Method.....	1042
SetValueStream Method.....	1043
TnxINIConfiguration Class	1043
Constructors.....	1044
Create Constructor.....	1044
Destructors.....	1044
Destroy Destructor.....	1044
Methods.....	1044
EnsureComponentConfiguration Method.....	1044
Flush Method	1045
GetComponentConfiguration Method.....	1045
TnxIPv4Transport Class	1045
Methods.....	1046
ProtocolName Method.....	1046
TnxIterationList Class	1046
Methods.....	1047
Add Method	1047
LoadSettingsFromStream Method.....	1047
SaveSettingsToStream Method.....	1047
TnxKeyAsVariantField Class	1048
Constructors.....	1048
Create Constructor.....	1048
TnxKeyFieldDescriptor Class	1048
Destructors.....	1049
Destroy Destructor.....	1049
Methods.....	1049
CheckValid Method.....	1049
IsEqual Method.....	1049
LoadFromReader Method.....	1050
SaveToWriter Method.....	1050
Properties.....	1051
Ascend Property.....	1051
Desc Property.....	1051
Field Property	1051
FieldNumber Property.....	1052
KeyChars Property.....	1052
KeyFieldEngine Property	1052
KeyLength Property	1053
NullBehaviour Property.....	1053
OverrideLengthBytes Property	1053
OverrideLengthChars Property.....	1054
TnxLocaleDescriptor Class	1054
Methods.....	1054
CheckValid Method.....	1054
IsEqual Method.....	1055
LoadFromReader Method.....	1055
SaveToWriter Method.....	1055
Properties	1056

CompareCodepage Property.....	1056
Flags Property.....	1056
IgnoreKanaType Property.....	1056
IgnoreNonSpace Property.....	1057
IgnoreSymbols Property.....	1057
IgnoreWidth Property.....	1057
Locale Property.....	1058
OverrideStorageCodepage Property.....	1058
StorageCodepage Property.....	1058
UseStringSort Property	1059
TnxLocalizedDictionaryItem Class	1059
Destructors.....	1059
Destroy Destructor.....	1059
Methods.....	1060
AddLocaleDescriptor Method.....	1060
CheckValid Method.....	1060
FindRelatedDescriptorOfType Method.....	1061
IsEqual Method.....	1061
RemoveLocaleDescriptor Method.....	1061
Properties.....	1062
LocaleDescriptor Property	1062
UsedLocaleDescriptor Property.....	1062
UsedStorageCodePage Property	1063
TnxLockedOptionsExtendableServerObject Class	1063
Destructors.....	1063
Destroy Destructor.....	1063
Methods.....	1064
esoOptionClear Method.....	1064
esoOptionSet Method.....	1064
TnxLoggableComponent Class	1064
Methods.....	1065
IcLog	1065
IcLog Method (array of string).....	1065
IcLog Method (string).....	1065
IcLog Method (string, , TnxMemSize).....	1066
IcLog Method (string, array of const).....	1066
Properties.....	1066
EventLog Property.....	1066
EventLogEnabled Property.....	1067
TnxMainIndicesDescriptor Class	1067
Methods.....	1067
CheckValid Method.....	1067
Clear Method	1068
IsEqual Method.....	1068
LoadFromReader Method.....	1069
SaveToWriter Method.....	1069
Properties.....	1069
DefaultIndex Property.....	1069
TnxMemoBlobStream Class	1070
Destructors.....	1070
Destroy Destructor.....	1070
Methods.....	1071
Read Method	1071
Seek Method	1071

Truncate Method.....	1071
Write Method	1072
TnxMemoField Class	1072
Properties.....	1073
AsWideString Property.....	1073
TnxMemTable Class	1073
Constructors.....	1073
Create Constructor.....	1073
Destructors.....	1074
Destroy Destructor.....	1074
Events	1074
OnCreateTable Property.....	1074
Properties.....	1075
FieldDefs Property.....	1075
IndexDefs Property	1075
IndexFieldCount Property.....	1075
IndexFieldNames Property.....	1076
IndexFields Property.....	1076
IndexName Property.....	1076
KeyExclusive Property.....	1076
KeyFieldCount Property.....	1077
KeyPartialLen Property.....	1077
MasterFields Property.....	1077
MasterSource Property.....	1078
StoreDefs Property	1078
TableName Property.....	1078
UsedTableName Property.....	1079
TnxMinMaxValidationDescriptor Class	1079
Destructors.....	1080
Destroy Destructor.....	1080
Methods.....	1080
AddKeyField Method.....	1080
CheckValid Method.....	1080
IsEqual Method.....	1081
LoadFromReader Method.....	1081
RemoveKeyField Method.....	1081
SaveToWriter Method.....	1082
Properties.....	1082
KeyField Property.....	1082
Max Property	1082
MaxAsVariant Property.....	1083
Min Property	1083
MinAsVariant Property.....	1083
TnxNestedTableDescriptor Class	1084
Constructors.....	1084
Create Constructor.....	1084
Properties	1084
Name Property.....	1084
TnxNoChangeValidationDescriptor Class	1085
Destructors.....	1085
Destroy Destructor.....	1085
Methods.....	1086
AddKeyField Method.....	1086
CheckValid Method.....	1086

IsEqual Method.....	1086
LoadFromReader Method.....	1087
RemoveKeyField Method.....	1087
SaveToWriter Method.....	1087
Properties.....	1088
KeyField Property.....	1088
TnxNoNullCompKeyDescriptor Class	1088
Methods.....	1088
CheckValid Method.....	1088
LoadFromReader Method.....	1089
SaveToWriter Method.....	1089
TnxNullActiveBroadcast Class	1089
TnxPersistent Class	1090
Methods.....	1090
FreeInstance Method.....	1090
New Instance Method.....	1091
TnxPooledSession Class	1091
Destructors.....	1091
Destroy Destructor.....	1091
Methods.....	1092
AddRef Method.....	1092
Release Method.....	1092
Properties.....	1092
SessionPool Property.....	1092
TnxQuery Class	1093
Constructors.....	1093
Create Constructor.....	1093
Destructors.....	1093
Destroy Destructor.....	1093
Methods.....	1094
ExecSQL Method.....	1094
Properties	1094
ParamCheck Property.....	1094
SQL Property	1094
Text Property	1095
TnxRefKeyDescriptor Class	1095
Methods.....	1095
LoadFromReader Method.....	1095
SaveToWriter Method.....	1096
TnxRegisterableComponent Class	1096
Methods.....	1097
FindRegisteredClass Method.....	1097
GetRegisteredClasses Method.....	1097
TnxServer Class	1097
Properties	1098
CommandHandler Property.....	1098
EnabledModules Property.....	1098
EventLog Property.....	1098
EventLogEnabled Property.....	1099
NamedPipeTransport Property.....	1099
SecurityMonitor Property.....	1099
ServerEngine Property.....	1100
ServerName Property.....	1100
SQLEngine Property.....	1100

WinsockTransport Property	1101
TnxServerManager Class	1101
Properties	1101
ServerEngine Property	1101
TnxSession Class	1102
Destructors	1102
Destroy Destructor	1102
Properties	1102
ServerEngine Property	1102
TnxSessionOwnedDataAccessStateComponent Class	1103
Constructors	1103
Create Constructor	1103
Methods	1104
AfterConstruction Method	1104
Properties	1104
Session Property	1104
Timeout Property	1104
TnxSessionPool Class	1105
Destructors	1105
Destroy Destructor	1105
Properties	1105
ServerEngine Property	1105
TnxSetting Class	1106
Methods	1106
Assign Method	1106
TnxSettings Class	1106
Constructors	1107
Create Constructor	1107
Destructors	1107
Destroy Destructor	1107
Methods	1108
AddSetting Method	1108
TnxSimpleLock Class	1108
Constructors	1108
Create Constructor	1108
Destructors	1109
Destroy Destructor	1109
Methods	1109
Failed Method	1109
Lock Method	1109
Unlock Method	1110
TnxSqlCheckValidationDescriptor Class	1110
Methods	1110
IsEqual Method	1110
LoadFromReader Method	1111
SaveToWriter Method	1111
Properties	1112
Constraint Property	1112
ErrorMessage Property	1112
TnxSqlUpdateObject Class	1112
Constructors	1113
Create Constructor	1113
Destructors	1113
Destroy Destructor	1113

Properties.....	1113
DeleteSql Property.....	1113
InsertSql Property.....	1114
ModifySql Property.....	1114
Params Property.....	1114
Sql Property	1115
TnxStateComponent Class	1115
Constructors.....	1116
Create Constructor.....	1116
Destructors.....	1116
Destroy Destructor.....	1116
Methods.....	1116
BeforeDestruction Method.....	1116
Close Method	1117
Connect Method.....	1117
GetConfigSettings Method.....	1117
GetStatsCaptions Method.....	1118
GetStatsValues Method.....	1118
LoadConfig Method.....	1118
LoadSettingsFromStream Method.....	1119
Open Method	1119
SaveConfig Method.....	1119
SaveSettingsToStream Method.....	1120
SwitchToSavedActive Method.....	1120
UIStateVisible Method.....	1120
Properties.....	1121
Active Property	1121
ActiveDesignTime Property.....	1121
ActiveRuntime Property.....	1121
Connected Property	1122
Enabled Property.....	1122
OnStateChanged Property	1122
OnStateTransChanged Property.....	1123
SavedActive Property.....	1123
StartTime Property.....	1123
State Property	1124
StateTransition Property.....	1124
TnxStatementDataSet Class	1124
Constructors.....	1125
Create Constructor.....	1125
Destructors.....	1125
Destroy Destructor.....	1125
Methods.....	1125
ParamByName Method.....	1125
Prepare Method.....	1126
Unprepare Method.....	1126
Properties.....	1126
DataSource Property.....	1126
Log Property	1127
ParamCount Property.....	1127
Params Property.....	1127
Prepared Property.....	1128
ReadOnly Property.....	1128
RecordsRead Property.....	1128

RequestLive Property.....	1129
Row sAffected Property.....	1129
TnxStoredProcedure Class	1129
Methods.....	1130
ExecProc Method.....	1130
Prepare Method.....	1130
RefreshParam Method.....	1130
Properties.....	1131
StoredProcName Property.....	1131
TnxTable Class	1131
Constructors.....	1131
Create Constructor.....	1131
Destructors.....	1132
Destroy Destructor.....	1132
Methods.....	1132
AddIndex Method.....	1132
AddIndexEx Method.....	1133
ChangePassword Method.....	1133
CreateTable Method.....	1134
CreateTableEx Method.....	1134
DeleteIndex Method.....	1134
DeleteTable Method.....	1135
EmptyTable Method.....	1135
Exists Method.....	1135
PackTable Method.....	1136
PackTableEx Method.....	1136
RelIndexTable	1136
RelIndexTable Method (Integer).....	1136
RelIndexTable Method (string).....	1137
RelIndexTableEx.....	1137
RelIndexTableEx Method (Integer, TnxAbstractTaskInfo).....	1137
RelIndexTableEx Method (string, TnxAbstractTaskInfo).....	1138
RenameTable Method.....	1138
RestructureTable Method.....	1139
RestructureTableEx Method.....	1139
Events	1139
OnCreateTable Property.....	1139
Properties.....	1140
Exclusive Property.....	1140
FieldDefs Property.....	1140
IndexDefs Property.....	1140
IndexFieldCount Property.....	1141
IndexFieldNames Property.....	1141
IndexFields Property.....	1141
IndexName Property.....	1142
KeyExclusive Property.....	1142
KeyFieldCount Property.....	1142
KeyPartialLen Property.....	1143
MasterFields Property.....	1143
MasterSource Property.....	1143
Password Property.....	1143
ReadOnly Property.....	1144
StoreDefs Property.....	1144
TableName Property.....	1144

WriteOnly Property.....	1145
TnxTableDescriptor Class	1145
TnxTablesDescriptor Class	1145
Constructors.....	1146
Create Constructor.....	1146
Destructors.....	1146
Destroy Destructor.....	1146
Methods.....	1147
AddTable Method.....	1147
CheckValid Method.....	1147
Clear Method	1147
GetTableDescriptorFromName Method.....	1148
GetTableIndexFromName Method.....	1148
GetTables Method.....	1148
IsEqual Method.....	1149
LoadFromReader Method.....	1149
RemoveTable	1149
RemoveTable Method (Integer).....	1149
RemoveTable Method (string).....	1150
SaveToWriter Method.....	1150
TnxTextKeyFieldDescriptor Class	1151
Methods.....	1151
IsEqual Method.....	1151
LoadFromReader Method.....	1151
SaveToWriter Method.....	1152
Properties	1152
IgnoreCase Property.....	1152
TnxThreadWithDatabase Class	1153
Constructors.....	1153
Create Constructor.....	1153
Properties.....	1153
Database Property.....	1153
TnxThreadWithSession Class	1154
Constructors.....	1154
Create Constructor.....	1154
Properties.....	1155
Session Property.....	1155
TnxThreadWithTable Class	1155
Constructors.....	1155
Create Constructor.....	1155
Properties	1156
Table Property.....	1156
TnxTransContext Class	1156
Constructors.....	1157
Create Constructor.....	1157
Destructors.....	1157
Destroy Destructor.....	1157
Methods.....	1157
Commit Method.....	1157
Rollback Method.....	1158
StartTransaction Method.....	1158
StartTransactionWith Method.....	1159
StartTransactionWithEx Method.....	1159
TransactionCorrupted Method.....	1160

TryStartTransaction Method.....	1160
Properties.....	1161
DatabaseCount Property.....	1161
Databases Property.....	1161
Default Property.....	1162
FailSafe Property.....	1162
InTransaction Property.....	1162
TnxWinsockTransport Class	1163
Constructors.....	1163
Create Constructor.....	1163
Destructors.....	1164
Destroy Destructor.....	1164
Methods.....	1164
BeginBroadcast Method.....	1164
GetBoundAddresses Method.....	1164
GetConfigSettings Method.....	1165
GetName Method.....	1165
LoadConfig Method.....	1165
LoadSettingsFromStream Method.....	1166
SaveConfig Method.....	1166
SaveSettingsToStream Method.....	1166
Properties.....	1167
BroadcastThreadPriority Property.....	1167
CallbackThreadCount Property.....	1167
CallbackThreadPriority Property.....	1167
ConnectionFilter Property.....	1168
ListenAddresses Property.....	1168
ListenThreadPriority Property.....	1168
2 Interfaces.....	1169
InxActiveBroadcast Interface	1169
Methods.....	1169
Resume Method.....	1169
Stop Method	1170
Suspend Method.....	1170
InxBroadcastReply Interface	1170
Methods.....	1171
GetServerID Method.....	1171
GetServerName Method.....	1171
GetServerVersion Method.....	1171
InxBroadcastReplyHandler Interface	1172
Methods.....	1172
ReplyReceived Method.....	1172
InxFieldDescriptor Interface	1172
Methods.....	1173
GetFieldDecPI Method.....	1173
GetFieldType Method.....	1173
GetFieldUnits Method.....	1173
Properties.....	1174
FieldDecPI Property.....	1174
FieldType Property	1174
FieldUnits Property	1174
InxFieldsSource Interface	1175
Methods.....	1175
BlobsSupported Method.....	1175

GetCursor Method.....	1175
GetFieldAsVariant Method.....	1176
GetFieldCount Method.....	1176
GetFieldDescriptor Method.....	1176
GetFieldForFilter Method.....	1177
GetFieldFromName Method.....	1177
Properties.....	1178
FieldCount Property.....	1178
FieldDescriptor Property.....	1178
InxLogData Interface	1178
Methods.....	1179
GetCategory Method.....	1179
GetMsg Method.....	1179
GetMsgNumber Method.....	1179
GetPriority Method.....	1180
GetUTCWhen Method.....	1180
GetWhen Method.....	1180
GetWhenBias Method.....	1181
Properties.....	1181
Category Property.....	1181
Msg Property	1181
MsgNumber Property.....	1182
Priority Property.....	1182
UTCWhen Property.....	1182
When Property.....	1182
WhenBias Property.....	1183
InxSessionCallback Interface	1183
Methods.....	1184
Post Method	1184
Request Method.....	1184
3 Functions.....	1185
_DoNothingStub Function	1185
_DoNothingStub2 Function	1185
nxCheckValidAliasName Function	1186
nxCheckValidRelativeTableName Function	1186
nxCheckValidRootTableName Function	1186
nxCheckValidStoredProcName Function	1187
nxCheckValidTableName Function	1187
nxSplitAddress Function	1188
nxTableNameIsTempDatabase Function	1188
nxTableNameIsTempGlobal Function	1188
nxTableNameIsTempStatement Function	1189
4 Structs, Records, Enums.....	1189
Tnx1xFileType Enumeration	1189
TnxApplyAt Enumeration	1190
TnxBaseHeader Record	1190
TnxBlobCopyMode Enumeration	1191
TnxBookmark Record	1192
TnxCachedDataSetOption Enumeration	1193
TnxClassListType Enumeration	1193
TnxDataMessage Record	1194
TnxDataSetInternalState Enumeration	1194
TnxDataSetOption Enumeration	1195

TnxEngineAction Enumeration	1195
TnxFieldType Enumeration	1197
TnxFilterType Enumeration	1198
TnxIndexPathPosition Enumeration	1199
TnxKeyBuffer Record	1199
TnxKeyIndex Enumeration	1200
TnxLockConflictType Enumeration	1201
TnxLockPresent Enumeration	1201
TnxLockRequestType Enumeration	1201
TnxLockResult Enumeration	1202
TnxLockType Enumeration	1202
TnxLogPriority Enumeration	1203
TnxMessageHeaderFlag Enumeration	1203
TnxNullBehaviour Enumeration	1204
TnxOpenMode Enumeration	1204
TnxParamType Enumeration	1205
TnxRecordCountOption Enumeration	1205
TnxRecordGetBatchExOption Enumeration	1206
TnxSearchKeyAction Enumeration	1206
TnxServerModule Enumeration	1207
TnxSetKeyOption Enumeration	1207
TnxSettingType Enumeration	1208
TnxShareMode Enumeration	1208
TnxSqlParamDesc Record	1209
TnxState Enumeration	1210
TnxStatementExecDirectPhase Enumeration	1210
TnxStatementType Enumeration	1211
TnxStateTransition Enumeration	1211
TnxTableScope Enumeration	1212
TnxTaskStatus Record	1212
TnxTransportLogOption Enumeration	1213
TnxTransportMode Enumeration	1213
5 Types.....	1214
EnxComponentExceptionClass Type	1214
PnxBaseHeader Type	1214
PnxBaseSession Type	1215
PnxBookmark Type	1215
PnxDataMessage Type	1215
PnxKeyBuffer Type	1216
PnxLockResult Type	1216
PnxSqlParamDesc Type	1216
PnxSqlParamList Type	1217
PnxTaskStatus Type	1217
TFieldDefClass Type	1218
TFormatSettings Type	1218
TNotifyErrorEvent Type	1218
TnxAbstractServerObjectClass Type	1219
TnxAbstractSessionClass Type	1219
TnxAddSessionEvent Type	1219
TnxApplyAtSet Type	1220
TnxAutoIncStepRange Type	1220
TnxBaseAutoIncDescriptorClass Type	1221
TnxBaseBlobDescriptorClass Type	1221
TnxBaseBlockHeapDescriptorClass Type	1221

TnxBaseCustomDescriptorClass Type	1222
TnxBaseDefaultValueDescriptorClass Type	1222
TnxBaseEngineExtenderClass Type	1223
TnxBaseFieldsValidationDescriptorClass Type	1223
TnxBaseFieldValidationDescriptorClass Type	1223
TnxBaseHeapDescriptorClass Type	1224
TnxBaseIndicesDescriptorClass Type	1224
TnxBasePooledTransportClass Type	1224
TnxBaseRecordCompressionDescriptorClass Type	1225
TnxBaseRecordDescriptorClass Type	1225
TnxBaseStreamDescriptorClass Type	1225
TnxBaseTableDescriptorClass Type	1226
TnxBaseTransportClass Type	1226
TnxBaseUpdateHandlerClass Type	1227
TnxBlockSignature Type	1227
TnxCachedDataSetOptions Type	1227
TnxChooseServerEvent Type	1228
TnxConnectionLostEvent Type	1228
TnxCreateTableEvent Type	1229
TnxCursorClass Type	1229
TnxCursorID Type	1229
TnxCustomDescriptorClass Type	1230
TnxDatabaseID Type	1230
TnxDataSetInternalStates Type	1230
TnxDataSetOptions Type	1231
TnxDictionaryItem Class Type	1231
TnxDirectKeySetFieldsMethod Type	1232
TnxEngineActions Type	1232
TnxFieldDescriptorClass Type	1232
TnxFieldsDescriptorClass Type	1233
TnxFieldsValidationsDescriptorClass Type	1233
TnxFieldType Type	1233
TnxFieldValidationsDescriptorClass Type	1234
TnxFileDescriptorClass Type	1234
TnxFilesDescriptorClass Type	1234
TnxFilterID Type	1235
TnxFindServersEvent Type	1235
TnxGrowSize Type	1236
TnxHeapBlobDescriptorClass Type	1236
TnxHeapRecordDescriptorClass Type	1236
TnxIndexDescriptorClass Type	1237
TnxIterationListClass Type	1237
TnxKeyDescriptorClass Type	1237
TnxKeyFieldDescriptorClass Type	1238
TnxKeyFilterID Type	1238
TnxLocaleDescriptorClass Type	1238
TnxLoginCallback Type	1239
TnxLoginEvent Type	1239
TnxLogPriorities Type	1240
TnxMainIndicesDescriptorClass Type	1240
TnxMappingMethod Type	1240
TnxMessageHeaderFlags Type	1241
TnxNestedTableDescriptorClass Type	1241
TnxNetIdleEvent Type	1242

TnxOnAcceptConnection Type	1242
TnxPersistentClass Type	1242
TnxRecordGetBatchExOptions Type	1243
TnxRegisterableComponentClass Type	1243
TnxRemoveSessionEvent Type	1243
TnxReplyCallback Type	1244
TnxServerFilterTimeOutEvent Type	1244
TnxServerModules Type	1245
TnxSetKeyOptions Type	1245
TnxSqlParamList Type	1246
TnxStatementID Type	1246
TnxTablesDescriptorClass Type	1246
TnxTaskID Type	1247
TnxTransContextID Type	1247
TnxTransID Type	1247
TnxTransportLogOptions Type	1248
6 Variables.....	1248
nxGetFailedFlag Variable	1248
nxLockTimeOut Variable	1248
nxUnlockTimeOut Variable	1249
7 Constants.....	1249
nxatAliasName Constant	1249
nxatAliasPath Constant	1250
nxc_MagicNumber_Cursor Constant	1250
nxc_MagicNumber_Database Constant	1250
nxc_MagicNumber_Session Constant	1251
nxc_MagicNumber_Statement Constant	1251
nxc_MagicNumber_TaskInfo Constant	1251
nxc_MagicNumber_TransContext Constant	1252
nxc_MainSettingsExtension Constant	1252
nxc_SigHeaderBlockV1 Constant	1252
nxc_SigHeaderBlockV2 Constant	1253
nxcBlobTypes Constant	1253
nxcDefaultServerSearchTimeOut Constant	1254
nxDefaultFilterTimeOut Constant	1254
nxMaxBlobChunk Constant	1254
8 Files.....	1255
nxBaseServerComp.pas	1255
nxConfigSettings.pas	1255
nxdb.pas	1255
nxlIComponent.pas	1257
nxlITransport.pas	1257
nxptBasePooledTransport.pas	1258
nxsdDataDictionary.pas	1259
nxsdServerEngine.pas	1260
nxsdTypes.pas	1262
nxServerComp.pas	1263
nxServerManager.pas	1264
nxtwWinsockTransport.pas	1264
Part XXVI Supported Visual Studio versions	1265

Part XXVII Installing the NexusDB ADO.NET Provider	1266
Part XXVIII Visual Studio Pack Editions	1268
Part XXIX Choosing the server connection method	1269
Part XXX Single User Mode	1270
Part XXXI Entity Framework Support	1271
Part XXXII Connection String Parameters	1276
Part XXXIII Creating the Northwind sample database	1278
Part XXXIV Creating a simple NexusDB Application	1279
Part XXXV Deploying an Application	1282
Part XXXVI Online Support	1283
Index	1285

www.nexusdb.com

1 Hardware and Software Requirements

Hardware and Software Requirements



NexusDB can run on any windows version from Windows 98 onward, on any PC with a Pentium or better processor. Windows 95 with winsock 2 and DCOM updates (these are downloadable from Microsoft) can run a NexusDB client application only.

For network-based C/S systems the TCP/IP protocol is required or, alternatively, a protocol which supports Named Pipes.

NexusDB Developer and Embedded Server edition currently support Delphi 5, 6, 7, and 2005 and C++ Builder 6. Make sure that the relevant service packs have been installed.

By default NexusDB is compiled with speed-enhancements that requires processors with MMX support (Pentium >133Mhz). If required, MMX can be ifdef'd out to support non-MMX Pentium and Pentium Pro CPU's.



2 NexusDB Architecture

2.1 Historical Context

Historical Context



NexusDB is the third generation of a long line of database management systems. The first generation, BtreeFiler, was implemented in the late 1980's. FlashFiler, a TurboPower product, was the second generation, developed in the mid 90's and was open sourced to the Delphi community in 2003. NexusDB represents the next generation, completely rewritten, building on the heritage from this long-lived line of database management systems.

NexusDB has numerous improvements over its predecessors. The significant improvements NexusDB has made include:

- Snapshot transactions
- Improved SQL - DDL and 2003 compliance
- New sub-engine architecture
- New memory manager
- Improved transports - performance, compression
- Batch reads/inserts
- General ease-of-use enhancements and client simplification .

Snapshot transactions are read-only transactions that eliminate the need for issuing locks within the database. They work by creating copies of the original data blocks when that data has been changed; in essence, database readers do not block database writers without creating any locks. Snapshot transactions greatly enhance overall performance for read-only operations especially SQL selects.

The SQL engine has been greatly enhanced to support DDL (Data Definition Language). NexusDB V1 supports SQL DDL statements such as "create table" and "create index". Version 2 goes even further and supports more than core compliance of SQL 2003 with a large number of useful vendor specific extensions. The full details of the supported SQL syntax are discussed in the SQL Reference chapter of this manual.

NexusDB uses a sub-engine architecture that enables encapsulation of critical database functionality into different modules. This is a significant difference from earlier and other DBMS's. This architecture provides the underpinnings for future extension without modifying the core database engine. NexusDB is truly unique in this regard.

New data transports are included with NexusDB. These transports form a high level messaging system that can be used for different application requirements from a high speed LAN to slow WAN networks. These transports are optimized through extremely advanced techniques (e.g. I/O completion ports) as well as supporting compression and encryption.

Each of these features are covered the manual. Many developers have demonstrated that NexusDB easily beats other DBMS's with significant improvements in performance and functionality. NexusDB has many improvements over, for instance, FlashFiler and represents a significantly improved new generation in this line of databases.

2.2 NexusDB V2 Features

NexusDB V2 Features



Version 2 of NexusDB introduces a number of products. The main core product, the database engine, comes in three editions:

- NexusDB Embedded
- NexusDB C/S Developer
- NexusDB C/S Enterprise

The current Version1 Client/Server product is now called the Developer C/S edition. The Embedded edition is the same as the Developer C/S edition without the transports. Thus the Embedded version can be used for developing desktop single-user applications. To deploy these as multi-user applications typically requires the addition of transports from the C/S edition. The Enterprise edition has extra, Enterprise-level, features such as inbuilt audit trails added to the C/S edition.

NexusDB Embedded and Developer Editions

NexusDB Embedded will retain, feature-wise, the same functionality as the C/S Developer product, except for the transport components. The new features (over and above the Verions 1 functionality) are summarized as follows:

Core Engines:

- Referential integrity
- Full-text engine using single-word indexing
- New fieldtypes
- Variable length keys
- Password-protected tables
- Variable length records
- RecNo support
- SQL supported in TnxDataset.Filter
- Field validation objects
- Conditional indices

SQL Engine:

- UDF's
- Stored Procedures
- Triggers
- Transactional keywords
- Support for multiple databases in queries
- Enhanced live dataset support

Additional to these new features we've worked on a lot of small issues and feature requests making NexusDB what the current developers, have asked for it to be. It is not possible to detail all of these feature enhancements here, but a complete discussion of all features can be gleaned from the Delphi Developer Guide section of this manual.

NexusDB Enterprise Edition

NexusDB Enterprise edition will build upon the features of the Developer edition and include the following:

Core Engines:

- Bulk index operations (i.e. faster and whilst tables are in use)
- Fulltext search with bitmap index (i.e. full support for "and/or" searches and other full expression matching)
- Extended Autoinc sub-engine that reuses deleted keys
- Advanced user security
- Live backup scheduling built into the Nexus Server
- C/S/C Messaging and events
- Database replication
- Auditing

SQL Engine:

- Views
- Sysadmin support (security keywords etc)

NexusDB Memory Manager

All three NexusDB editions include the new Memory Manager V3 which

- out-performs even the current V1 MM
- scales extremely well on multi CPU systems
- includes a finer allocation granularity
- efficiently manages idle threads
- features a new built-in sharemem functionality (without requiring a DLL)

This V3 edition of the NexusDB Memory Manager is available as a separate product.

NexusDB ADO.NET Provider

The NexusDB ADO.NET Provider V2 will be available after the above products have reached final release status, but already has two major new enhancements:

- Visual Query builder
- no need for Win32 DLL in the NexusDB edition for connecting to remote servers

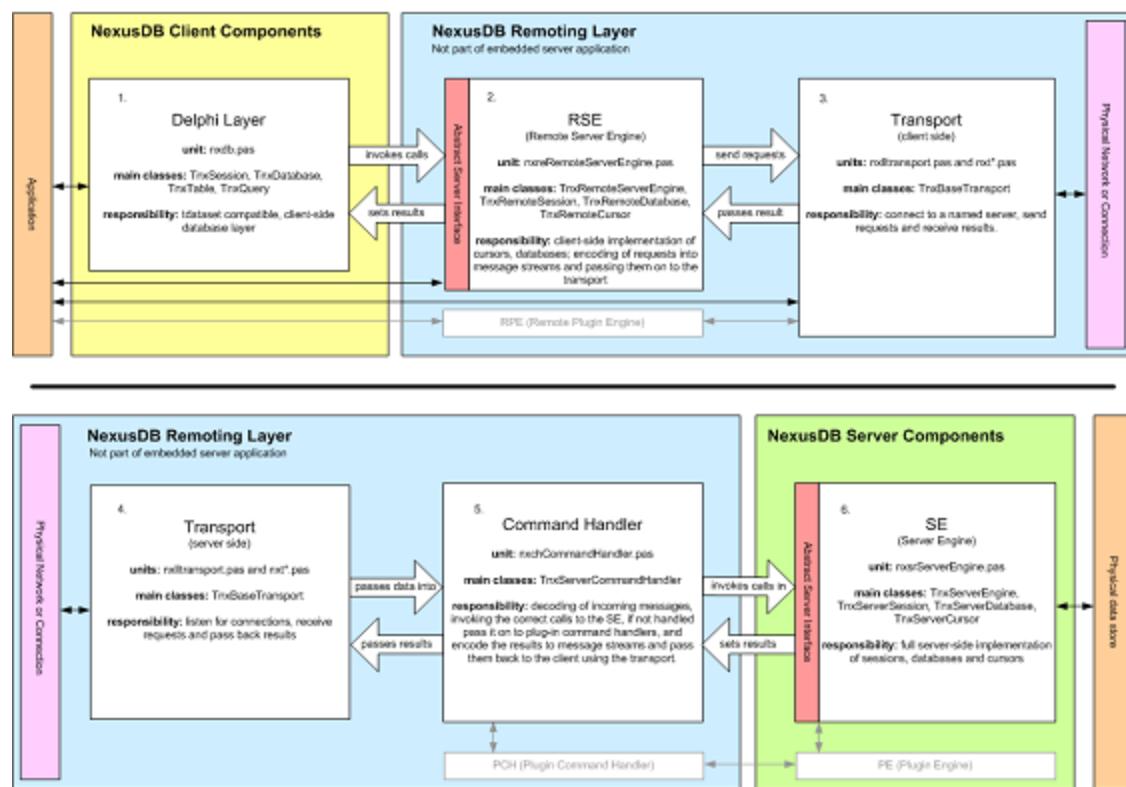


2.3 NexusDB Architecture

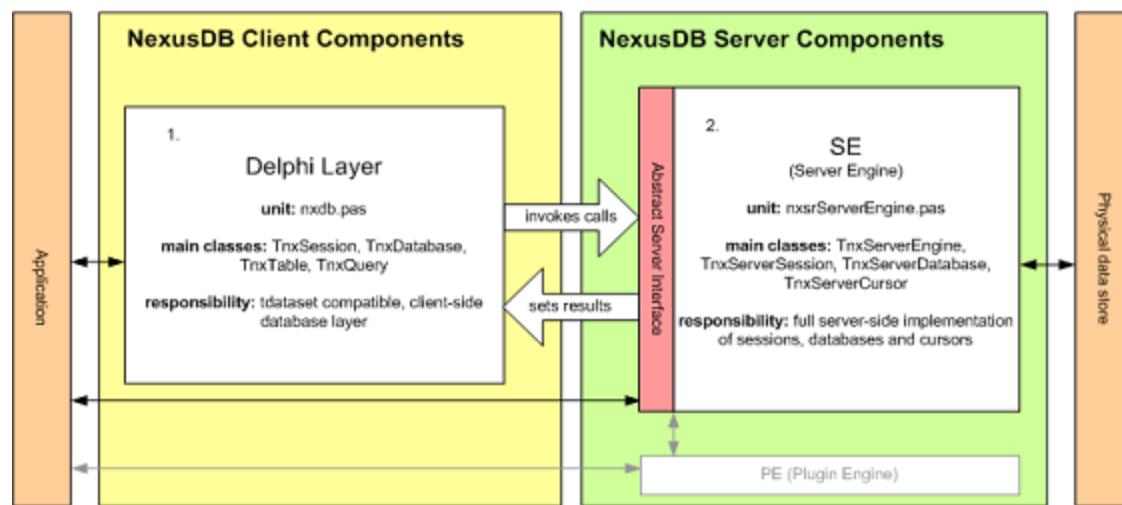
NexusDB Architecture



NexusDB has been fundamentally designed with a Client/Server architecture. It has three distinct layers: client, server and remoting. Each layer has specific responsibilities within this architecture. The Client layer provides high level data access capabilities. The Server layer stores and retrieves data and the Remoting layer connects the Client layer with the Server layer. The overall architecture looks like this:



It is important to note that even as an embedded database, NexusDB preserves the client/server architecture; the only difference is the absence of the Remoting layer. This architecture thus reduces to:



The client layer provides standard TDataSet descendants that are used to build native or web based user interfaces. This layer is easy to use and follows the traditional Borland data access paradigm. The client communicates through a well-defined interface with a server instance.

The remoting layer is responsible for connecting the client layer to the server. It does so through various network protocols: TCP/IP, named pipes or COM. It also supports encryption and compression through each of these protocols. NexusDB is completely extensible and can support any additional protocol.

If the client and server run within the same process the remoting layer is not necessary. Embedding the server within the client process is one of the flexible deployment options NexusDB supports. Deployment options are discussed in depth later in this section.

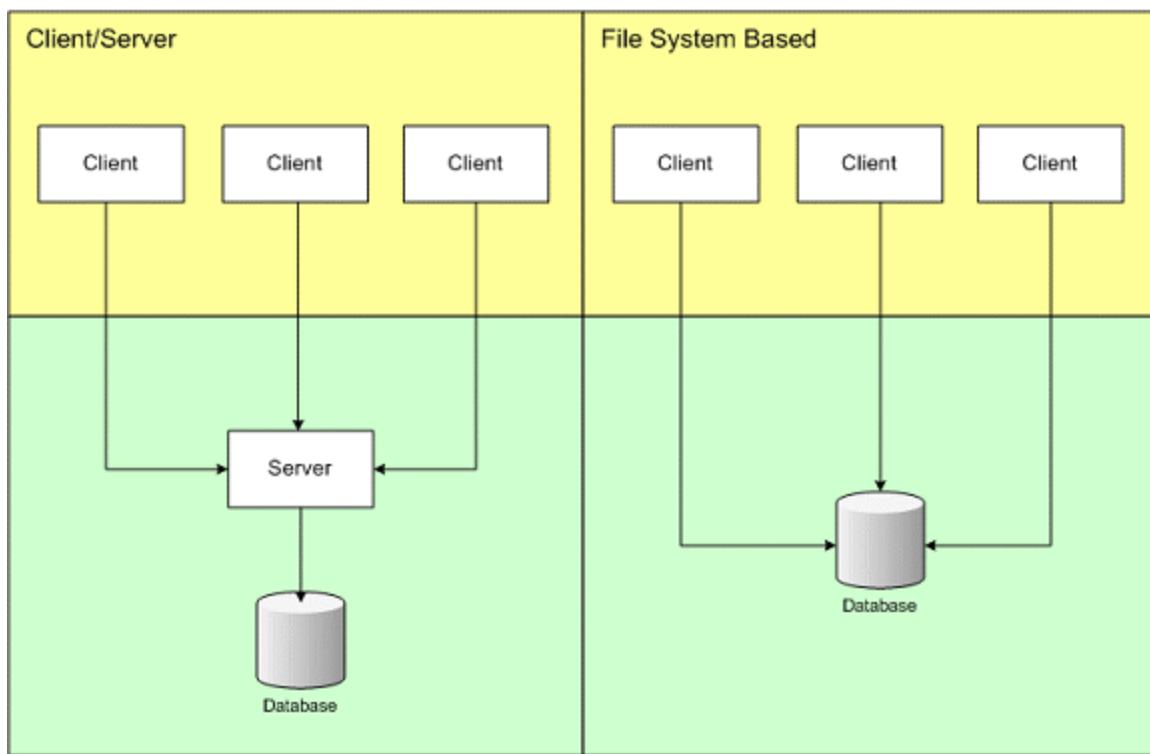
The server layer is the core database engine and is the only layer that directly interacts with the data files. Concurrency control, transaction management and data caching services are efficiently implemented within the server layer. The server layer supports ANSI standard SQL and can be accessed via ODBC or dbExpress APIs.

It is important to contrast NexusDB's client/server architecture with databases that use file-based architectures. In a file-based architecture, there is no separation of client and server, thus each client directly manipulates the data files. There are significant differences in the following areas: concurrency locking mechanisms, caching capabilities, and scalability.

Most database engines that are file-based use file locking through a file system API (e.g. Windows LockFile). These locks are called virtual byte locks; this means that they lock portions of a file that do not physically exist. Each client respects these locks when reading or writing to a particular table.

File-based databases suffer from scalability bottlenecks because of many factors. Network traffic is increased because each client must retrieve the entire files contents because there is no server process to apply the selection criteria. Disk caching of data blocks is not possible because there is no way for each individual client to keep track of data that has changed since the last read. External file locks are much slower than their Client/Server counterparts: in-process synchronization primitives.

The following diagram illustrates the differences:



2.3.1 Client

Client



The Client layer of NexusDB provides access to the data stored on the server. This layer is composed of the following components: TnxSession, TnxDatabase, TnxTable, and TnxQuery. These components support the traditional Borland database paradigm and enable developers to use data-aware controls.

The session component represents a thread specific connection to the server. The session also provides the security context. The TnxSession attaches directly to a server engine component, which may be a remote server engine or the actual server engine embedded within the same application.

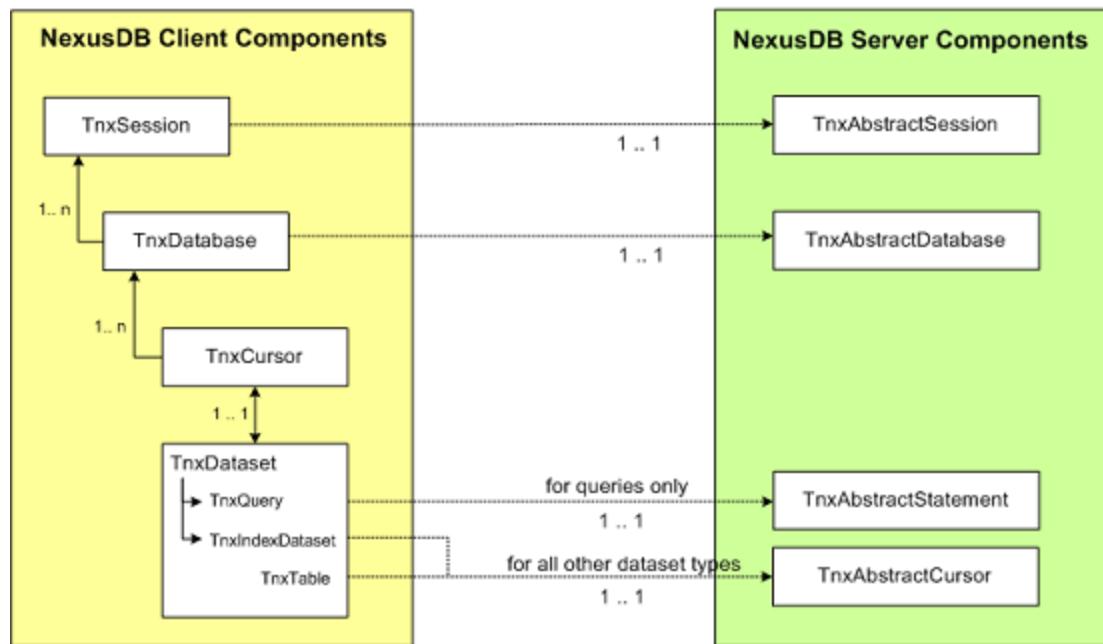
A database specifies a particular server alias or alias path. The TnxDatabase component is associated with a session component. The TnxDatabase is not mandatory since the TnxTable and TnxQuery components allow the developer to specify the alias name directly. For ease of maintenance, use of TnxDatabase is recommended.

Tables and queries are used to access the data in the database and are TDataSet descendants. TnxTable provides optimized navigational access to the underlying server table cursors. TnxQuery

supports ANSI standard SQL for data retrieval, modification and definition. Both components support standard TDatasource components and data-aware controls.

The Client layer components interoperate in this way:

All client side components are dependent on each other and each holds a reference to its server object.
These dependencies are shown here.



The client components work together to coordinate communications with their server side counter parts. As illustrated there is a one to one correspondence between the client and the server's representation of that client. If the server engine is within a separate process, all communications occur through the remoting layer.



2.3.2 Server

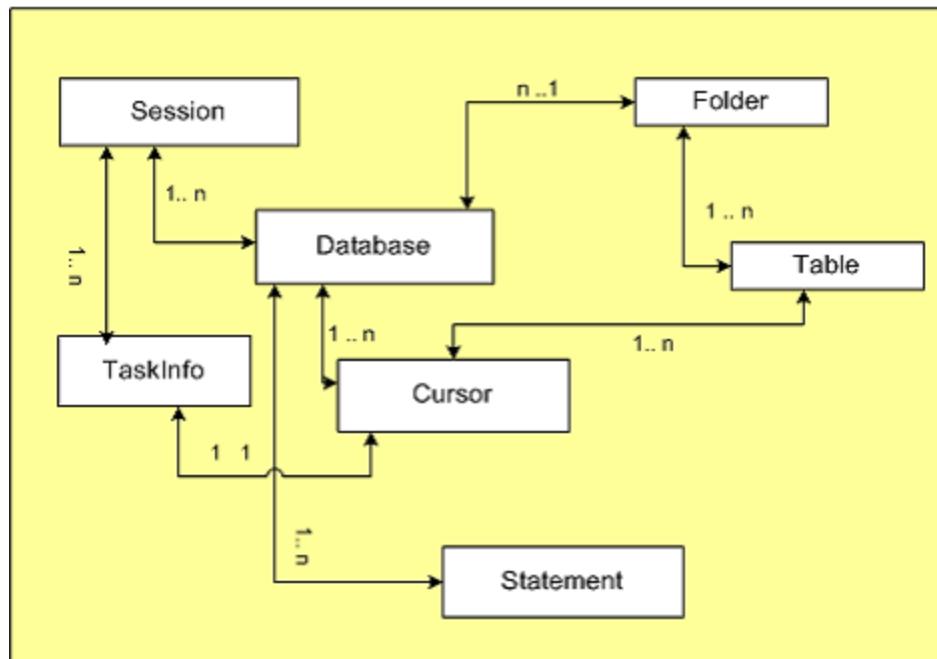
Server



The Server layer is the core of NexusDB and provides transactional data storage and retrieval services. The two main components of this layer are: TnxServerEngine and TnxSQLEngine. Other auxiliary components of this layer are responsible for monitoring server engine activity, providing security and logging events.

The Server Engine provides session, database and cursor support. The Server Engine contains object instances for each opened table and a cursor for each client.

Each table object uses the particular sub-engines that are registered for that table as specified in the data dictionary. These sub-engines communicate with the Buffer Manager to store and retrieve the data. The buffer manager provides transactional support for storage and retrieval of this data.



The SQL Engine parses SQL queries and interacts with the Server Engine to execute those requests. The SQL Engine supports DML, DDL of the ANSI standard. The specific SQL grammar supported is detailed in the SQL Reference chapter.



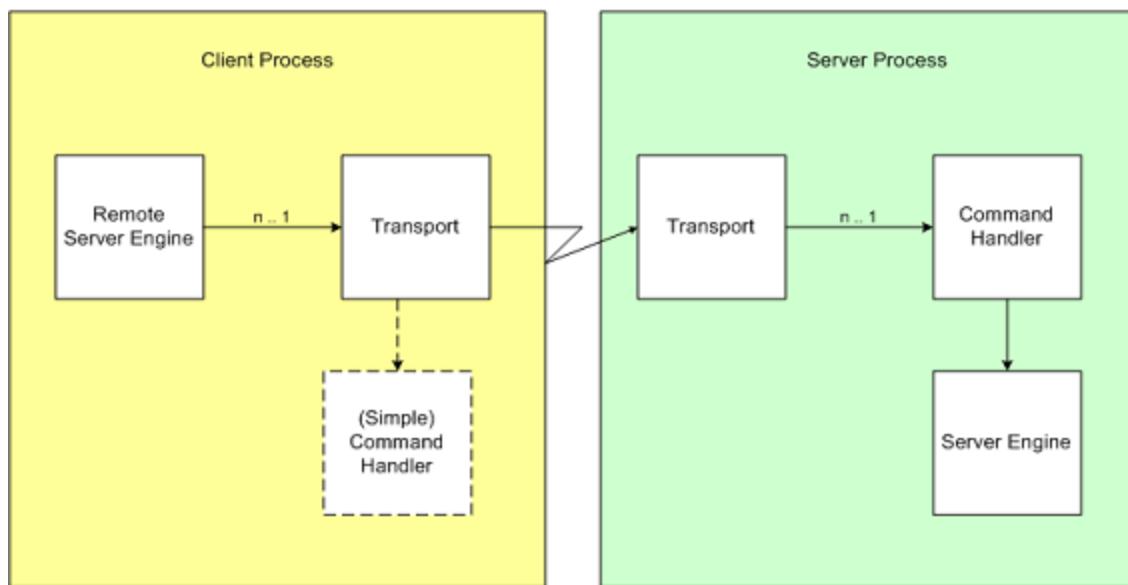
2.3.3 Remoting

Remoting



A remoting layer is responsible for communicating messages between two processes over a network. NexusDB uses a remoting layer to connect its Client and Server layers supporting a variety of network protocols, compression and encryption options.

NexusDB implements this remoting layer with these components: Remote Server Engine, various transport components and the Server Command Handler. The Remote Server Engine is used in the client process, the Server Command Handler in the server process and the transport components in both processes.



The Remote Server Engine (RSE) is a representation of the actual Server Engine. The RSE forwards the Client layer's requests through a pair of transport components to a Server Command Handler. The Server Command Handler dispatches these requests to the actual Server Engine and returns the results back through the transports to the RSE.

The NexusDB transport system is a message based synchronous communication system. Transport components can implement any suitable protocol to send and receive data. NexusDB includes two transport implementations that support TCP/IP and Named Pipes. Transports can also be encrypted or compressed for optimal security and performance.



2.4 Deployment Options

Deployment Options

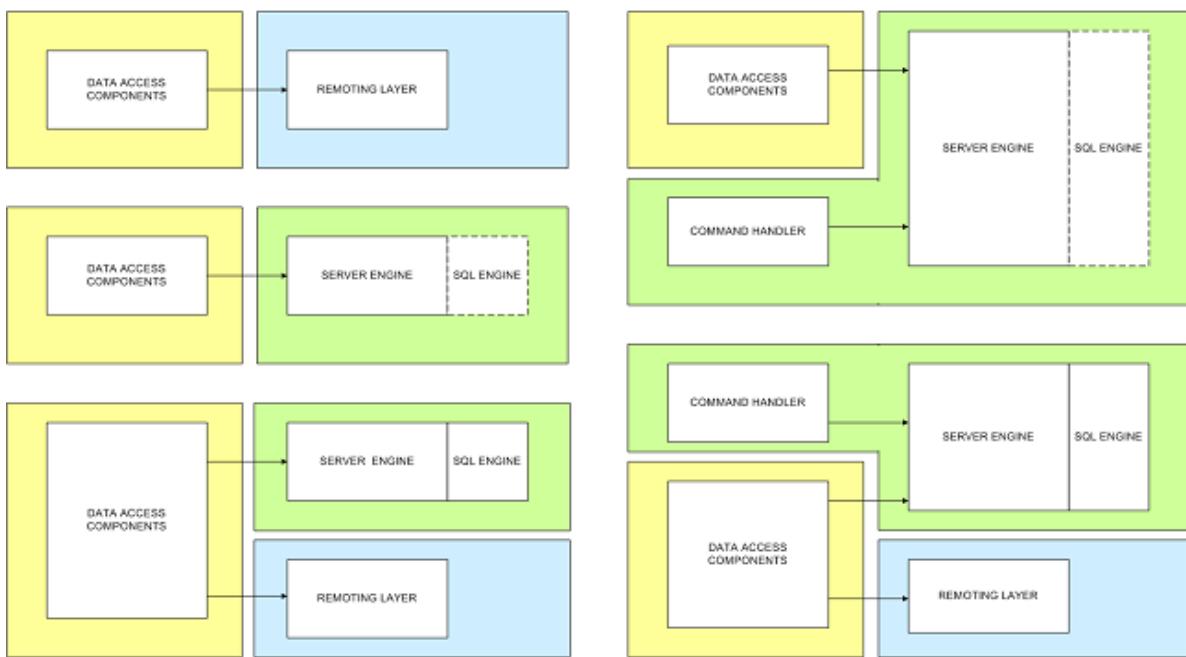


NexusDB is extremely flexible in its deployment options. There are many databases that support embedded deployment or Client/Server deployment; NexusDB supports both deployment options. NexusDB also supports a hybrid mixed deployment option whereby an embedded database engine can be exposed to remote clients.

In an embedded mode, both Client and Server layers are contained within a single process. The remoting layer is not present in an embedded deployment. NexusDB in embedded mode works equally well in web applications such as ISAPI extensions or Apache DSO's.

As a Client/Server deployment NexusDB runs as a separate Server process that networked clients can connect to through the remoting layer. This separate Server process can either be a regular executable or a windows service. The included nxServer.exe can be registered as a windows service.

A Hybrid deployment is a combination of an embedded and client/server deployment. Just as an embedded deployment the Client and Server layers are contained within one process. In a hybrid deployment, this same process exposes a remoting layer transport and includes the server command handler. This remoting layer and command handler enables separate clients to utilize the server engine in exactly the same way as a Client/Server deployment.



As illustrated above, there are many different ways to deploy and utilize NexusDB's layered architecture. Each combination above represent typical deployment combinations that are used to achieve certain application objectives. <diagram should focus on the three deployment options: embedded, c/s and hybrid. Should be typical usages of them. >



3 NexusDB Concepts

3.1 Aliases

Aliases



Aliases are named references to file system paths. An AliasPath is a direct reference to a specific file system path. The TnxDatabase component has an Alias and an AliasPath property. These properties are mutually exclusive. Use the Alias property when Aliases are defined in the nxServer.exe. For embedded deployments, simply specify an AliasPath in the TnxDatabase component.

The nxTrans.cfg file is created in every directory that is configured as a NexusDB alias for two reasons:

- it marks the directory as "in use"
- it stores the highest LSN (Log Sequence Number) of all tables

In general, the file is locked by a server if it has a lock on a database folder (i.e. directory). It will be locked once you have made a TnxDatabase component active. In the default settings this lock stays until the last TnxDatabase instance pointing to this directory is closed and the server decided it's not worthwhile anymore to cache this database. If you want the server to immediately unlock databases and tables, when the last instance pointing to it is closed, you can enforce this on the Server Engine configuration screen on the server by enabling the CloselnactiveTables or CloselnactiveFolders setting. The latter releases the lock on a folder and thus the nxTrans.cfg file when no tables are open.



3.2 AutoInc Fields

AutoInc Fields



NexusDB tracks a "autoinc" value in the file header. This value starts out as 0 on a newly created table. GetAutoIncValue directly returns this value. SetAutoIncValue directly sets this value.

When a record is inserted the server checks if it contains an autoinc field and if the field is still NULL. If yes it increments the autoinc value in the table header and assigned the incremented value to the autoinc field.

That means:

- If you assign (client side) a value to the autoinc field the server side processing will not be done and the value in the file header will not be incremented. It is your responsibility to update that value using SetAutoIncValue.
- You have to make sure that between your call to GetAutoIncValue and later SetAutoIncValue no one else modifies the autoinc value in the header which can happen as a result of either calling SetAutoIncValue or inserting a record into the table that contains NULL in the autoinc field.



3.3 Blob Handling

Blob Handling



Short for Binary Large Object, a collection of binary data stored as a single entity in a database management system. BLOBS are used primarily to hold multimedia objects such as images, videos, and sound, though they can also be used to store programs or even fragments of code.

Are blobs stored internally to the table always loaded into memory?

Like everything else, the blob engine uses the buffer manager to access the table, blobs are stored in "segments" (variable sized parts of blocks allocated through the "heap engine"), these segments are organized in a b-tree with actual blob data stored in the leaf nodes of the tree. If you turn off client layer blob caching each read/write call on a blob stream will be passed through to the blob engine and only the segments that make up the blob between offset and offset+size of the read/write operation (+ the internal nodes of the b-tree to get to them) will be accessed by the blob engine to perform the operation.

Now let's take a detailed look at how blob access works on a ServerEngine level, and later how this interacts with the dataset layer.



3.3.1 Normal Blob Handling Server Side

Normal Blob Handling Server Side



At the server engine level blobs are almost completely independent of records. A quick look at nxsdServerEngine.pas will reveal the functions for blob handling. Each of these functions will start and commit a (implicit) transaction if there is no transaction already active. As a user of NexusDB you normally never call these functions directly, instead the TrxBlobStream class encapsulates the calls to these functions and provides a Delphi typical TStream interface.

In the record buffer each blob field is represented as a 64bit value. The meaning of this value is defined by the specific blob engine used by the table. (Currently there's only one blob engine implemented, but there might be additional ones in the future.)



3.3.2 Normal Blob Handling Client Side

Normal Blob Handling Client Side



Initially the BlobNr stored in the record buffer is 0. When you write to a blob field a 0-length blob is created by calling BlobCreate, the returned BlobNr is stored in the record buffer and BlobWrite is called to write the data to the Blob.

Each blob operation will be performed instantly on the server and, if you don't have an explicit transaction running, it will also be committed to disk instantly.

Advantages

- Each read and write operation is performed on its own. If you work with a 20 MBytes blob but you only read the first 500 bytes, only these 500 bytes will be transferred from server to client. The same applies for writing just a few bytes into an already existing large blob. A potential drawback here is if you do many small reads and writes, each one will send its own message to the server.
- TnxBlobStream uses a so called "ChunkSize". If you try to write more bytes than the set ChunkSize value in a single call, it will make multiple calls to the BlobWrite function, and each will write the changes to disk immediately. This allows you to write a 3 GBytes blob without needing 3GB of RAM or Temporary Storage on the server (which would be needed to store all the dirty pages of one big transaction).

Possible problems

There are a few potential problems to this approach (all assuming you don't have an explicit transaction):

- If you edit or append a record: If you write to an (up to now) empty blob field and then cancel the operation, the blob data will already be written to disk. In this case there will be no record in the table that reference this blob. The only way to recover the space used by such blobs is to pack the table.
- If you edit a record: If you modify an existing blob and cancel the operation these modifications will already be written to disk. Your cancel does only affect the record buffer, not the blobs.
- If you edit a record: If you clear an existing blob and cancel the operation the blob will already be deleted and the BlobNr in the record is now invalid. Trying to access this blob or delete the record will most likely result in a "General Blob Error".



3.3.3 Blob Caching

Blob Caching



Let's now take a look at how Blob Caching changes this behaviour.

When you access a blob while in edit mode, the complete blob is read from the server into an in-memory buffer on the client (only in bmRead and bmReadWrite mode). This obviously rules out using 3GB blobs given that you only have 2GB of address space for your complete process. Any reads or writes to the blob will be done using the local copy of the blob and will not be transferred to the server until you post the whole record.

When you finally post your changes the following happens (if there are any "dirty" in other words changed, added or deleted blobs):

- An explicit transaction is started
- The blobs are written from the local cache to the server, this always writes the complete blob, even if you just changed a few bytes.
- The insert or modify operation is performed on the record.
- Depending on the outcome of the post the transaction is committed or rolled back.

Advantages

- None of the potential problems listed above can happen anymore.
- There is only one large read/write operation per blob you accessed. If your client side code makes many calls to read from or write to the blob stream in small chunks you will see a significant speed increase.

Possible problems

- If you only read/write a few bytes in a large blob you will see a drop in performance because the complete blob is always sent over the network
- If you have very large blobs (a few 100MB or even GB) you might run out of memory on the server (large transactions) or client (local cache)

As most blobs are only a few MB in maximum size and you normally read and/or write the complete blob, blob caching is enabled by default and should only be turned off if you really know what you are sure it's the right decision.

What if you do have an explicit transaction active?

Well, as long as you never commit the transaction if you have

- cancelled any operation involving blobs or
- got an unresolved error on post (e.g. key violation),

you can turn blob caching off. Depending on the exact type of database operation you are performing this might increase or decrease performance.

There is also a 2nd flag in the table options which affects the ways how blob caching works if an explicit transaction is present. In the default setting no nested transaction will be started during post. This improves performance, but can lead to problems if you ever commit an explicit transaction after getting an error on post and cancelling the edit instead of resolving the problem and posting again. (You should rollback the transaction instead.)



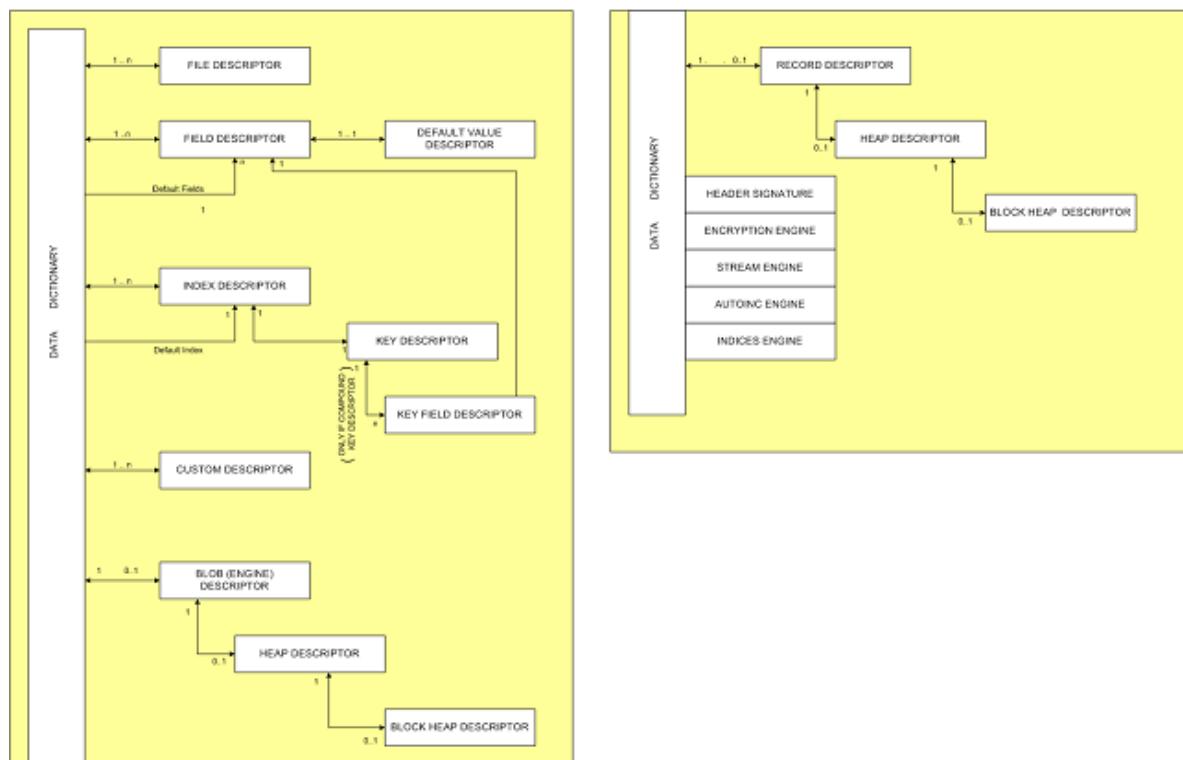
3.4 Data Dictionary

Data Dictionary



Overview

NexusDB uses a Data Dictionary for each table to store information about its structure. The Data Dictionary is a class that contains field definitions, index descriptors, sub-engine registration, default values and data block options. The following diagram illustrates the parts of the Data Dictionary:



The Data Dictionary is the most powerful and flexible way to create and restructure tables in NexusDB. NexusDB uses the Data Dictionary class internally to manage table structures; it is stored in and retrieved from the table itself. The Data Dictionary is the only way sub-engine registration can be specified when creating or restructuring a table. Sub-engines are discussed in the next section.

Blocksize

The BlockSize of a table determines the maximum size of the records you can store, and the maximum size of index keys. The default of 4kb is normally the best choice for speed. It is recommended to only increase BlockSize if you have need of large fields.

As default, a table is created on disk with the minimum number of blocks required. The FileDescriptor's InitialSize property can be increased before creation, if it is desired to preallocate disk space for the table. After the table has been created, the FileDescriptor's GrowSize controls how many blocks are added to the file size when it needs to expand. Increase these properties to keep the file fragmentation down. InitialSize and GrowSize are expressed as numbers of blocks.



3.4.1 Descriptors

Descriptors



Within the dictionary there are arrays of "Descriptors". These "Descriptors" describe various parts of a table and is where applicable sub-engines are registered. NexusDB has the following Descriptors:

- Field - describes the Field data: name, data type, size, required, etc. In addition, it also specifies the Default Value descriptor.
- Index - describes the Index (name, unique, etc.) and contains a Key descriptor to indicate which fields to index. It also has an associated Index sub-engine.
- Key - describes a complete key which can be a composite of many Key Field descriptors. Also specifies the Key sub-engine.
- Key Field - for a specific field specifies the Key Comparison object to use and the various attributes of this field (e.g. Ascending, Ignore Case).
- File - describes various aspects of the file system file: block size, file type, extension, grow/increment size, and initial size.
- Blob - indicates which Blob sub-engine is used for this table and potentially a Heap Descriptor.
- Record - indicates which Record sub-engine is used for this table and potentially a Heap Descriptor.

- Default Value - describes the default value for a particular field.
- Heap - indicates which Heap sub-engine is used and has a Block Heap descriptor.
- Block Heap - indicates which Block Heap sub-engine is used.

The Data Dictionary directly contains arrays of: Field, Index and File descriptors. The Data Dictionary contains a single instance of both a Blob and Record descriptor. Key, Key Field, Default Value, Heap and Block Heap descriptors are not used directly by the Data Dictionary. These descriptors are used by other descriptors as indicated above.



3.4.2 Overview of Sub-Engines

Overview of Sub-Engines



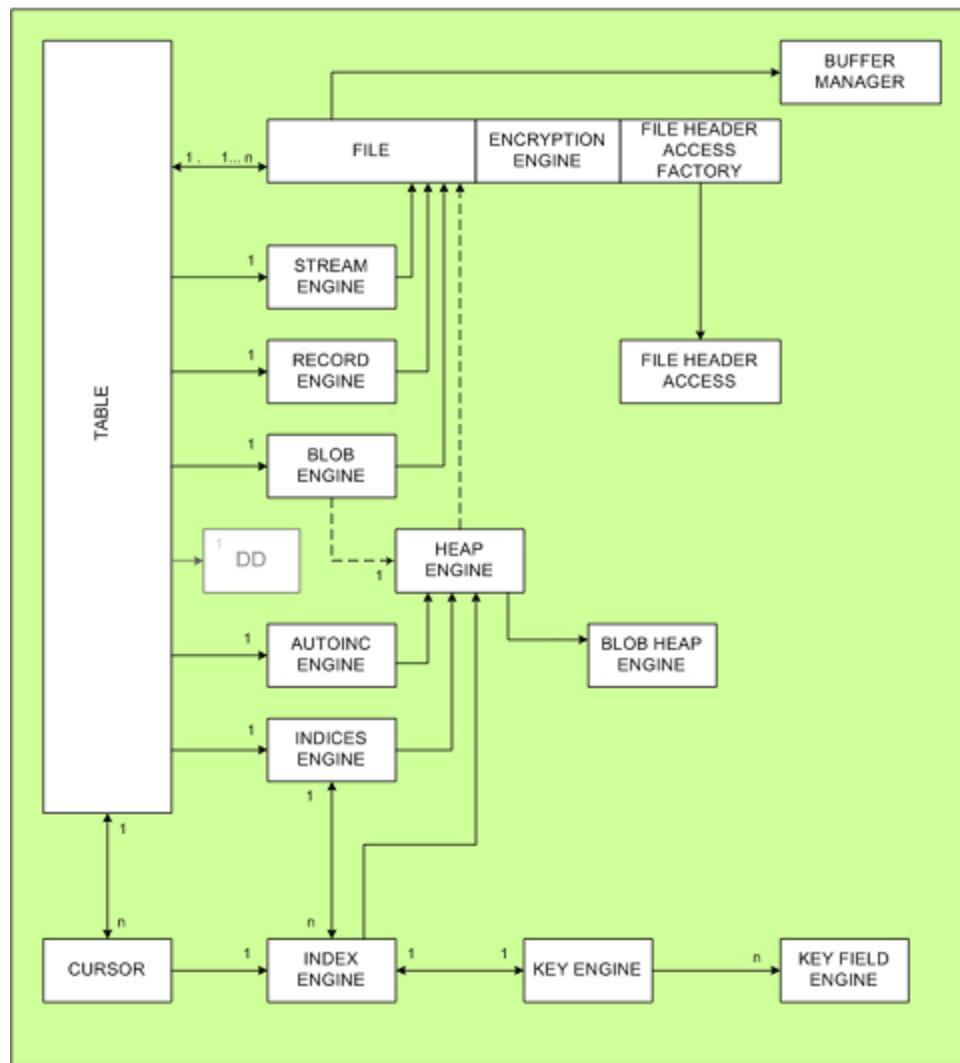
Sub-engines are modules of implementation that are used by the core engine to provide specific database services. The core engine works on the interface defined by the abstract base class of each sub-engine type. NexusDB ships with a standard set of sub engines to provide a complete database implementation.

Sub-engines are a unique architectural feature that differentiates NexusDB from other database implementations. Sub-engines can be replaced or modified to best meet application demands. This architecture provides unmatched flexibility and extensibility and encapsulates specific responsibilities to specialized modules.

The core engine, without change, will be able to support virtually any set of features in the future. For example, sub-engines can change the way NexusDB stores its data, compares its key values or structures its Indices. If your application requires a specific indexing algorithm, create a sub-engine.

Here are NexusDB's sub-engines and their associated responsibilities:

- Key - Responsible for generating and comparing Key Values
- Auto Inc - Responsible for assigning Auto Increment values
- Index - Responsible for Inserting, Deleting and Finding Keys. The actual storage of the Key values in the Index File and navigational routines are also handled within this sub-engine.
- Indices - This sub-engine is a container of Index sub-engines. It dispatches Key changes to the appropriate Index sub-engines.
- Blob - Responsible for storing and retrieving Blobs.
- Record - Responsible for storing and retrieving Records.
- Encryption - Responsible for encrypting and decrypting blocks of data.
- Heap - Responsible for allocating and deallocating blocks within a file.
- Block Heap - Responsible for allocating and deallocating segments of data (i.e. bytes) within a block.



Each sub engine has a well defined interface and is registered to particular database objects within the data dictionary. The diagram above details the relationships between each sub engine and how each works together to provide key services to the core database engine.

Descriptors specify sub-engines by name, not by direct object references. This decouples the actual sub-engine implementation from the Data Dictionary specification. This means that anyone can easily add sub-engine implementations and specify them within existing descriptors.



3.5 Data Security

Data Security



NexusDB ensures that your data files are secure by providing an extensible encryption framework. Utilizing the sub-engine architecture, any number of encryption sub-engines can be registered to encrypt your data files. Only one encryption sub-engine can be used per table. Each table can specify which encryption sub-engine it wishes to use within its data dictionary.

Different algorithms can be implemented within each sub-engine to find the best balance of encryption strength and speed for your application. NexusDB includes a default encryption sub-engine which utilizes the Blowfish algorithm, but you are free to implement different encryption algorithms.

The Remoting Layer also supports encryption of the data that is sent across the network. Included is a transport wrapper that uses the Blowfish algorithm that encrypts or decrypts the stream for another transport. This wrapper tunnels through the attached transport. You may also implement your own encryption algorithm if you have specific security requirements.

In addition to the included encryption options, NexusDB is working closely with StreamSec, a specialized security vendor, to implement a wide variety of encryption capabilities. StreamSec has enhanced their StrSecII library for use with NexusDB. More information is available at StreamSec's website: <http://www.streamsec.com>.

NexusDB has an extensible security framework that is implemented by a security monitor. The default security monitor uses a user access policy that is controlled by three security token: Admin, Read and Write. Each user can have any combination of these tokens within the server. Each token has specific rights:

- The Admin token enables users to restructure databases, modify aliases and perform general administrative tasks.
- The Write token grants permission for the user to add data to tables.
- The Read token permits viewing of the data.

The combination of the Write and Read tokens enable modification to existing data within the tables. You may use the included security monitor or write your own to support intricate security policies. These can be based on the structure of the database as well as data stored within specific records.



3.5.1 Security Issues

Security Issues



Security is a way of blocking functionality. Unlike standard software products, which provide predetermined functionality, performance and reliability etc, security products limit [access to] functionality and must do so in a way that cannot be circumvented. It is difficult to ensure a system is secure, and it may be assumed to be secure until proven otherwise. This can be a problem, as the person who finds the system to be insecure is often the reason security is first sought.

Additionally, security is really not something you will get just by buying a retail product. By virtue of the concept of security you can't effectively evaluate a security product unless you know exactly what

kind of security you are looking for, but unfortunately many people expect this information to be part of the product they purchase. That is possible only up to a point. To design a successful security product, several assumptions need to be made regarding users, environments and risks etc. Differing assumptions about these variables and their relationships may result in a different solution.



3.6 Fulltext Indexing

Fulltext Indexing



NexusDB delivers a modular, flexible and in any imaginable and unimaginable direction extendable generic framework for indexing "tokens", where a token is defined as a sequence of Unicode characters.

In a first step, so called "token extractor's" are responsible for turning a field into a stream of tokens. So far there exists one token extractor class that will return the contents of string and memo fields as one single token. You will be able to write your own token extractors and register it with the server engine. One of our customers has for example written another token extractor which uses XPath expressions to extract token lists out of XML documents.

In the next step the tokens are send through a chain of token filters, e.g. separator at specific characters / Unicode character categories, upper/lower case, stop words, aliases, ... Each of these filters in the chain can transform an input token into any number of output tokens.

In the last step this token stream is feed into token indices. There are currently 2 implementations of a token index. One that feeds the tokens into a normal index, resulting in a number of keys per record in that index (one key per token contained) which allows to directly use FindKey/SetRange on that index to find records containing specific tokens. This engine is included in the Developer Edition.

The 2nd token index engine uses bit-arrays per token to store the information which record contains specific tokens. This makes it possible to very quickly and efficiently evaluate complex expression searches which return a list of records that match the expression search. If you want to display that information in a grid a result set will have to be build though. This engine will be included in the Enterprise version.

The design is modular enough to easily implement any other token index you can imagine.

Every element in this generic indexing framework is extendable/replaceable by deriving your own classes and registering them. As the indexing takes place directly in the engine core it will always be updated in real-time, doesn't increase network traffic and will correctly participate in transaction / nested transaction processing.



3.7 In-Memory Tables

In-Memory Tables



Nexus memtables are a bit different from other memtable implementations. That often leads to a bit of misunderstanding in the beginning as it is unlike e.g. kbmMemTable.

It is easier to think of NX memtables as usual tables that are lost whenever you stop the server process. Like usual tables Nexus memtables are created by the server when you tell it to do so.

This means:

- The table belongs to a database and can therefore be part of transactions with other (mem)tables in that same database.
- The table data is kept at the server and transferred to/from the clients via transports. The transports only come into play with external servers, not with an embedded serverengine. But the principle is the same even in this situation. The server(engine) handles the physical data, be it a file or just some memory blocks.
- Memory tables can have different scopes.
- When using an exposed server the same table can be used by different clients.

To read more about Memory tables please refer to

???update



3.8 Monitors and Extenders

Monitors and Extenders



NexusDB has an internal event infrastructure that you can hook into by means of Monitors and Extenders. Monitors/Extenders allow changes to the default behaviour of database events. These internal events are fired on particular changes or conditions that occur within NexusDB's core engine. These events work very much like triggers. Since Monitors/Extenders are integrated into the core database engine, they are actually much more powerful than traditional SQL based triggers.

Monitors and extenders always work together. Monitors get notified on creation of internal server objects such as sessions, databases, cursors, etc.. When the server creates these objects, a monitor can attach extenders to these objects. Extenders handle specific events that occur within these objects (e.g. the posting of a record in a Cursor object) and can stop, change or extend the default behaviour (e.g. disallowing post for certain users).

Monitors and Extenders can be used for many purposes: logging, custom change notification systems, referential integrity and more. NexusDB includes a functional implementation of referential integrity based on using monitors and extenders. The event mechanism within NexusDB is extensive and complete; Monitors and Extenders empower you to exploit this mechanism.



3.9 Plug-ins

Plug-ins



Plug-ins use the NexusDB transport system to build completely integrated, custom distributed services. Plug-ins provide a middleware solution to enable remote interface invocation within NexusDB's infrastructure. The mechanics are very similar to an RPC (Remote Procedure Call) system.

There are three components that need to be created to leverage the Plug-in architecture: a Plug-in Engine, a Remote Plug-in Engine and a Plug-in Command Handler. The Plug-in Engine is the actual implementation of the extension or service. The Remote Plug-in Engine is the interface a remote client invokes the methods on directly; it serves as a proxy for the actual implementation (i.e. the Plug-in Engine). The Plug-in Command Handler listens in the server process and dispatches the commands received from the Remote Plug-in to the actual Plug-in Engine implementation.

The Plug-in Engine and the Remote Engine are analogous to NexusDB's Server Engine and Remote Server Engine. The Plug-in Command Handler is analogous to NexusDB's Server Command Handler. These components use the existing transport layer to handle remote invocation of methods on the server.

Plug-ins open up a world of possibilities and can be used for things like Server Administration or Diagnostic information. Plug-ins leverage the existing database connection and transport mechanism to invoke custom methods within the server from the client. The Plug-in model parallels the core server model utilization so its use is very natural. All this adds up to a flexible middleware solution with virtually no additional overhead.

The plug-in framework is a very powerful, low-level message based communication system. You can do almost anything with it, but it requires you to write a bit of code and to take care of some housekeeping yourself.



3.10 Real-time Backups

Real-time Backups



Live Backups use what's called a Snapshot Transaction. A snapshot transaction represents the complete database in a consistent and stable state. As long as all data modifications in your program always leave the database in a consistent state while no transaction is open, the backup will always be in a consistent state as well.

There is one possible problem.

During the lifetime of a snapshot transaction the server keeps copies of the original version of all blocks modified by other transactions. This isn't as bad as it sounds as only one copy is required the first time a specific block is changed... if the same block is changed multiple times during a snapshot transaction no additional copies are required. If new blocks are added to a table during a snapshot transaction no copies are required as there is no "original" block. There is also special code in place that makes sure that multiple active snapshot transactions share the same copy of the original block if possible.

Even with all these optimizations in place the amount of data required can become quite big under the following conditions:

- Your backup takes a very long time (e.g. backing up a multi Gigabyte database).
- There are massive changes to the database during that time.

These blocks will be stored in the "block cache" until MaxRAM has been reached and moved out to "temporary storage" as needed to keep total memory requirements below MaxRAM. If temporary storage has been completely filled the engine has no other choice than to ignore the MaxRAM setting and keep the blocks in memory. If this continues past the point where physical memory is available the OS will start swapping out these blocks to disk. This is the point where it gets ugly and the server will come to a crawl.

To prevent this problem you just have to make sure that temporary storage will be big enough to store all changes that can happen during the runtime of the backup. We also recommend scheduling backups at times where you expect the minimum modification to the data.



3.11 SMP Scalability

SMP Scalability



NexusDB utilizes Symmetric Multi-Processor systems completely. The overall affect is higher server performance - work gets done at the server in a shorter period of time. Throughput is maximized due to three reasons:

- Fully threaded core.
- New memory manager improves allocation/deallocation and reduces memory fragmentation.
- Designed to minimize single-threaded serialization points.

A fully threaded core is important because NexusDB will harnesses the full power of your SMP machines unlike databases that require a CPU affinity to be set. The fully threaded core enables the OS to schedule these threads across all of the CPU's, thus maximizing performance. This is a significant advantage and a requirement for enterprise class scalability.

The replacement memory manager reduces heap fragmentation and enables threaded allocation and deallocation. Most default memory managers (including the Delphi one) fragment the heap over time. They also synchronize all memory allocation and deallocation, thus creating a huge bottleneck for multi-threaded applications. For server applications it is essential to minimize fragmentation and maximize threaded memory allocation. The NexusDB replacement memory manager does exactly this and it can be used in all your applications as well.

The design of NexusDB has been crafted from the beginning to minimize serialization points to maximize threaded throughput. It is important that these are initial design goals since refactoring a code base to account for threading is not optimal at best and problematic at worst. NexusDB has been designed to be an enterprise class database management system with all the performance characteristics of those systems.



3.12 SQL Joins

SQL Joins



This is a quick introduction to joins - mostly for people who are new to the subject.

What is a join?

Fundamentally, a join is a cross product of the rows (records) from all participating tables. That is, all possible combinations of all rows from the tables.

Example:

Say you have two tables, T1 (2 rows) and T2 (3 rows):

T1	T2
'A'	1
'B'	2
	3

then a join between T1 and T2 will produce the following result table, T3:

T3	

'A'	1
'A'	2
'A'	3
'B'	1
'B'	2
'B'	3

Though we use only two input tables here, there is no formal limit to the number tables you can join simultaneously.

Declaring joins

In SQL, joins can be specified in two different ways:

- implicitly, or
- explicitly with the JOIN keyword.

Even if you've never used JOIN in a query, there's a good chance you have been using joins implicitly. The join whose result we saw previously can be expressed like this:

```
SELECT * FROM T1, T2 --  
implicit join
```

or like this:

```
SELECT * FROM T1 CROSS JOIN  
T2 -- explicit join
```

The two forms are exactly equivalent.

In general, for cross joins, you end up with a result table with $\text{RowCount}(T1) * \text{RowCount}(T2) \dots * \text{RowCount}(Tn)$ rows and $\text{ColumnCount}(T1) + \text{ColumnCount}(T2) \dots + \text{ColumnCount}(Tn)$ columns. In our case, that's $2 * 3 = 6$ rows with $1 + 1 = 2$ columns.

It's clear from this that results of cross joins can become huge very quickly. Fortunately, most joins work on only a small subset of the complete cross product.

The formal definition for an explicit cross joins is:

```
<Table> CROSS JOIN <Table>
```

Combining related information

Usually, joins are not used to build complete cross products but rather to combine related information from various tables.

Lets assume we have our two tables, T1 and T2, again, but this time they also have a common column, Code, whose values correspond between the two tables:

T1		T2	
Code	Value	Code	Value
'Z01'	'A'	'Z01'	1
'Z02'	'B'	'Z02'	2
		'Z02'	3

We can now correlate the data using an implicit join like this:

```
SELECT * FROM T1, T2 WHERE
T1.Code = T2.Code;
```

The result:

Code	T2.Code	Value	Value_2
'Z01'	'Z01'	1	'A'
'Z02'	'Z02'	2	'B'
'Z02'	'Z02'	3	'B'

The same logical join operation can be expressed explicitly like this:

```
SELECT * FROM T1 JOIN T2 ON
T1.Code = T2.Code;
```

Except for the column headings and order, the result is equivalent:

Code	Value	T2.Code	T2.Value
'Z01'	'A'	'Z01'	1
'Z02'	'B'	'Z02'	2
'Z02'	'B'	'Z02'	3

In general, the ON clause on the JOIN looks and works just like a WHERE clause on a SELECT, except that an ON clause must specify one or more relations between the two tables being joined. Also, many SQL engines support only the equality operator for joining, but some support others, like \geq or \leq .

The formal definition for a join with an ON clause is:

```
<Table> JOIN <Table> ON
<conditional expression>,
```

where <conditional expression> is usually a form similar to

```
<Table1.ColumnX =
Table2.ColumnY [ AND
Table1.ColumnZ =
Table2.ColumnW]....>
```

For joins where the names of the columns being joined on are the same between the participating tables, there's a more compact way to express the same join:

```
SELECT * FROM T1 JOIN T2
USING(Code);
```

The result of a JOIN with a USING clause differs slightly from one with an ON clause by listing the columns used for joining only once each:

Code	Value	T2.Value
'Z01'	'A'	1
'Z02'	'B'	2
'Z02'	'B'	3

The formal definition for a join with a USING clause is:

```
<Table> JOIN <Table>
USING(<column list>);
```

There's also an ultra-short form, the NATURAL JOIN:

```
SELECT * FROM T1 NATURAL
JOIN T2;
```

NATURAL JOIN is the same as JOIN USING with the list of columns in the USING clause containing ALL columns that have their names in common between the participating tables. In our case, the join above corresponds to

```
SELECT * FROM T1 JOIN T2
USING(Code, Value);
```

However, since - in our case - no values in the Value columns are the same between the two tables, the result is an empty set.

The three previous join types discussed (NATURAL, JOIN ON, JOIN USING) are referred to collectively as inner joins. You are allowed to express that literally specifying the INNER keyword before the JOIN, e.g.:

```
NATURAL INNER JOIN
INNER JOIN ON...
INNER JOIN USING...
```

but in all three cases, the INNER keyword is optional.

So far, we have not seen any explicit joins (using the JOIN keyword) that could not have been expressed equally well using an implicit join (SELECT with a list of source tables). Indeed, it is only a matter of user preference which form is used - the two forms have the same performance characteristics on most SQL implementation.

Outer joins

The difference between inner and outer joins has to do with NULL values. If no NULL values are involved in a join*, then an outer join is equivalent to its inner form. By 'involved' in the previous sentence is meant compared among tables to determine whether or not a row should appear in the result.

Note that the NULLs need not be explicit: An empty table expression has a value of NULL, as we shall see shortly.

Generally in SQL, comparing a value of NULL to anything else (including another NULL) gives an undefined result. So for an expression like

```
WHERE T1.Code = T2.Code
```

if either T1.Code or T2.Code (or both) is NULL, then the result of the expression will become False*. For a join, this means that there will be no rows in the result table for source rows where any of the columns participating in the join expression are null. This is what you'd intuitively expect, so that's fine for most situations. Sometimes, however, it is nice to get a row in the result that represents missing information in the source tables.

Or rather Not True (i.e. Unknown). Also note that - although not relevant right here - NULL <> NULL is Not True as well.

As an example, if you have a customer table and an orders table, you can join the two giving a result with all customer and order information combined:

Customers	
CustNo	Name
1	IBM
2	Oracle
3	Informix

Orders		
OrderNo	CustNo	OrderAmount

1	1	100,000
2	1	250,000
3	3	10,000
4	4	5,000

```
SELECT * FROM Customers JOIN
Orders USING(CustNo);
```

CustNo	Name	OrderNo	OrderAmount
1	IBM	1	100,000
1	IBM	2	250,000
3	Informix	3	10,000

But what if we would like all customers to appear - even the ones with no orders? That's where outer joins come in. An outer join is like an inner join except we get to specify that rows from either the left, the right, or both tables in the join with no matching record in the 'opposite' table should appear in the result.

Example:

```
SELECT * FROM Customers LEFT
OUTER JOIN Orders
USING(CustNo);
```

CustNo	Name	OrderNo	OrderAmount
1	IBM	1	100,000
1	IBM	2	250,000
2	Oracle	<null>	<null>
3	Informix	3	10,000

Here, by specifying LEFT, we've said that rows from the left-hand table in the join (Customers) with no corresponding row in the right-hand table (Orders) - when considering the join operator (CustNo) - should appear in the result. Since there is no data from Orders to combine the Customer data from such rows with, the columns for Orders all become <null>.

Similarly, if we want to list all customer/order combinations, plus all orders with no matching customer, we can specify a RIGHT join:

```
SELECT * FROM Customers
RIGHT OUTER JOIN Orders
USING(CustNo);
```

CustNo	Name	OrderNo	OrderAmount
1	IBM	1	100,000
1	IBM	2	250,000
3	Informix	3	10,000
<null>	<null>	4	5,000

Finally, we can ask for a combined LEFT and RIGHT join known as a FULL join:

```
SELECT * FROM Customers FULL
OUTER JOIN Orders
USING(CustNo);
```

CustNo	Name	OrderNo	OrderAmount
1	IBM	1	100,000
1	IBM	2	250,000
2	Oracle	<null>	<null>
3	Informix	3	10,000
<null>	<null>	4	5,000

Note that In all three cases, since LEFT, RIGHT, and FULL all imply outer joins, the OUTER keyword is actually obsolete and can be omitted.

Note also that I've specified a USING clause in all examples because I happen to use a matching column name between the tables, but an ON clause or a NATURAL clause would work the same in principle.

The formal definitions for an outer joins is

```
<Table1> [NATURAL] [LEFT |
RIGHT | FULL] [OUTER] JOIN
<Table2> ....
```

When and why use explicit joins?

Joins can often be executed more efficient than their non-join equivalent. Consider the following example:

Say we want to list the names of all customers with orders. A typical non-join solution might look like this:

```
SELECT Name FROM Customers
WHERE
    EXISTS (SELECT * FROM
Orders WHERE Orders.CustNo =
Customers.CustNo);
```

This query uses what is known as a correlated subquery. What that means is that the subquery (the part in the parenthesis) refers to something from the enclosing expression (Customers.CustNo). Query engines will typically process this query by iterating over the rows in the Customers table, executing the subquery for each one. Since there is a certain fixed overhead involved in executing any query (including a sub-query), the following solution - without the subquery - will almost always be considerably faster:

```
SELECT DISTINCT Name FROM
Customers JOIN Orders
USING(CustNo);
```



3.13 Transactions and Data Integrity

Transactions and Data Integrity



This section describes the way in which database transactions are implemented in NexusDB. Any such discussion must embrace a large terminology. To ensure consistency in terminology, we begin with a short section, independent of NexusDB, of database transactions and deadlocks. The importance of locks (record, table and database) is considered in the context of not only ensuring the most efficient use of transactions, but also to help minimize the occurrence of deadlock situations. It's important to mention the different types of locks available and how they are granted by the database engine:



3.13.1 Locks and Locking

Locks and Locking



NexusDB use different kinds of locking to perform optimally in multi user environments.

Content Locks

Content locks can be either record-level (these are always write locks, acquired by calling `Edit`) or table-level (these can be either read or write locks, acquired by calling `LockTable`).

Content locks can be contending for the same resource. If so, then an existing record-level content lock can prevent table-level content locks from being granted. Similarly a table-level content lock will prevent a record-level content lock from being granted.

In NexusDB only one cursor can own a write lock on the same table. Many cursors can own a read lock. As long as any cursor holds a read lock no write lock can be placed on the table. Table level locks interact with record level locks. Record level locks are always write locks. You cannot acquire a table level read lock if any cursor holds a record lock. You cannot acquire a table level write lock if any OTHER cursor holds a record lock. You can only acquire a record lock if there are no table level locks or if you own a table level write lock.

A table level read lock prevents anyone including yourself from changing the table while you hold the lock. A table level write lock prevents anyone else from changing the table while you hold the lock.

Transaction Locks

Transaction locks are independent from content locks. A shared transaction lock is acquired when you read from a table in the context of a transaction. An exclusive transaction lock is acquired when you write to a table in the context of a transaction.

How are locks implemented?

Each server side table object has 2 different synchronization objects that manage access to the table:

TnxLockContainer

TnxLockContainer handles transaction locking (shared/exclusive locks). This lock container is only used when the table is accessed in the context of a transaction. Read access needs a shared lock. Write access requires an exclusive lock. At any point in time there can be no locks, a single or multiple shared locks or a single exclusive lock. No other combination is possible.

TnxReadWritePortal

TnxReadWritePortal handles thread synchronization. Each thread that wants to read from the table outside of a transaction must acquire a read lock on this portal. (This read lock is always released before a server call returns to the client. It is only acquired for the split second it takes to perform the actual read of a single record). A thread that wants to commit a transaction which has acquired an exclusive lock on the table must acquire a write lock on this portal before it can commit its changes. This is needed to prevent reading threads from accessing the buffer manager while memory pages are copied from the transaction buffers into the real buffers.

In summary there can be different types of access to a table:

Read access

In the context of a transaction this acquires a shared lock until the transaction is completed. Outside a transaction read access acquires a read lock on the table for the split second it takes to actually perform the read.

Write access

This can only be done in the context of a transaction (a transaction is automatically started and immediately committed if no active transaction is present). Write access will acquire an exclusive lock

until the transaction is completed and committed. It also requires an active transaction (which has already acquired an exclusive lock, preventing read access from inside other transactions) and acquires a write lock (preventing reads from outside a transaction) for the time it takes to commit the changes to disk.

Transaction isolation levels

First a quick description of the different possible isolation levels:

READ UNCOMMITTED or DIRTY READ

You can read an uncommitted transaction that might get rolled back later. This isolation level is also called a dirty read. This is the lowest isolation level.

READ COMMITTED

This ensures that data that another application has changed and not yet committed can not be read, but it does not ensure that the data will not be changed before the end of the current transaction.

NON REPEATABLE READ

This occurs when a transaction reads the same record more than once and, between the two (or more) reads, a separate transaction modifies that record. Because the record was modified between reads within the same transaction, each read produces different values, which introduces inconsistency.

REPEATABLE READ

When it's used, then dirty reads and no repeatable reads cannot occur.

PHANTOM

Phantom behavior occurs when a transaction attempts to read a record that does not exist and a second transaction inserts the record before the first transaction finishes. If the record is inserted, the record appears as a phantom to the first transaction, inconsistently appearing and disappearing.

SERIALIZABLE

This is the most restrictive isolation level. When used, phantom values cannot occur. It prevents other users from updating or inserting records into the data set until the transaction is complete.

There is only one physical type of transaction in NexusDB. The distinction between implicit and explicit transactions is a virtual one. Any operation that results in write access to a table, checks if a transaction is already present. If not present, then a transaction is started (implicit), the operation performed (insert, modify, delete) and the transaction committed/rolled back depending on the success of the operation.

An explicit transaction is a client side controlled transaction.

Using the above terminology, we can now classify NexusDB as follows:

Dataset state or context	NexusDB
outside of an explicit transaction	read committed

in the context of an explicit transaction	serializable
locking granularity	table level



3.13.2 Deadlocks

Deadlocks



Deadlock Types

There are 2 different kinds of deadlocks than can occur:

1) User A acquires a record-level content lock. (Releasing that lock means that user A must either cancel its modifications or post them, posting results in an implicit transaction that requires an exclusive transaction lock.) User B starts an explicit transaction (which acquires an exclusive transaction lock and wants to change the record that user A has locked (which requires a record-level content lock)).

To resolve this deadlock user A must release its content-lock. It must ether cancel its modification or user B must rollback the transaction, so that user A can post its modification.

This type of deadlock could and has occurred with NexusDB 1.

2) User A starts a transaction and reads from a table (acquires a shared lock). User B starts a transaction and reads from the same table (acquires a shared lock). User A wants to write to the table (tries to acquire an exclusive lock on the table and must wait for user B to release its shared lock). If user B now completes its transaction without trying to write to the table everything is ok. But if user B wants to modify the table it too must acquire an exclusive lock. Now both transactions have a shared lock, want an exclusive lock and are waiting for each other. We have a deadlock.

To resolve this deadlock one of the transactions must be rolled back and retried later. This type of deadlock is new in NexusDB.

Schemes for Avoiding Deadlocks

If you only have a single table, or only develop single-exe applications that do not use distributed processing or threads, then you can be absolutely certain that you will never need to plan for deadlocks. The moment you have more than one point in your application where you perform database operations in the context of a transaction, the chances are that at least some of them can be executed at the same time (by different users/threads) if the transaction locking is at the table level and can differentiate between shared and exclusive locks. An awareness of concurrency issues and consequent deadlock situations is thus paramount to an understanding of how your applications will perform in the real world when using a NexusDB backend.



3.13.3 Optimistic Record Locks

Optimistic Record Locks



NexusDB supports optimistic recordlocks. For this to work on a table or live resultset, a field of type nxtRecRev must exist in it. This field type is a special from of Word32 field, which is read only. When you insert a new record, the field has the value 0. Every time you update a record, the record revision is increased by 1.

Pessimistic locking works by placing a lock, and then reading the current record if you call Edit. The lock guarantees that nobody else can modify the record until you post your changed version back to the server.

Optimistic locking doesn't place any lock before reading the current record when you call Edit. While your client side table is in "edit mode", anyone else can come and change or delete your record. When you try to post your changes, the server first checks if the record still exists at all; if not, you get a "optimistic locked record deleted" exception. After that it finally has to acquire a pessimistic lock on the record for the split second it takes to actually update it. If another user has a long term pessimistic lock on the record, you get a timeout exception on post and can retry or cancel. After the short term lock has been acquired, the record revision stored in the record on disk is compared to the record revision of the changed record you are trying to post. If it's different a "optimistic locked record modified" exception is raised, otherwise the changed record is successfully updated. As the correct handling of these exceptions requires special client side code, optimistic locking is a per cursor client side setting.

Optimistic vs Pessimistic Locking

Pessimistic locking means that no one else can edit the record while the lock is active, but you are guaranteed that no one else has changed the record when you go to post.

Optimistic locking means others can edit it, but when you go to post, the post will fail if someone else has changed the record while you were editing it. NexusDB uses a special numeric field of type RecRev to implement optimistic locking. Every update to a record increases this field and an update command on the server side fails if the old and newvalue of this field don't match. Optimistic locking applies to recordlocks, not to transaction locks. Transaction locks are always pessimistic.



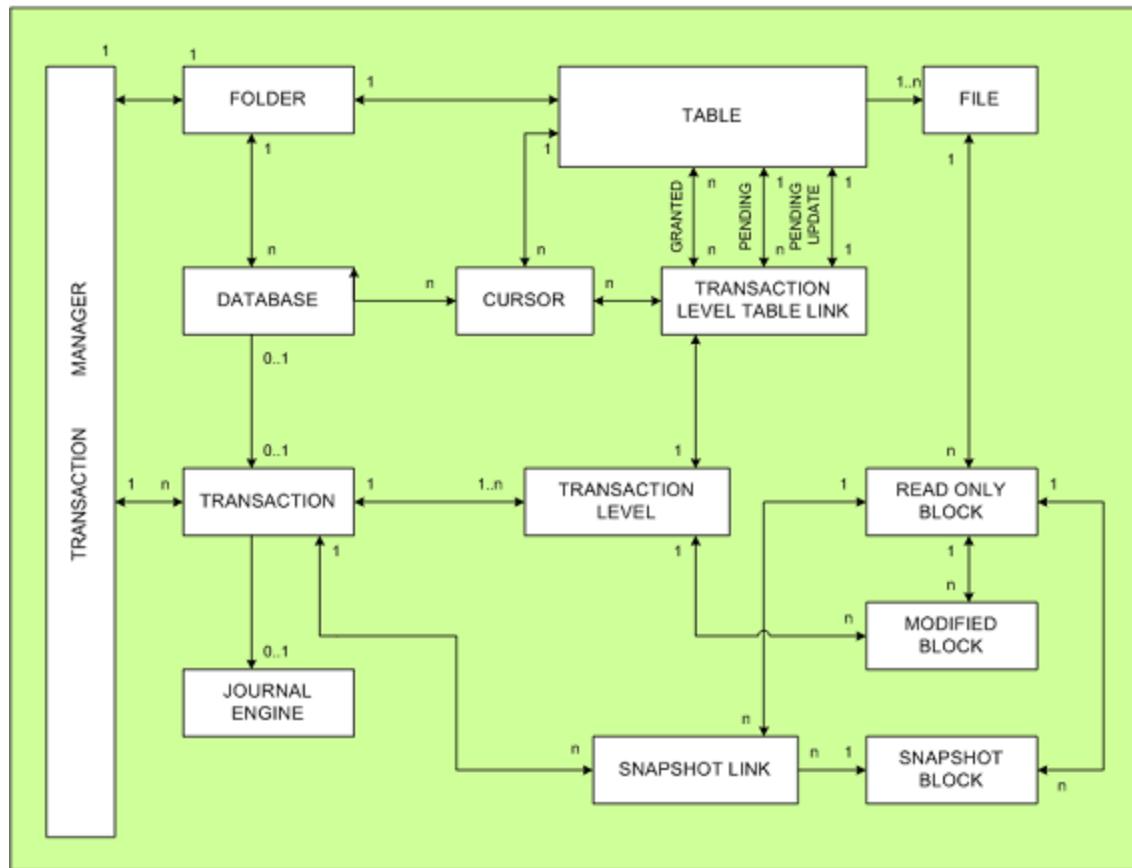
3.13.4 Transaction Types

Transaction Types



From the beginning, NexusDB has been designed to completely utilize and support transaction-based processing. In addition to standard transactions, NexusDB supports different types of transactions for a wide range of applications. NexusDB supports these additional transaction types: failsafe, nested and snapshot transactions.

The transaction manager in NexusDB takes care of all transaction relevant management.



Isolation Levels

Open cursors outside the context of an explicit transaction have READ COMMITTED isolation. They can read from tables which are currently locked by an active transaction, no matter if that table is locked in shared or exclusive mode and will not place a lock of their own.

If a cursor belonging to a database with an active transaction is used to read or write it will join the context of that transaction and place a shared (read) or exclusive (write) lock or updates an existing shared lock to exclusive lock if required. Multiple transactions can own a shared lock or a single transaction can own a exclusive lock. This locking system guarantees SERIALIZABLE transaction isolation for cursors in the context of a transaction.



3.13.4.1 Nested Transactions

Nested Transactions



Nested transactions are when transactions contain other transactions in a "nested" fashion. NexusDB supports nested transactions so that correct transactional semantics can be applied throughout your code without compromise. Proper support of nested transactions means that you may commit or rollback an inner or nested transaction without affecting the outer transaction's ability to commit or rollback.



3.13.4.2 Snapshot Transactions

Snapshot Transactions



In addition, NexusDB supports SNAPSHOT isolation level. Snapshot transactions are always read only. They never place any locks. Cursors in the context of a snapshot transaction see the database in exactly the (READ COMMITTED) state it was in at the moment the transaction was started. This is achieved by using a block level multi-versioning system. SQL SELECT statements will automatically be executed in the context of a snapshot transaction if no transaction is already active for that database. This guarantees that even long running queries will see a consistent view of the database without placing any locks. With this setup we have been able to achieve an almost perfect degree of concurrency, since readers never block writers. Whereas writers block readers only for an extremely short time to execute UPDATE/COMMIT.



3.13.4.3 Failsafe Transactions

Failsafe Transactions



An explicit transaction has its own "server cache" in memory on the server. This cache is only visible to that transaction. All updated records are written to the server cache. Pages of the cache which have been changed are flagged "dirty" and when the transaction is committed will be written to the tables on disk. If the power fails when some, but not all, of the dirty pages have been written to disk the database will be left in an inconsistent state.

To prevent this possibility set the Failsafe property of your TnxDatabase component to true. As the transaction proceeds clean "before images" of the cache pages which are about to be updated are

written to a Transaction Journal File (TJF). When a transaction is committed all the dirty pages are written as "after images" to the TJF and the completed file is closed and flushed to disk. The dirty pages are then written to the database tables on disk and when complete the TJF is deleted. If the power is lost the TJF can be used later to automatically complete the transaction.

For a Rollback the dirty cache pages are discarded and the TJF is deleted.

There are 3 settings available with Failsafe transactions:

- Mode 1 ("Dialog") – before and after images are stored. If a reboot finds these images then a dialog is displayed and the operator asked whether to commit or rollback the changes found.
- Mode 2 ("Always Rollback") – stores before images only. If a reboot finds these images an automatic rollback is performed. This can be thought of as a conservative, or cautious, mode of operation.
- Mode 3 ("Always Commit") – stores after images only. If a reboot finds these types of images an automatic commit is performed.

Mode 2 and 3 require no user interaction. This makes them perfect if the NexusDB Server is running as a service.

The use of the journal file in this manner implements a two-phase commit transaction protocol. The two-phase commit protocol is an industry standard way of ensuring the highest level of data integrity. Many transaction theory references discuss this protocol in depth, for more information on the two-phase commit protocol it is recommended you consult those external references.

How is a commit performed?

Committing a failsafe transaction roughly means:

- write journal header marked as incomplete
- write all before and/or after pages to the journal (depending on the journal engine used)
- flag the journal header as completed
- write the changes to the real table files
- delete the journal

Using the rollback journal engine gives a consistent experience for the client: if the commit call returned with DBIERR_NONE all changes have been committed to disk, otherwise not.

Using e.g. the commit journal engine that the transaction will be committed **if** a journal file that already reached stage c) is found on start of the server. If the journal file is still marked as incomplete it is simply deleted. For the next release of NexusDB I've removed the dialog that was shown if an incomplete journal file was found as it is meaningless. As long as the journal is still marked as incomplete no action needs to be taken beyond deleting the journal to recover the database to a consistent state. The dialog maybe made some people believe there was an error that resulted in data corruption when in reality everything worked correctly.

There are 3 points to activate failsafe transactions: per Server, per Database or per explicit transaction. Turning it on at one level forces it on for all higher levels. There is currently no way to force it off from a higher level if it is on at a lower level.



3.13.4.4 Implications for Developers

Implications for Developers



Keep your pure readers completely out of any transactions (running canned queries will automatically use snapshot transactions). That way they will never block any writers and will always see a consistent view of the database (READ COMMITTED). Run your UPDATE statements either without any active transaction (the single UPDATE statement will implicitly run in a normal SERIALIZABLE transaction this way) or if you have multiple UPDATE statements which belong together run them in a normal transaction together. Writers will block each other (as long as the transactions are short enough they'll just queue up and execute one after another) Writers will block readers for the split second it takes to update a few pointers inside the buffer manager during the actual COMMIT of a transaction (making sure that readers will only see completely committed transactions).

Snapshot transactions are read-only transactions that preserve the database's state at a given point in time; they take a "snapshot" of the database. The overall effect of this is that readers using snapshot transactions do not block writers. Throughput is increased and a consistent view of the data is provided.

Snapshot transactions are efficiently implemented within the server core. In order to use snapshot transactions, the developer simply specifies the transaction type when starting a transaction. Snapshot transactions are the default transaction type used for "select" SQL queries. They are also used to provide online backup support.



3.13.5 Advice for Developers

Advice for Developers



NexusDB implements pessimistic and optimistic record locking via navigational (cursor based) access with the same general semantics as BDE, MS-SQL, Oracle, Sybase etc., too. Usually controlled changes should always be done with pessimistic record locking as this correctly serializes record access in high contention scenarios and prevents unnecessary overhead from having to repeat the operation in case of a conflict as happens in optimistic locking.

Depending on the requirements of the system it might make sense to implement long, user controlled changes using optimistic locking. This prevents the "user goes for a coffee" scenario from blocking automated processes from updating the records. When the user finally posts and an exception results some form of conflict resolution needs to be implemented by the developer.



3.13.6 Referential Integrity

Referential Integrity

???todo



3.14 Transports

Transports



NexusDB has a number of Transports included. Here we will outline them and show ways for tweaking them in certain environments. In NexusDB each transport has its advantages and disadvantages. Here's a table detailing the various aspects of each Transport:

Transport	Advantages	Disadvantages
(D)COM Transport	<ul style="list-style-type: none"> • fast for in-process usage • allows a certain degree of security via Windows security 	<ul style="list-style-type: none"> • slow over networks and out-of-process usage • might be tricky to set up if security restrictions apply
Named Pipe Transport	<ul style="list-style-type: none"> • fast if client and server on the same machine • allows a certain degree of security via Windows security • Allows strong encryption 	<ul style="list-style-type: none"> • slow over networks • not easy and straightforward to set up due to windows security
TCP Transport	<ul style="list-style-type: none"> • fast over networks • very well scalable • easy to setup • Allows strong encryption 	<ul style="list-style-type: none"> • Slow if client and server on the same machine
Shared Memory Transport	???todo	???todo

Please note that the usage of fast and slow is relative between these transports. NexusDB implements some of the most advanced and fastest data transports available.

To make your application perform optimal consider the following points:

- what will be the typical environment your application will run? (Client/Server on same machine? Over the network? In-Process & Out-of-Process?)
- do you need security at all? Strong or basic security?
- do you need one or more transports?

Keep the mentioned (dis)advantages of the above table in mind and you should be able to make the right decision for your application.

COM Transports

What is the difference between the customcom transport and the registeredcom transport, and when to use which?

Registered com transport uses the ROT (running objects table) which is a system local list of running objects managed by windows to establish the initial communication between client and server.

Custom com transport just calls an event and you have to implement that event. passing the InxTransport that's passed into the event to the server side transport, call AttachRemoteTransport there, passing in the InxTransport and then pass the InxTransport you get back from that function back to the client. how one do this is completely up to him/her. if both client and server transport are in the same process it's very easy. but one can also use e.g. DCOM or any other mechanism to pass the InxTransports between client and server.



3.14.1 Selecting the correct Transport

Selecting the correct Transport



It's very important to use batch modes and compression when working on slow networks. In most Client/Server applications the network speed is the actual bottleneck.

NexusDB offers some very powerful settings to tweak the throughput for different network speeds. For this purpose the transports have support for built-in or custom compression engines. For your convenience NexusDB already comes with ZIP compression (registered as compression types 1-9) and RLE compression (type 10).

Maximum ZIP compression (type 9) compresses typical ASCII record data to at least 1:25 for data blocks bigger than 64 Kbytes. Our tests show that this compression in combination with a BlockReadSize of at least 256KByte gives very good results.



3.14.1.1 Why use compression and batch modes?

Why use compression and batch modes?



This is best explained with a simple example:

An application needs to fetch 63, 000, very well compressible (ratio 1:50), records over a 28.8kBit line. The total data to be transferred is around 81 mega bytes (85,509,224 bytes). This operation takes in NexusDB default mode (without batch and compression) 31,121 seconds which is about 2 records per second. Slow you say? Let's see.

The results are direct results of network lag time and maximum data transferred per second. Look at above example again: 28 kbit lines have typical ping times of around 150-200 ms. Fetching 63,000 records takes (at least) 126,000 messages sent over the network which results in 18,900 seconds minimum time for the transfer.

The other number to look at is the data amount. If we take into account the 81 mega bytes of data transferred over line (that can transfer a theoretical maximum of 3.6 Kbytes) we would need minimum time of 23195 seconds for transferring the data.

Now what can batch mode and compression do here? Let's use a BlockReadSize of 512 Kbytes and maximum ZIP compression to transfer the same data. Using NexusDB the data can be fetched in just 540 seconds. That's 116 records per second, which is over 50 times faster than in default mode. Why?

First we reduce the number of messages to 164 which gives us a minimum time of 24.6 seconds. For the data transfer we can compress the data to 1,706,639 bytes (ratio 1:50) with maximum zip compression in 512kByte blocks. This will then results in 462 seconds theoretical minimum.

Looking at the theoretical numbers the actual benchmarked NexusDB numbers look pretty good, taking network and message overhead into account.



3.14.1.2 Which compression and batch size should I use?

Which compression and batch size should I use?



Reading above example that sounds easy to answer, right? But unfortunately it isn't. The above example is an extreme example, real-world usage will in most cases not be that compressible and the network speed is typically variable. On a 100MBit for example ZIP compression makes the transfer slower by factor 2 because the compression takes longer than the actual transfer.

NexusDB lets you set the compression type and batch mode and size on a per client basis. This gives you full control of how a client uses them and thus you can tweak the client for different network connections.

To figure out the best settings for every application and its data needs to be looked by the developers on their own. The best way to this is by having an application option (e.g. ini file) that allows enabling and tweaking of the 2 or 3 properties involved. Our tests show that for connections below 2Mbit compression and batch modes become the way to go. Using speeds between 2Mbit and 10Mbit leaves both modes at about the same result, while for everything above 10Mbit the compression should be better left alone.



4 Appendices

4.1 System Capacities

System Capacities



Capacity	Details
Fields in Table	as many as you want as long as the data still fits in a single block
Fields in an Index Key	as many as you want as long as 4 keys still fit in a single block
Indices in a Table	255
Open BLOBs in a Table	The maximum number of open blobs is only limited by the available memory constraints of the operating system or hardware.
Open Tables	The maximum number of open tables is only limited by the available memory constraints of the operating system or hardware.
Records in a Table	2^{32}
Records in a Transaction	The maximum number of records in a single transaction is only limited by the available memory constraints of the operating system or hardware.
Tables in a Database	limited only by the os file system
Size Of Database Names	The maximum size of a database name is 60 bytes.
Size Of Field Constraint	unlimited
Size Of Field Descriptions	unlimited
Size Of Field Names	unlimited
Size Of Index Names	unlimited
Size Of Physical Table Files	2^{32} blocks, possible block sizes are 4, 8, 16, 32, 64kb; file size may be limited by file system used
Size Of Table Descriptions	unlimited
Size Of Table Names	limited by the used file system
Size of BLOB Blocks	blobs are stored in segments with 4 byte granularity
Size of BLOB Fields	The maximum length of a BLOB field is 4 gigabytes. Some access methods (e.g. Tdataset in D6 and older) are limited to 2 gb
Size of In-Memory Tables	The maximum length of an in-memory table is only limited by the available memory constraints of the operating system or hardware.
Size of Index Keys	The maximum size of an index key is 1/4 of the block size minus some bytes for per block overhead. The exact index key size depends on the key engine used.
Size of Index Pages	same as "physical table files"
Size of Records	The maximum record size is a bit under 64 kilobytes.
Size of String Fields	The maximum length of a string field is only limited by total record size. Some access methods may further limit the maximum string field size



4.2 Data Types

Data Types



Nexus supports the following field types:

Type	Capacity
nxtAutoInc	32 bit unsigned integer
nxtBCD	Precision 20, scale 4
nxtBLOB	4 GB
nxtBLOB Graphic	4 GB
nxtBLOB Memo	4 GB
nxtBoolean	8 bit boolean flag
nxtByte	8 bit unsigned integer
nxtByte Array	64 KB
nxtChar	8 bit character
nxtCurrency	Precision 20, scale 4
nxtDate	4 bytes
nxtDateTime	8 bytes
nxtDouble	5.0E-324 .. 1.7E308
nxtExtended	3.6E-4951 .. 1.1E4932
nxtGUID	16 bytes
nxtInt8	8 bit signed integer
nxtInt16	16 bit signed integer
nxtInt32	32 bit signed integer
nxtInt64	64 bit signed integer
nxtNullString	8192 characters
nxtRevRev	32 bit unsigned integer
nxtShortString	255
nxtSingle	1.5E-45 .. 3.4E38
nxtTime	4 bytes
nxtWideChar	16 bit character
nxtWideString	32767 characters

nxtWord16	16 bit unsigned integer
nxtWord32	32 bit unsigned integer



4.3 Error Messages

Error Messages

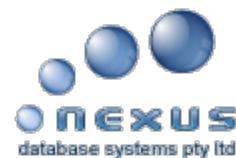


Here's a list of all NexusDB related error messages. The list is sorted by error number and contains a short description of the error.

???update

Constant name	Dec.	Hex.	Description
DBIERR_REENTERED	8455	\$2107	System has been illegally re-entered.
DBIERR_BOF	8705	\$2201	At beginning of table.
DBIERR_EOF	8706	\$2202	At end of table.
DBIERR_RECDELETED	8708	\$2204	Record/Key deleted.
DBIERR_NOCURRREC	8709	\$2205	No current record.
DBIERR_RECNOTFOUND	8710	\$2206	Could not find record.
DBIERR_ENDOFBLOB	8711	\$2207	End of BLOB.
DBIERR_OBJNOTFOUND	8712	\$2208	Could not find object.
DBIERR_FILECORRUPT	8962	\$2302	Corrupt file - other than header.
DBIERR_READERR	9217	\$2401	Read failure.
DBIERR_WRITEERR	9218	\$2402	Write failure.
DBIERR_FILENOACCESS	9221	\$2405	Cannot access file. File may be in use by another process.
DBIERR_NOMEMORY	9473	\$2501	Insufficient memory for this operation.
DBIERR_TABLEFULL	9479	\$2507	Table is full.
DBIERR_INDEXLIMIT	9487	\$250F	Too many Indices on table.
DBIERR_KEYVIOL	9729	\$2601	Key violation.
DBIERR_REQDERR	9732	\$2604	Field value required.
DBIERR_FIELDISBLANK	9740	\$260C	Field is blank.
DBIERR_INVALIDPARAM	9986	\$2702	Invalid parameter.
DBIERR_INVALIDHNDL	9990	\$2706	Invalid handle to the function.

DBIERR_UNKNOWNFILE	9992	\$2708	Cannot open file.
DBIERR_NOSUCHINDEX	9997	\$270D	Index does not exist.
DBIERR_INVALIDBLOBOFF SET	9998	\$270E	Invalid offset into the BLOB.
DBIERR_INVALIDDESCNU M	9999	\$270F	Invalid descriptor number.
DBIERR_INVALIDFLDTYPE	10000	\$2710	Invalid field type.
DBIERR_INVALIDFLDDESC	10001	\$2711	Invalid field descriptor.
DBIERR_INVALIDFLDXFOR M	10002	\$2712	Invalid field transformation.
DBIERR_INVALIDRECSTRU CT	10003	\$2713	Invalid record structure.
DBIERR_INVALIDDESC	10004	\$2714	Invalid descriptor.



5 Data Management Tools Home

Data Management Tools Home



In this Data Management Tools book we investigate the tools for managing NexusDB Databases in the following sections:

- The NexusDB Server engine
- The Enterprise Manager database management utility
- The Importer for converting data from other table formats to NexusDB

Each data management tool is discussed in depth. All aspects of setup and utilization are covered.

For the latest information on Nexus products be sure to regularly visit the website at www.nexusdb.com and, while you are there, add the various newsgroups (at news.nexusdb.com) to your reader.

©2004 Nexus Database Systems Pty Ltd.



6 The NexusDB Server

The NexusDB Server



NexusDB Server is a transaction-based database management system server implemented as a set of Delphi components. NexusDB is a lightweight but industrial strength database engine. It is the ideal database for use in desktop, client/server, web and n-tier applications.

©2004 Nexus Database Systems Pty Ltd.



6.1 Introduction

Introduction



In this part of the manual we introduce the NexusDB Server application. This is the utility to be deployed with multi-user C/S applications. The client applications must connect to a running instance of this utility to access tables remotely.

©2004 Nexus Database Systems Pty Ltd.

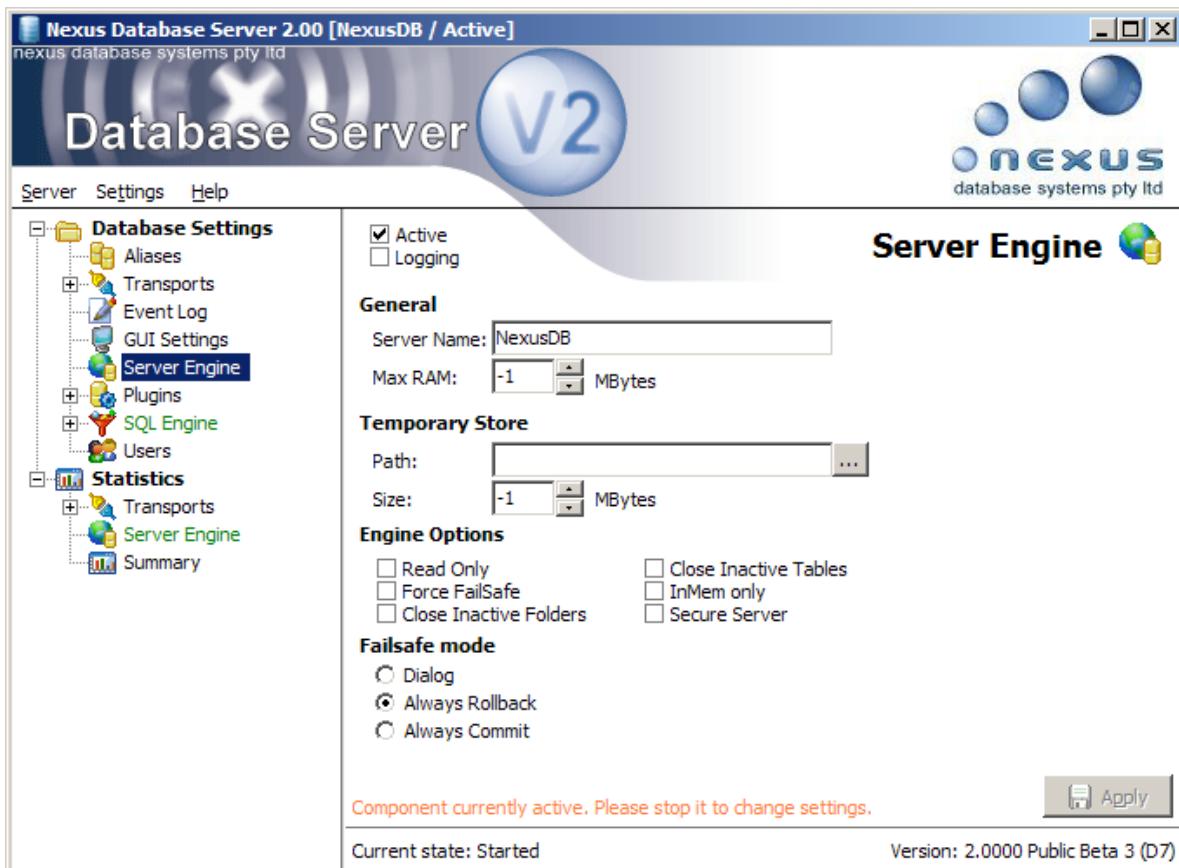


6.2 The Server User Interface

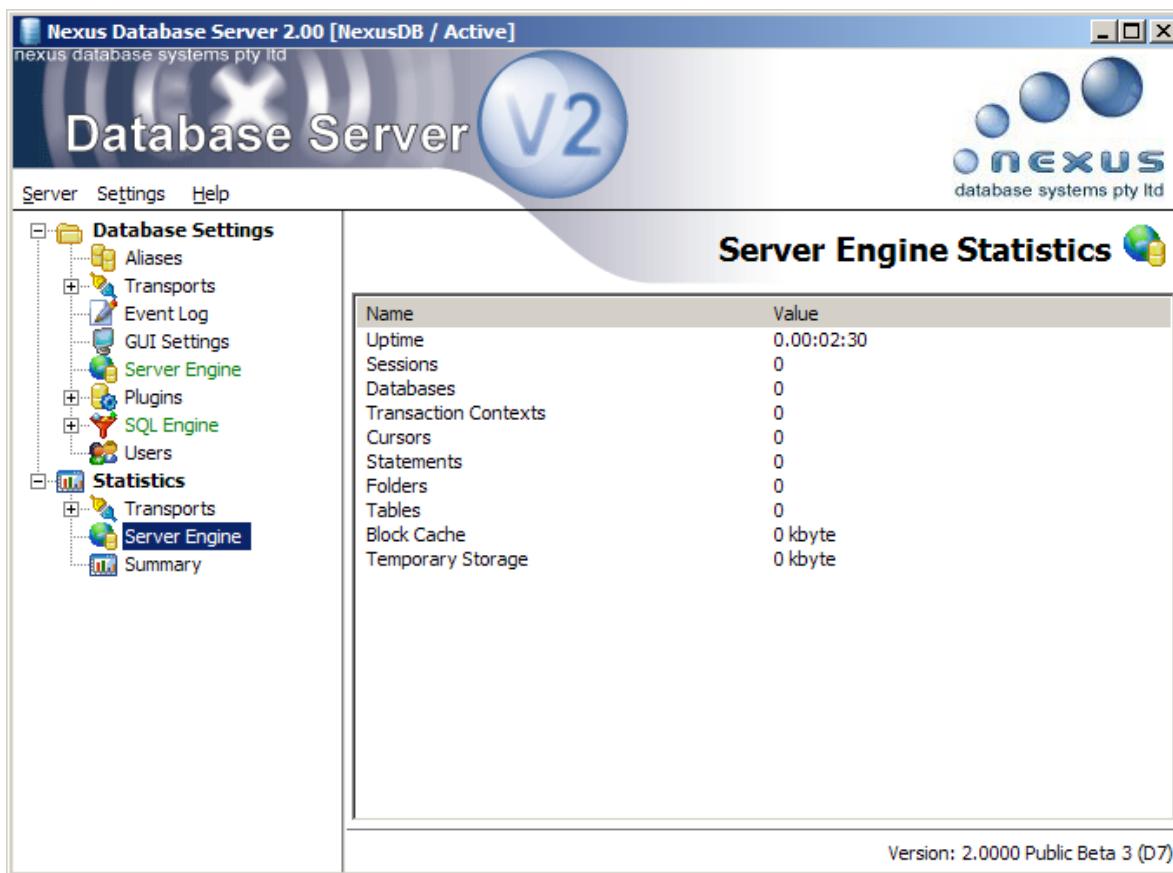
The Server User Interface



The NexusDB Server has a very clear and simple user interface. On the left hand side you find a tree view showing you all server modules that are available on this particular server. There are two main sections: Database Settings, which allows you to change the settings of each module,



and Statistics, which gives you an overview on the workload/status of the different modules.



The items in the tree view are colour coded: all activated modules are shown in green in the tree view; all stopped ones orange and inactivate ones red. Black modules are stateless modules that can be changed at any time.

For a better overview and easier reading, the tree view also allows categorizing of items, as you can see in the above example on the Transports node. Click on the usual +/- symbols to expand or collapse a category level in the tree view.

©2004 Nexus Database Systems Pty Ltd.



6.2.1 Settings Panel

Settings Panel



In this section you will get a detailed description for all supported settings in the default Server. Please note that the server user interface is easily and almost generically extensible, thus the listing will not be complete for changed servers.

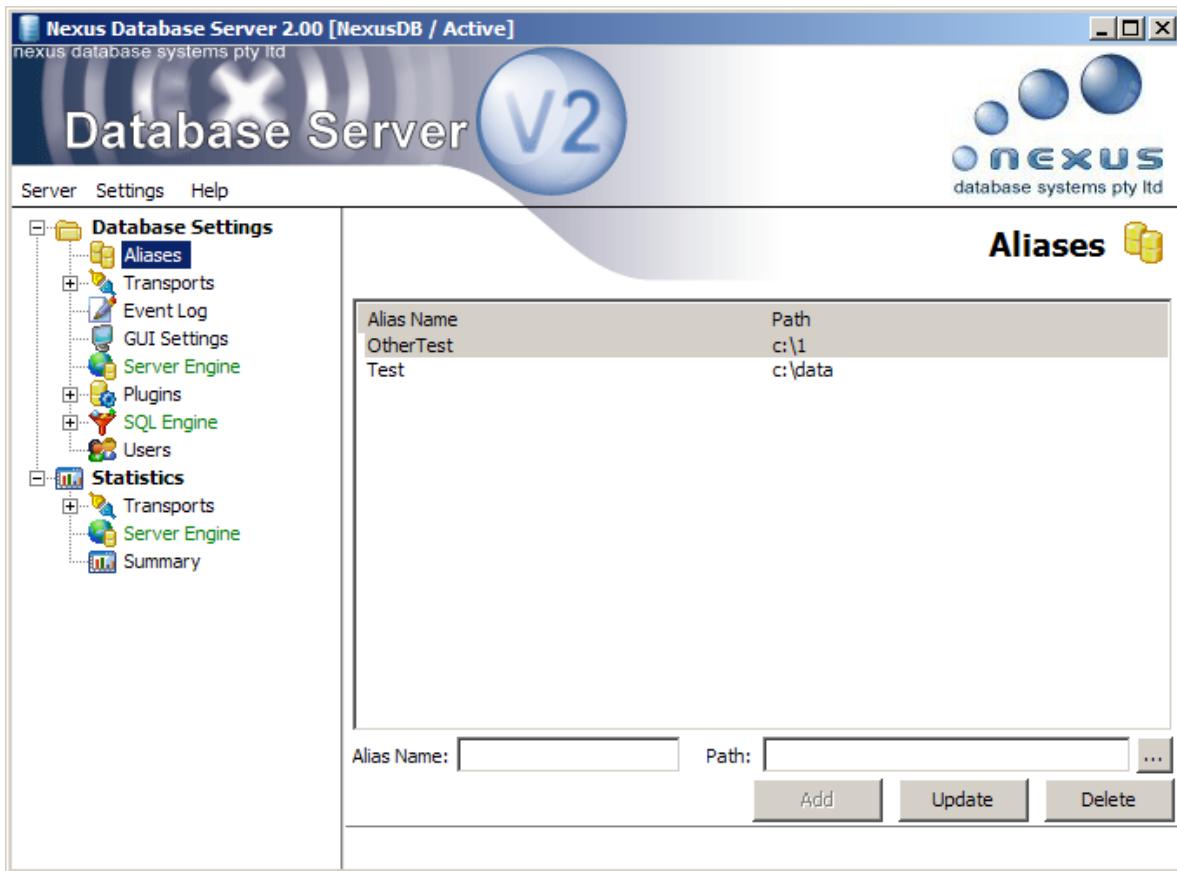
©2004 Nexus Database Systems Pty Ltd.



6.2.1.1 Aliases

Aliases

The Aliases window is divided into two parts: the list of current active aliases on top and the entry fields and action buttons for adding, updating and deleting aliases below.



Alias Name

NexusDB Alias names are restricted to comply with the SQL:2003 Regular Identifiers definition.

Path

The path can either be a fixed path like "c:\testdb\" or a fully qualified UNC name like "\yourserver\nexusdatabase\testdb". If you're working with Win NT services, make sure that the server can access the path when it is started. For example substituted drives or network connected drives are likely not visible at the time the service starts. Either make the NexusDB Server Service dependent from the needed network services or change the path to a value that the server can access at start time. Failing to do so will make the database inaccessible to clients.

Action Buttons

The action buttons are only activated if the according action is possible, otherwise they are disabled.

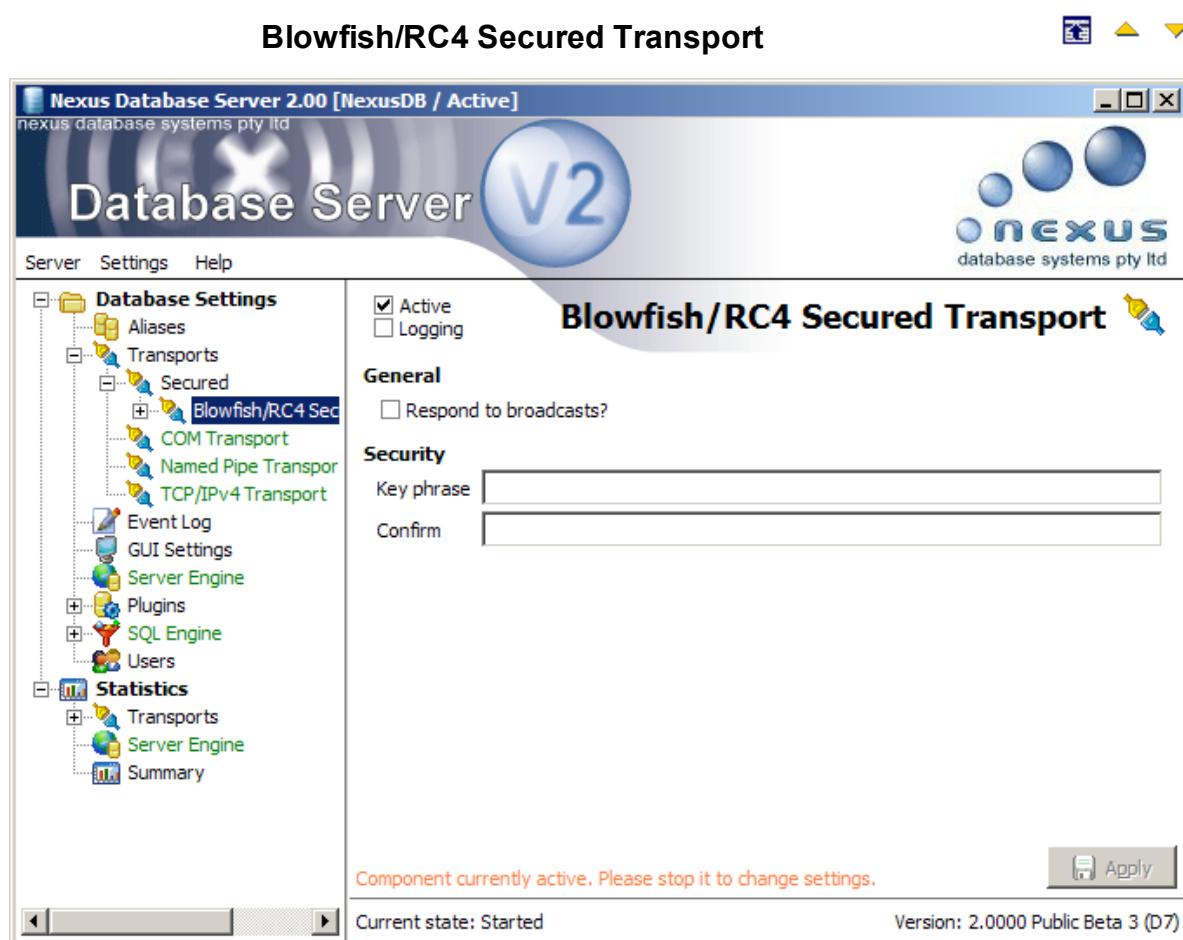
Pressing the **Add** button creates a new alias and is immediately available for all connected sessions. The **Update** action changes Alias Name and Path or both of the selected items in the list. The **Delete** button deletes the currently selected item after confirmation.

©2004 Nexus Database Systems Pty Ltd.



6.2.1.2 Transports

6.2.1.2.1 Blow fish/RC4 Secured Transport



The Blowfish/RC4 Secured Transport is a so-called Wrapper Transport. It encrypts requests and responses before they are sent via another regular transport like TCP or Named Pipes. It has only a few options:

Respond to Broadcast

When this option is active, it instructs the server to send back its identification and version number on broadcast requests received via the regular transport being used.

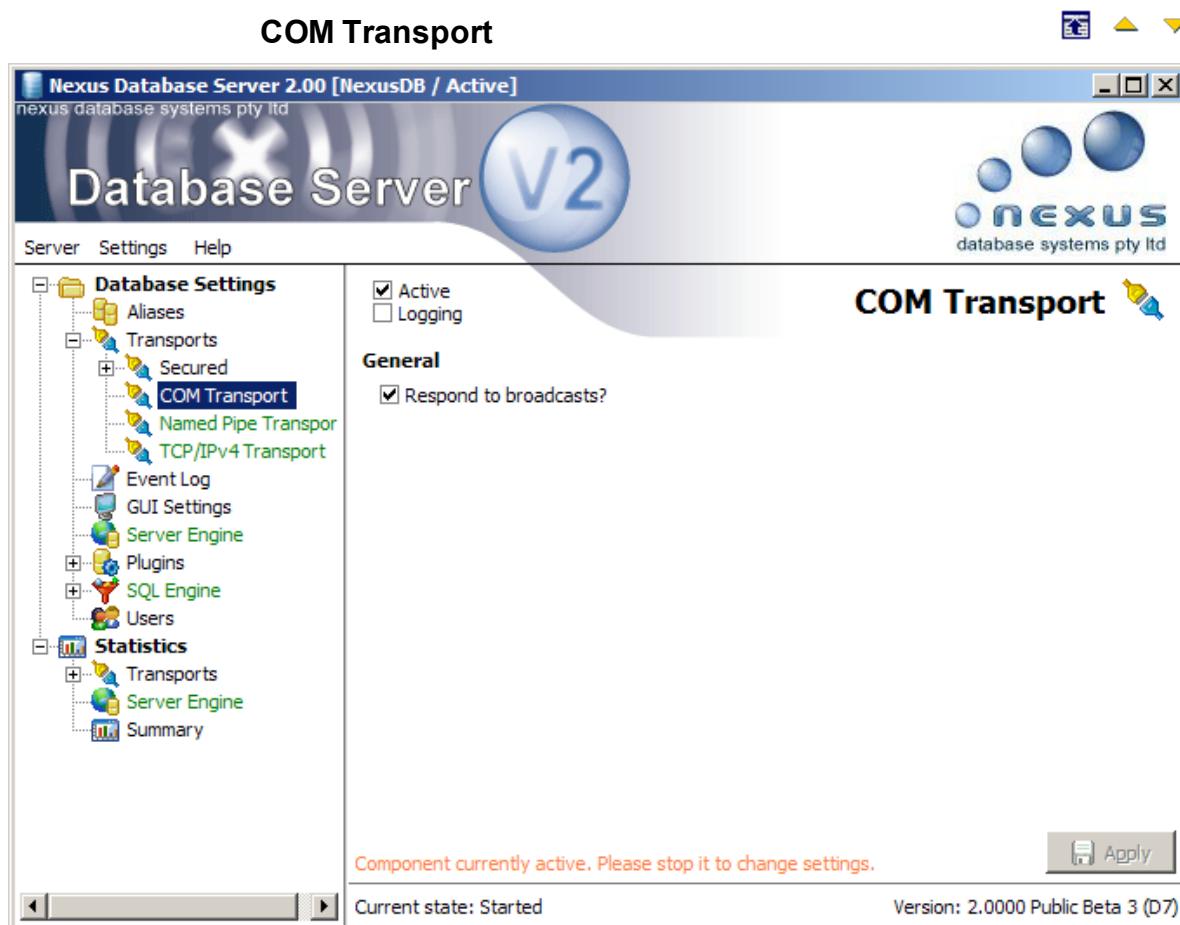
Key Phrase

Key Phrase is used for the encrypting your data. Please note that the same key phrase is needed on the client side for the communication to be successful.

©2004 Nexus Database Systems Pty Ltd.



6.2.1.2.2 COM Transport



The COM Transport only supports a single option:

Respond to Broadcast

When this option is active, it instructs the server to send back its identification and version number on broadcast requests received via the regular transport being used.

©2004 Nexus Database Systems Pty Ltd.

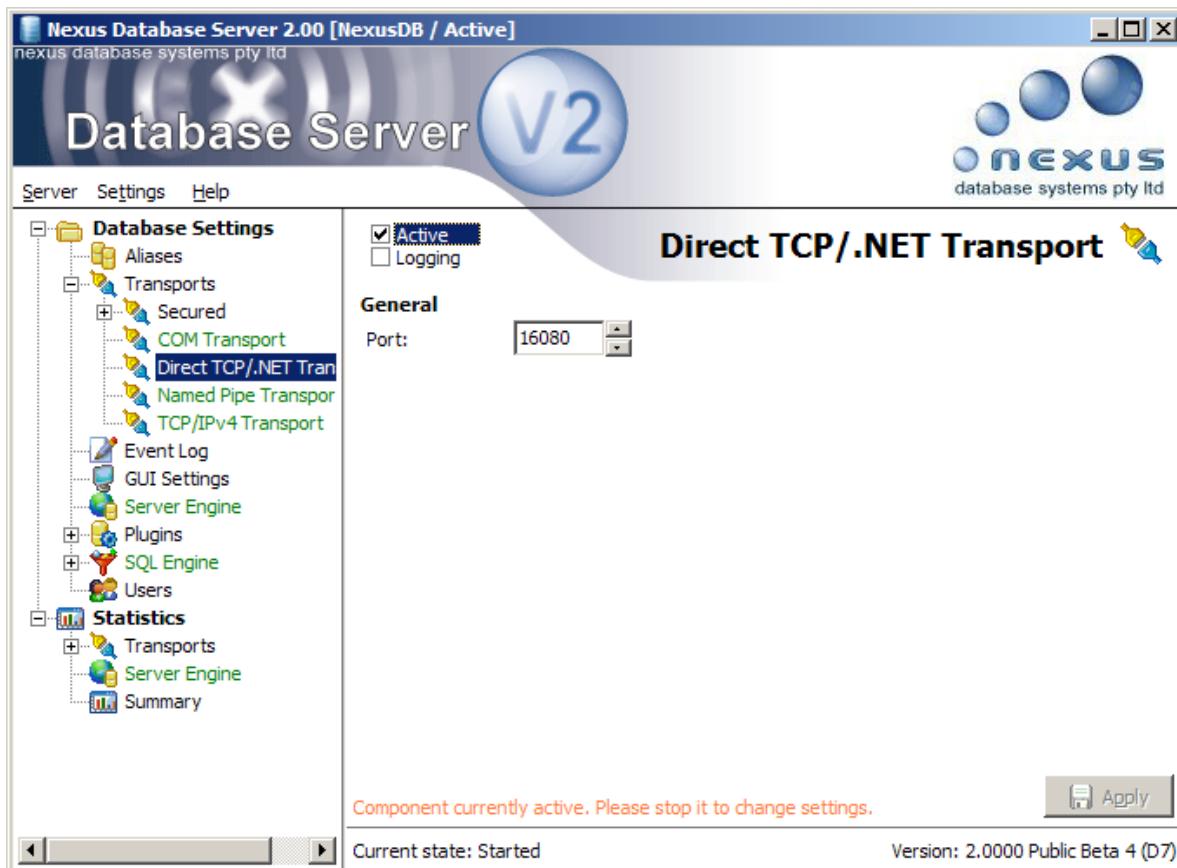


6.2.1.2.3 Direct TCP/.NET Transport

Direct TCP/.NET Transport



This transport is technically seen completely different from all others. It is actually not even a read NexusDB transport, but merely a simple TCP server that translates simple incoming request and sends, executes requests and sends back data. You can, if you like see it as a built-in middle-tier server.



If you want to use the Direct Mode of the NexusDB ADO Provider, this transport has to be active, because the ADO Provider uses it for communication.

There is (currently) only one option:

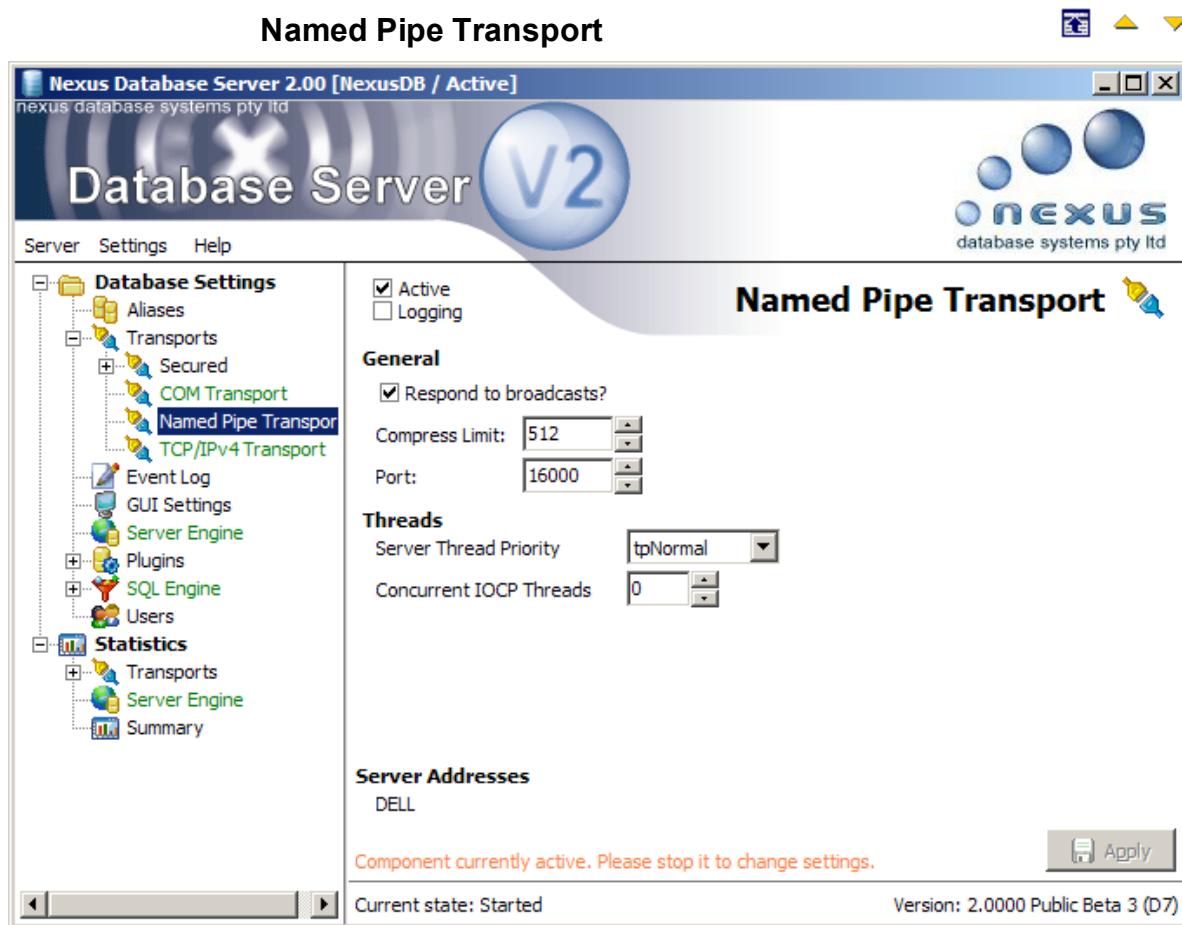
Port

The Port is the TCP port this transport listens on for requests. Make sure that it is routed through any possible firewall. The default value is 16080.

©2004 Nexus Database Systems Pty Ltd.



6.2.1.2.4 Named Pipe Transport



The Named Pipe Transport has a few options:

Respond to Broadcast

When this option is active, it instructs the server to send back its identification and version number on broadcast requests received via the regular transport being used.

Compress Limit

Compress Limit sets the minimum number of bytes at which the server initiates data compression for sent messages. If a message size is below that value compression is ignored. The server always responds to compressed requests by compressing the answer, even if the size of the answer is below the compression limit.

Port

Port in Named Pipe Transport is merely a unique identifier added to the pipe name. This way it is possible to run multiple Named Pipe Protocols in one or more servers on the same machine. **Please note** that this has to match with the client side for successful communication.

Server Thread Priority

The Server Thread Priority sets, as the name says the priority of the server thread. The server thread of a transport is the one that actually does the work for each request. Be careful with changing thread priorities as there is a fine balance between the different threads of the server needed to achieve optimal performance. For normal processing mode you should generally leave the priorities at their default settings. Valid settings are tpdle, tpLowest, tpLower, tpNormal, tpHigher, tpHighest and tpTimeCritical.

Concurrent IOCP Threads

Concurrent IOCP Threads is the number of threads processing IOCP (I/O Completion Port) signals. If this number is 0, it means that there is one thread initialised for each processor in the server machine. For general use, this is the optimal setting. For special purposes, it might be useful to change these settings. **Please note** that this setting is ignored for machines with a Windows version below Win NT. On these machines, IOCP is not available and is emulated and ignores this setting.

Server Addresses

Finally in the Server Addresses field you see the valid addresses of the server for this transport, which you have to use in the form `ServerName@Address` for connecting to this server.

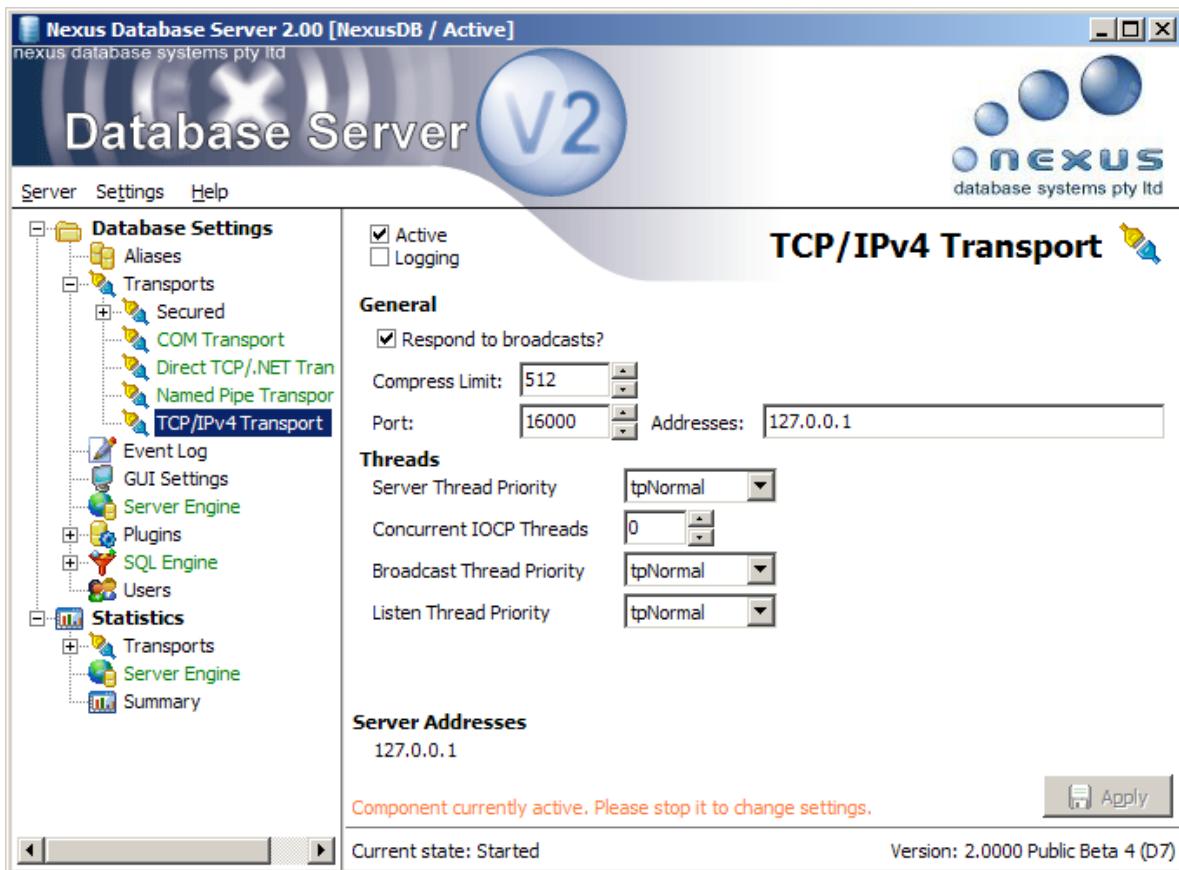
©2004 Nexus Database Systems Pty Ltd.



6.2.1.2.5 TCP/IPv4 Transport

TCP/IPv4 Transport





The TCP/IPv4 Transport has quite a few options as well, overlapping with the options for the Named Pipe Transport:

Respond to Broadcast

When this option is active, it instructs the server to send back its identification and version number on broadcast requests received via the regular transport being used.

Compress Limit

Compress Limit sets the minimum number of bytes at which the server initiates data compression for sent messages. If a message size is below that value compression is ignored. The server always responds to compressed requests by compressing the answer, even if the size of the answer is below the compression limit.

Port

Port for TCP/IPv4 is the IP port used for communication. Please note that this has to match with the client side for successful communication.

Adresses

Use the Addresses field to set the TCP addresses the transport will be bound to as a comma delimited list. You can use the asterisk (*) to bind the transport to all available addresses, which is also the default.

Server Thread Priority

The Server Thread Priority sets, as the name says the priority of the server thread. The server thread of a transport is the one that actually does the work for each request. Be careful with changing thread priorities as there is a fine balance between the different threads of the server needed to achieve optimal performance. For normal processing mode you should generally leave the priorities at their default settings. Valid settings are tpIdle, tpLowest, tpLower, tpNormal, tpHigher, tpHighest and tpTimeCritical.

Concurrent IOCP Threads

Concurrent IOCP Threads is the number of threads processing IOCP (I/O Completion Port) signals. If this number is 0, it means that there is one thread initialised for each processor in the server machine. For general use, this is the optimal setting. For special purposes, it might be useful to change these settings. **Please note** that this setting is ignored for machines with a Windows version below Win NT. On these machines, IOCP is not available and is emulated and ignores this setting.

Broadcast Thread Priority

Broadcast Thread Priority sets priority for the thread that is responsible for receiving and answering broadcast over for servers

Listen Thread Priority

Listen Thread Priority is the priority for the thread that is responsible for receiving the request and putting it into the Server Thread Queue.

Server Addresses

Again, in the Server Addresses field you see the valid addresses of the server for this transport, which you have to use in the form `ServerName@Address` for connecting to this server.

As mentioned the various priorities should be left alone unless the user knows exactly what he is doing and its impact on the server performance.

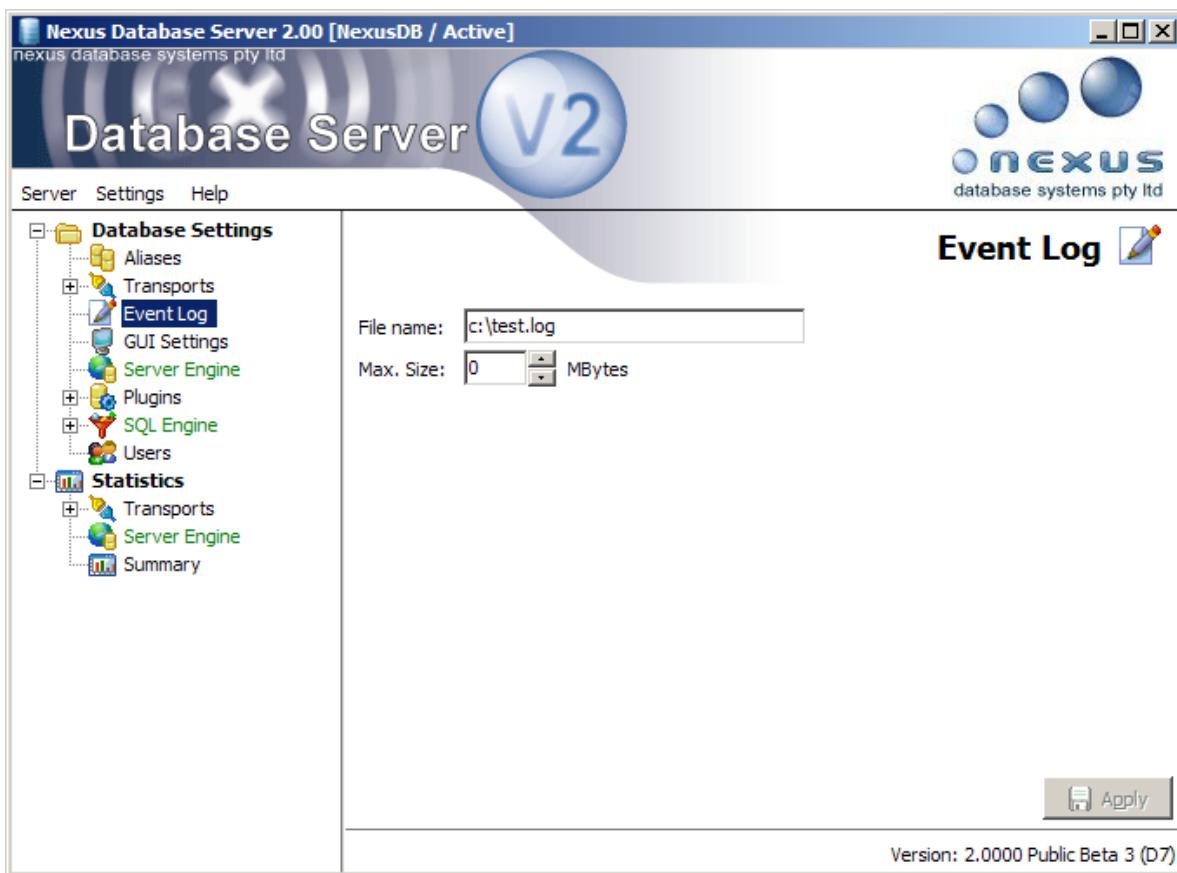
©2004 Nexus Database Systems Pty Ltd.



6.2.1.3 Event Log

Event Log





The Event Log has two options.

File Name

First one is the file name the event log is saved to.

Max Size

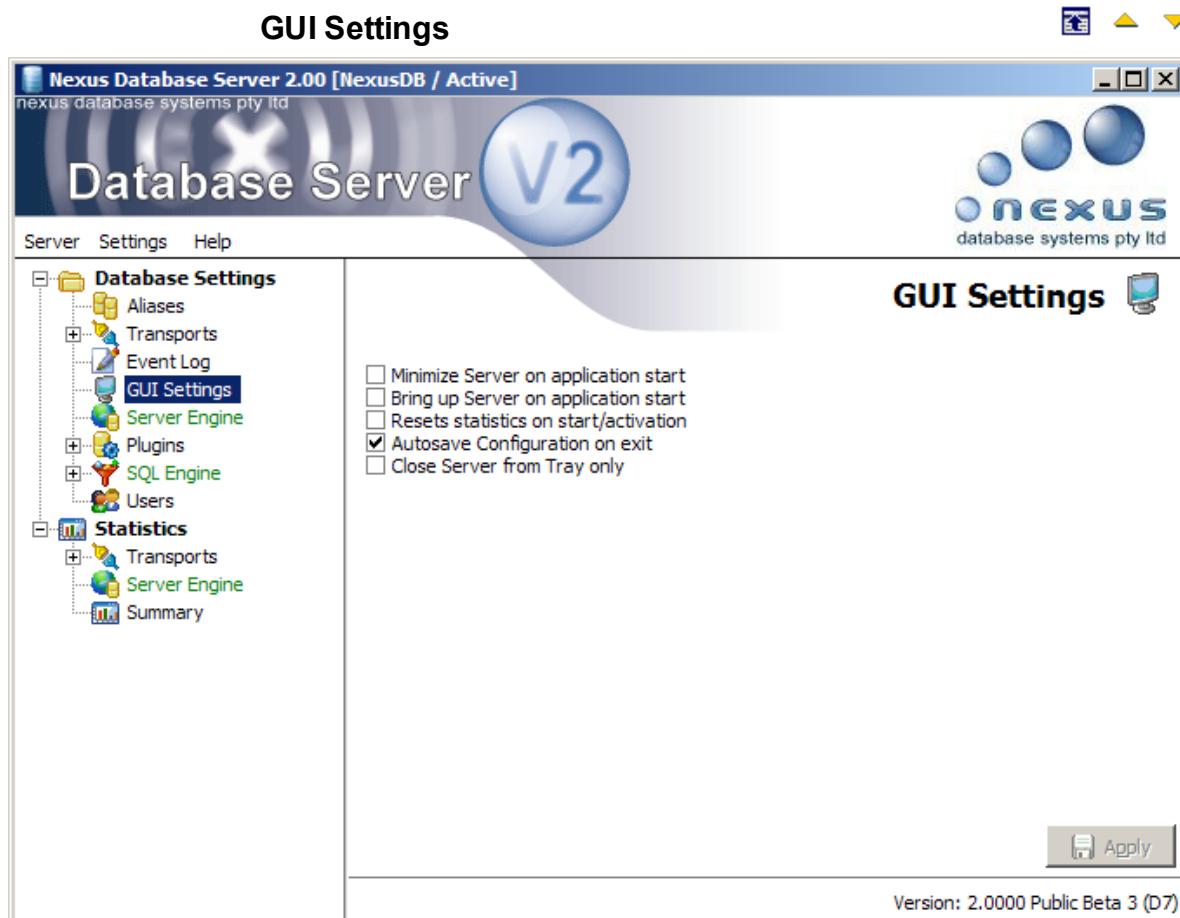
The second option is the maximum size of the event log in Megabytes. The current implementation of NexusDB only writes sparse information to the log files, mainly extended errors and exception information. If the file size reaches the maximum size it will truncate the file in 1 kilobyte 'blocks'.

All components used in the default NexusDB server are linked to the Event Log, thus all logging will end up in the given file. If the file name is empty ("") the logging will be disabled even if the component is set to active.

©2004 Nexus Database Systems Pty Ltd.



6.2.1.4 GUI Settings



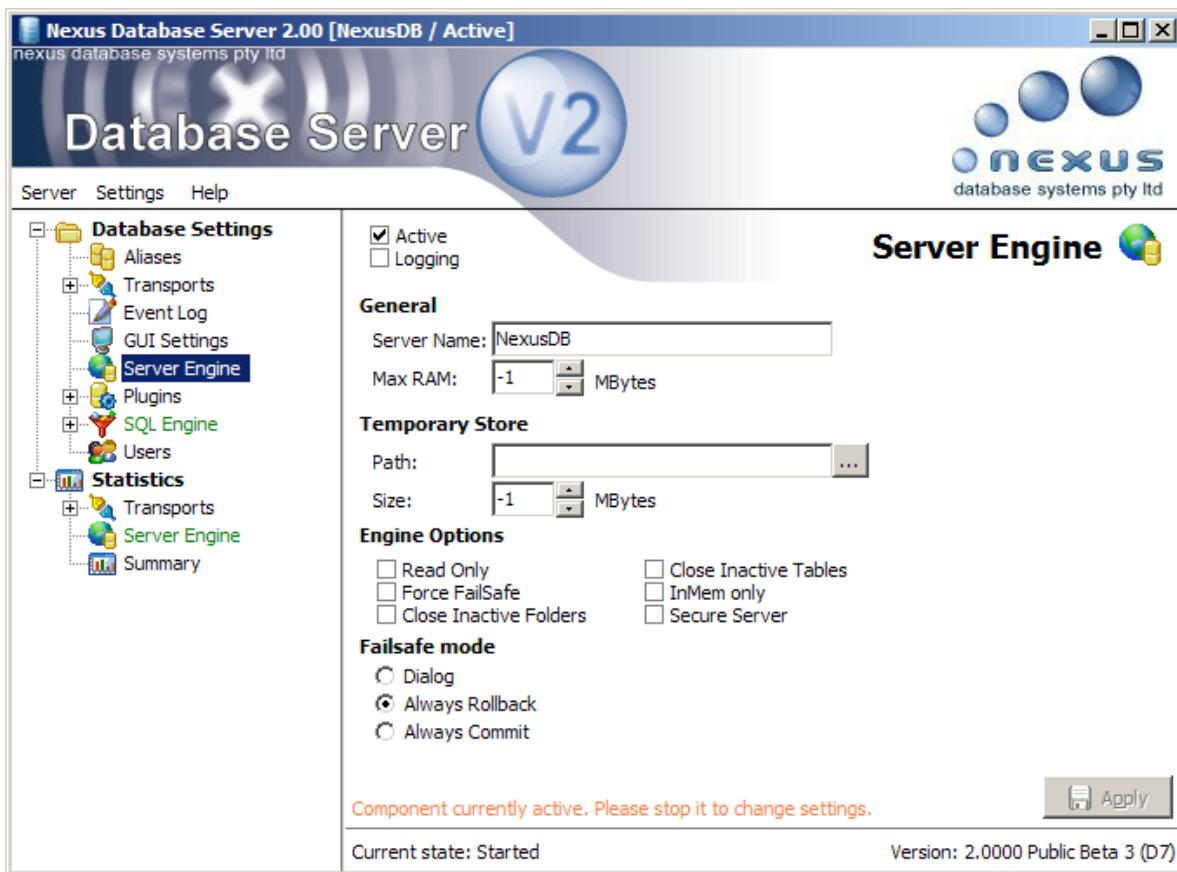
The GUI settings are purely settings for the user interface and start up mode and do not influence a running NexusDB server. The options are self-explanatory and mentioned here for completeness only.

©2004 Nexus Database Systems Pty Ltd.



6.2.1.5 Server Engine





The Server Engine Settings define how the core server engine operates.

Server Name

As a unique identifier, the server needs a name, which is entered in the Server Name option.

Max RAM

Max RAM defines the maximum amount of physical memory the server engine is allowed to allocate for its operation. This setting is in Megabytes and should be set. Best performance is achieved if it is set to a value large enough so that all tables fit into this memory. If this value is set to -1, the server will use a maximum of 50% of the physical memory available to the machine. This setting is very important for optimal performance of NexusDB and should be adjusted from time to time as the database size grows above its initial setting.

Temporary Storage

The settings for Temporary Storage are Path and Size. NexusDB uses its own temporary storage management and files instead of using the normal windows page file. This increases its performance and allows for dedicated virtual memory allocation for the database server. The files that NexusDB creates are stored in the directory identified by the path setting and have the maximum size stated in the size setting. The files are created as maximum size when the server engine is started with a special file system flag which makes sure they are deleted as soon as the process ends, even if it is killed (e.g. via the task manager or Delphi).

Engine Options

The Engine Options enable the user to change certain behaviour of the database server.

- The **read only** setting sets **all** databases and thus tables of the server to read only. This means also that the SQL data manipulation commands (insert, update, etc.) are not enabled anymore. It is still possible though to create in-memory tables.
- This leads us directly to the **InMem** only option, which is restricting the database server to in-memory tables only. No tables are saved to disk if this option is enabled and thus if the server is closed all data is lost.
- **Close Inactive Tables** and **Close Inactive Folders** influence the file caching of the server. Every open database and table in NexusDB has an exclusive file lock. This ensures that no other server or application can access the files while they are opened by the NexusDB server. In default configuration, these two options are disabled and the server keeps the locks once they are established and thus can safely keep them in cache if there is enough room. If a client reopens the tables/databases again, the server engine can access them from the much faster memory instead of having to access them on disk again.
- **Force Failsafe** makes the server work in failsafe mode only. Please note that this slows down the server significantly, but in certain circumstances and environment, data safety and consistency is essential. If the option is enabled the server engine writes a journal for every transaction and only if the transaction is successfully completed and the journal committed the actual transaction is considered successful. Otherwise the engine automatically applies the action set in the Failsafe mode setting.
- Finally the **Secure Server** option forces users to logon when establishing a connection **and** an admin user to log on to the server user interface. Please look at the following paragraph for more information on Users.

Failsafe Mode

If Force Failsafe is enabled, this option defines which action the server takes on restart, if he finds that a previous transaction was not successful.

The default option is **Always Rollback**, which always undoes unsuccessful transactions. **Always Commit** is the opposite while **Dialog** pops up a dialog and lets the user decide what to do.

Please note that Dialog should **never** be used if the server is running as a service or in unattended mode, as the server will not start until a Failsafe mode was selected in this dialog.

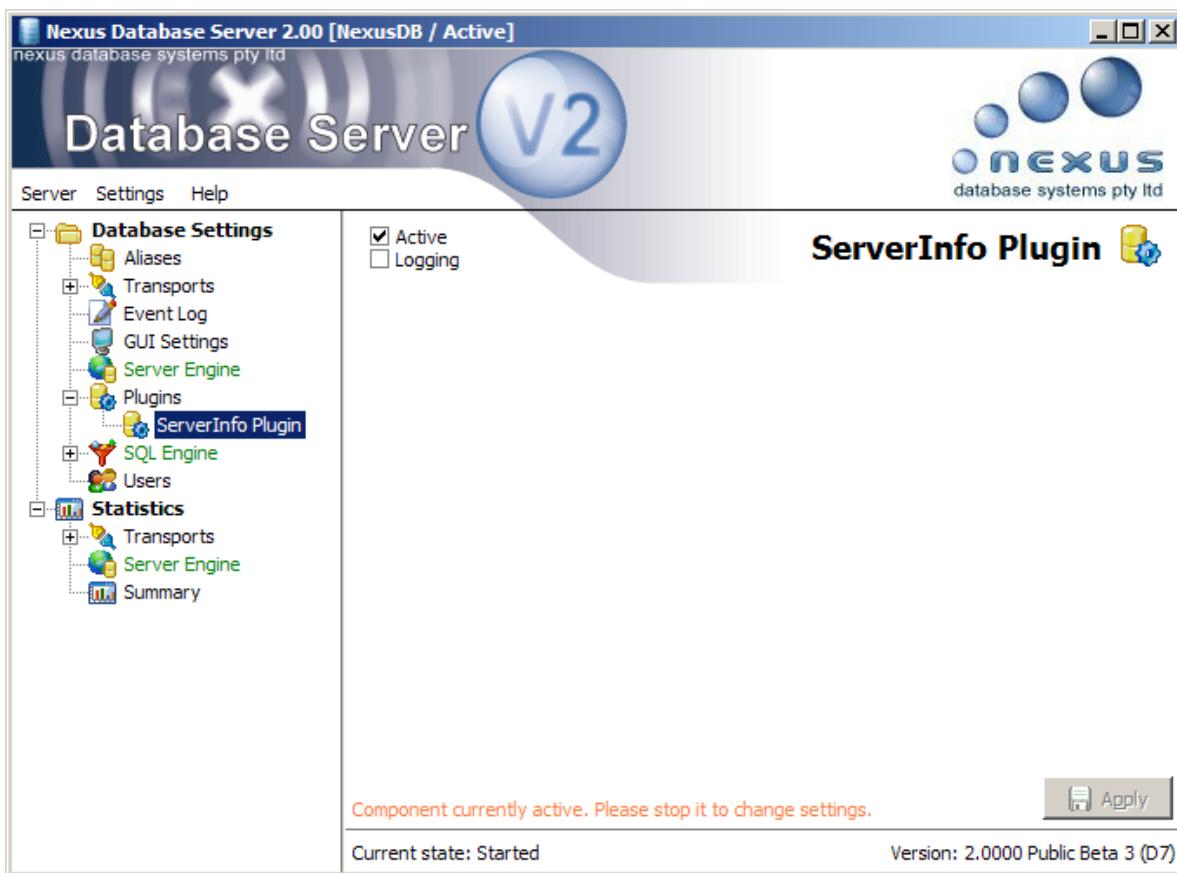
©2004 Nexus Database Systems Pty Ltd.



6.2.1.6 Plugins

Plugins





The NexusDB Server comes with one plugin already registered - the Server Info Plugin. This plugin gives client applications access to a host of information about the server such as date and time and so on. The full use of the plugin is available to all client applications built in Delphi and is covered in the Delphi Developers section of the documentation.

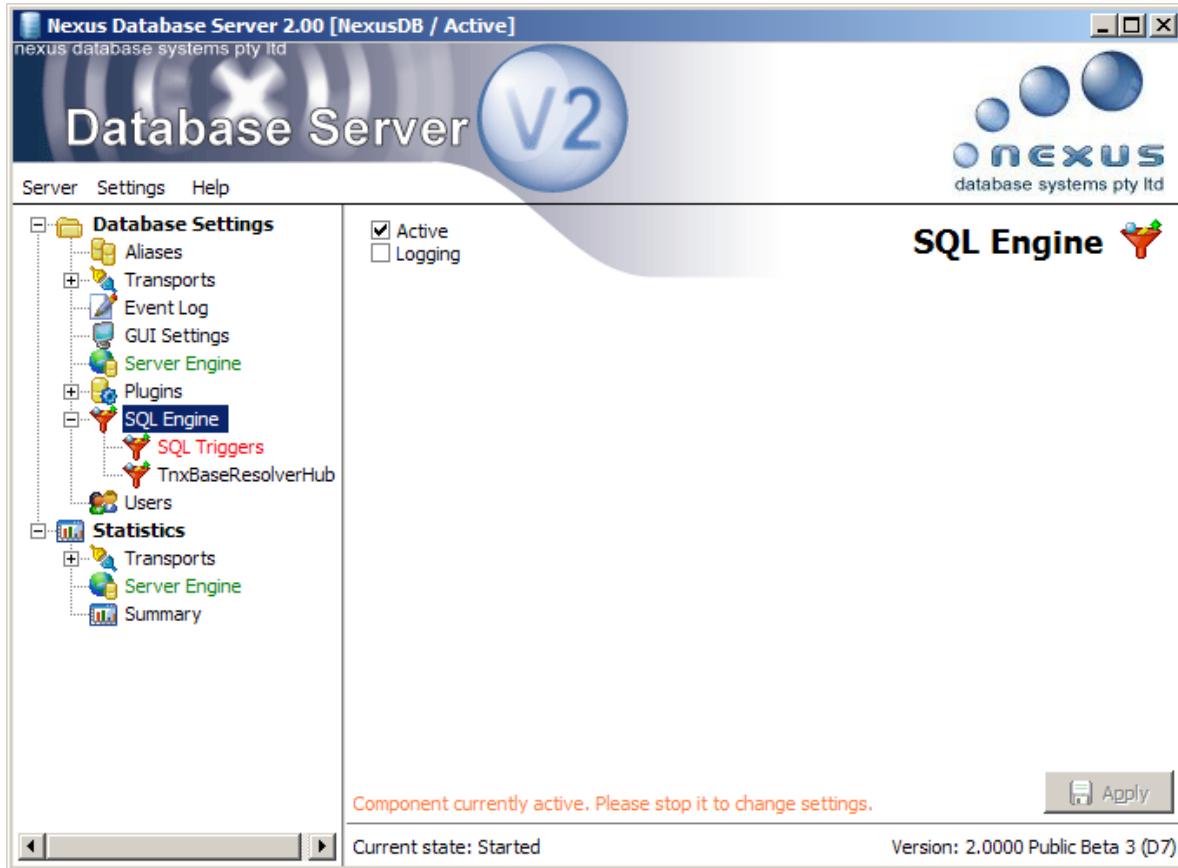
©2004 Nexus Database Systems Pty Ltd.



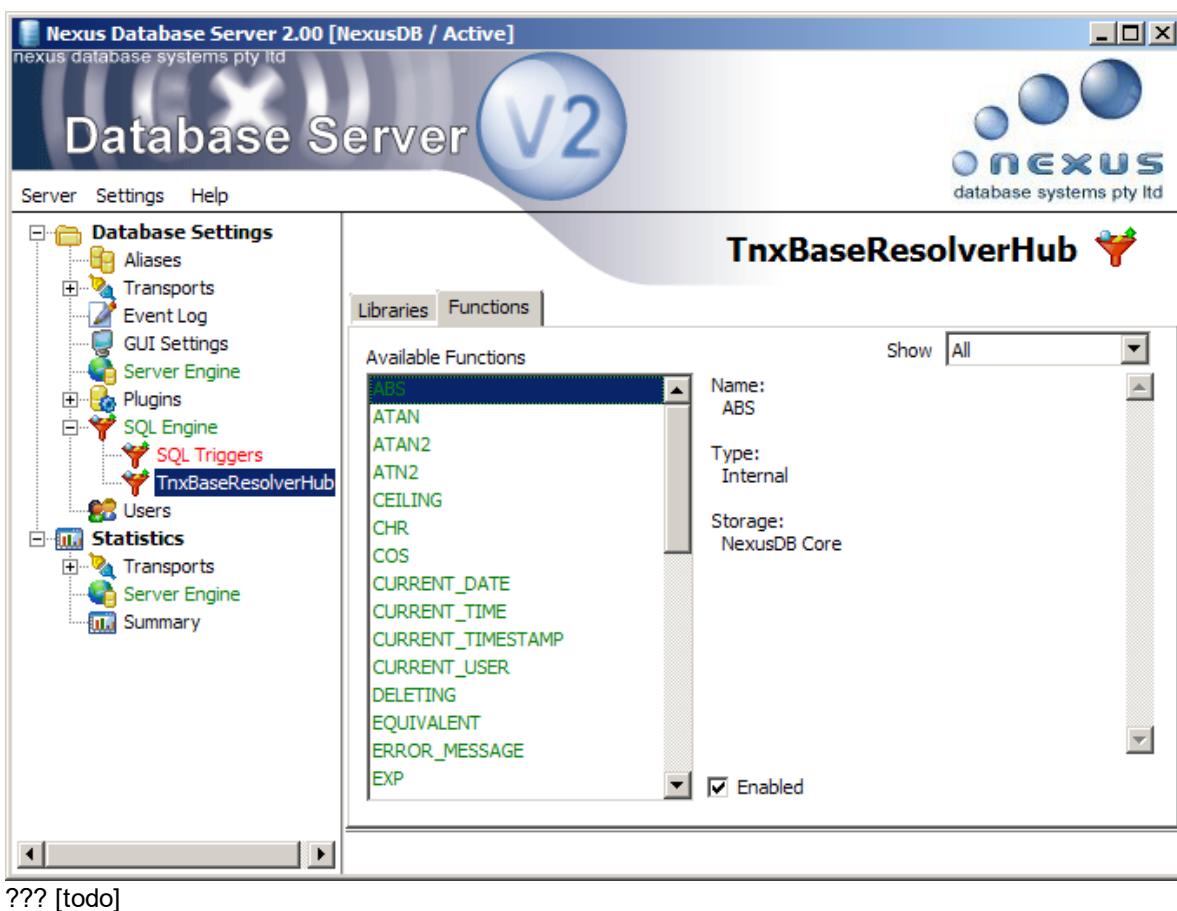
6.2.1.7 SQL Engine

SQL Engine





This page allows you to maintain the SQL engine, Triggers and User Defined Functions (UDFs). The Engine and Triggers can be activated and deactivated. The UDF page allows similar maintenance of the functions and libraries registered as shown in the following screen snapshot.



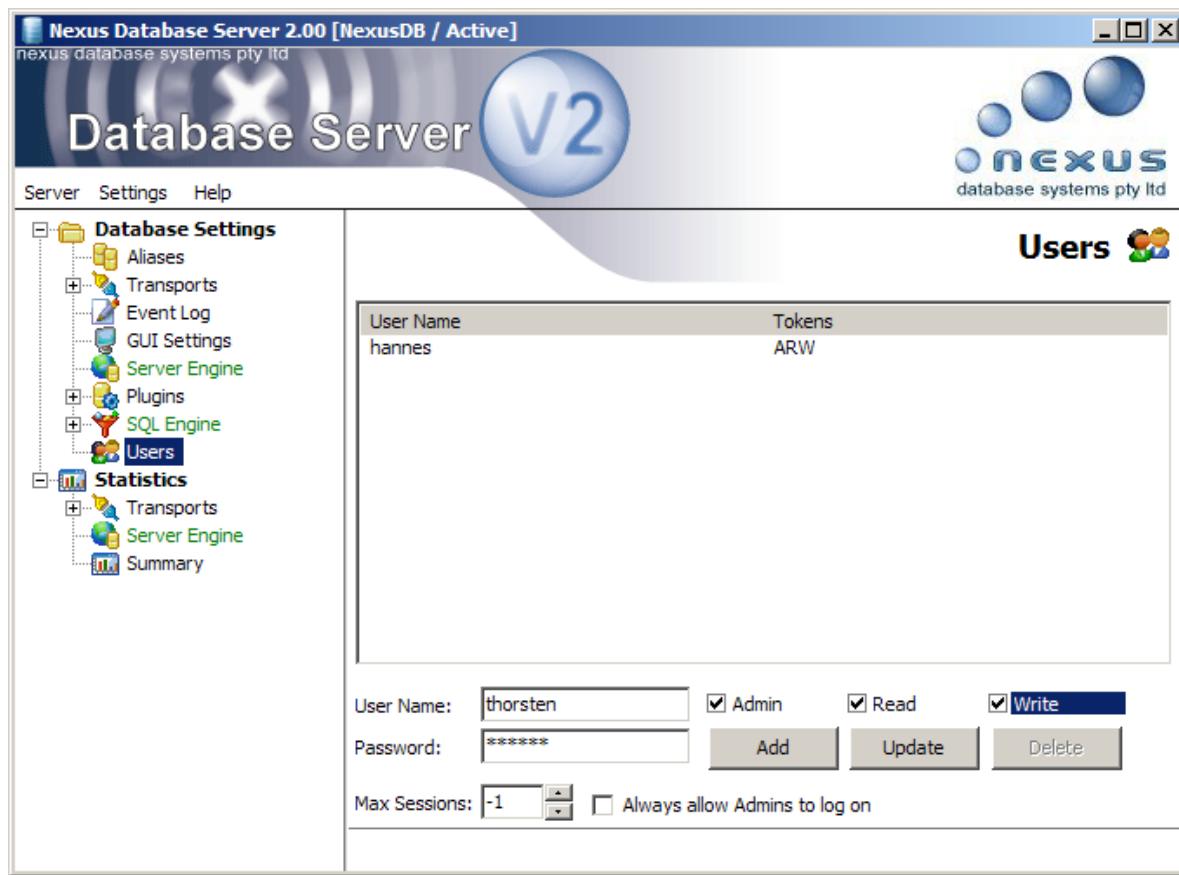
©2004 Nexus Database Systems Pty Ltd.



6.2.1.8 Users

Users





The Users options work in combination with the Secure Server setting of the Server Engine. If this option is set and at least one user with administration rights is defined, the server enforces a log on when a session or the server user interface is connected.

Add Users

Adding a user is straightforward, just enter User Name, Password and select the rights for the user. Then press the Add button to create a new user.

Change a User

To change a user, first select the one you want to change in the list and the details will be visible in the edit fields. Update the settings and press update and the selected user in the list will be replaced with the settings just entered.

Delete a User

If you click Delete, the selected line and thus the user is deleted.

Max Sessions

The Max Sessions option was added to limit the number of sessions that can be created on the server at any time. A client that tries to connect to the server once this limit is reached gets an according

error message. For this option to work the Secure Server option of the Server Engine must be activated.

If you still want an Administrator to be able to logon (even if the limit is reached) please set the **Always allow Admins to log on** to true.

Please note that these settings are applied immediately and are not reversible.

©2004 Nexus Database Systems Pty Ltd.



6.2.2 Statistics Panel

6.2.2.1 Transports

Transports

 A screenshot of the Nexus Database Server 2.00 interface. The title bar says "Nexus Database Server 2.00 [NexusDB / Active]" and "nexus database systems pty ltd". The main window has a header "Database Server V2". On the left is a navigation tree with "Database Settings" (Aliases, Transports, Event Log, GUI Settings, Server Engine, Plugins), "SQL Engine", "Users", and "Statistics" (Transport, Secured, COM Transport, Named Pipe Transport, TCP/IPv4 Transport). The "TCP/IPv4 Transport Statistics" tab is selected. The right pane shows a table of statistics:

Name	Value
Uptime	00:08:45
Local Address	127.0.0.1
Total Sockets	0
Connected Transports	0
Open Sessions	0
Messages sent	0
Actual bytes sent	0
Uncompressed bytes sent	0
Ratio sent	1.00
Messages received	0
Actual bytes received	0
Uncompressed bytes received	0
Ratio received	1.00

Version: 2.0000 Public Beta 3 (D7)

Each of the various transport statistics pages contains almost identical information and a typical screen shot is shown above. Each statistic provided is self-explanatory. Mostly these screens are

used in a diagnostic manner to, for instance, confirm that messages are being sent and/or received by the server.

©2004 Nexus Database Systems Pty Ltd.



6.2.2.2 Server Engine

A screenshot of the Nexus Database Server 2.00 interface. The title bar reads "Nexus Database Server 2.00 [NexusDB / Active]" and "nexus database systems pty ltd". The main window is titled "Server Engine" and features a large "Database Server V2" graphic. The menu bar includes "Server", "Settings", and "Help". On the left, a tree view shows "Database Settings" (Aliases, Transports, Event Log, GUI Settings, Server Engine, Plugins, SQL Engine, Users) and "Statistics" (Transports, Server Engine, Summary). The right pane is titled "Server Engine Statistics" and contains a table of statistics:

Name	Value
Uptime	00:09:05
Sessions	0
Databases	0
Transaction Contexts	0
Cursors	0
Statements	0
Folders	0
Tables	0
Block Cache	0 kbyte
Temporary Storage	0 kbyte

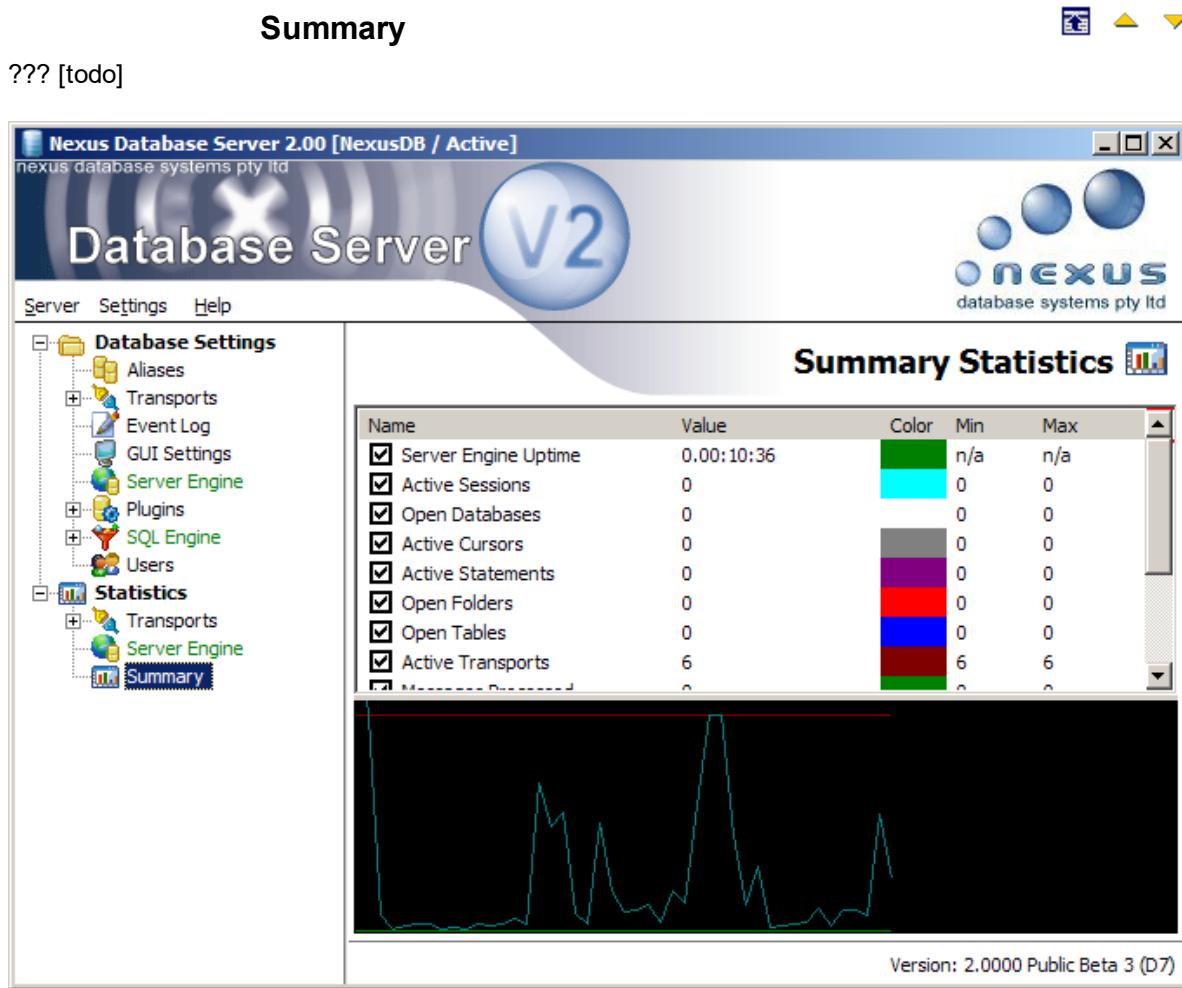
Version: 2.0000 Public Beta 3 (D7)

The Server Engine statistics allow administrators to monitor the usage of the engine for all connected users. The snapshot above, for example, shows a server with no clients connected. As with the Transports statistics, this is useful for both monitoring and diagnostic purposes.

©2004 Nexus Database Systems Pty Ltd.



6.2.2.3 Summary



©2004 Nexus Database Systems Pty Ltd.



6.3 Configuring and Testing the NexusDB Server

Configuring and Testing the NexusDB Server

Here are the steps to quickly configure the NexusDB Server and verify that it works correctly:

Step one is to start and setup the server.

1. Start up NexusDB server executable (nxServer.exe)

2. Select the Server Engine item from the Database settings on the left hand tree view. The Server Engine configuration displays.
3. Make sure that a Server Name is specified and MaxRAM is set to a sensible size. You might also want to change other settings.
4. Select the Aliases item from the NexusDB Server main menu. The Alias Configuration window displays.
5. Add aliases and verify that your alias paths are defined correctly.
6. Now activate the Server modules by selecting "Start all modules" from the Server menu. You can also (de) activate single modules by selecting the item in the tree view and checking the active check box on the top of each configuration.
7. Make sure you have at least one transport active! (Note: The Named Pipe transport needs the server to be running on Win NT or higher. Clients are working on Win 9x/Me.)

The second step is to verify you can connect to the NexusDB Server using Enterprise Manager on the same machine. Start NexusDB Enterprise Manager and look out for the servers found. If your server name appears your server is set up correctly. This will verify you can at least see the NexusDB Server from the same machine.

The third step is to verify you can connect to the NexusDB Server from a different workstation. When Enterprise Manager starts, it sends out a broadcast for available servers. Both the TCP/IP and Named Pipes transports should show the NexusDB Server on the other machine.

©2004 Nexus Database Systems Pty Ltd.



6.4 Running the Server as a WinNT Service

Running the Server as a Win NT Service



The server can be installed as a Win NT service. For this you need to be logged in as administrator and type

Install

```
nxServer /install
```

Uninstall

```
nxServer /uninstall
```

When installing the server it will prompt you for a user account and password the server should run as. This is expected in the format DomainName\AccountName. Domain Name can be "." for the local domain. Please note the selected account must have the "logon as service" rights. You can assign these to a user account with the Microsoft® policy editor. To do this in Win XP, go to the Start menu then select Control Panel/Administrative Tools/Local Security Policy. The policy editor will start. In the tree view select Local Policies/User Rights Assignment. On the right hand panel look for "Logon As Service" and double click it. Press "Add User or Group" and add the wanted Account Name to the list of allowed accounts.

The service will be installed as Automatic; this means it will be started on any subsequent machine boot. Once the service is installed the application prompts if the service should be started immediately.

We've also added some optional parameters for your convenience. The complete set is:

```
Help, ?      Shows Help
Install      Installs the
service
Uninstall    Uninstalls the
service
Start        Start Service after
installation
Nostart      Do not ask for
starting service
Silent       No Dialogs at all
Username:name User account
in "domain\username" form
for the service
Password:pw   password of
the service account
```

If neither nostart nor start and silent are specified, the installation asks if the service should be started.

Examples

```
nxServer.exe /install
```

Will prompt for account, then install the service and ask if it should be started

```
nxServer.exe /install /usern
ame:.
\test /password:test /silent
```

Will (try to) install the service with the given account. If successful it will not start the service or prompt for it.

```
nxServer.exe /install /username:  
\\test /password:test /start
```

Will (try to) install the service with the given account. If successful it will start the service and tell you that it was installed and started.

```
nxServer.exe /install /username:  
\\test /password:test /nostart
```

Will (try to) install the service with the given account. Will not start or prompt for starting the service and tell you that it was installed.

```
nxServer.exe /install /username:  
\\test /password:test /start  
/silent
```

Will (try to) install the service with the given account. If successful it will start the service. No messages will appear.

©2004 Nexus Database Systems Pty Ltd.



6.5 The Server Settings File

The Server Settings File



The NexusDB settings storage was completely rewritten for Version 2 and now saves its settings in a plain INI file.

The configuration file not only holds the actual settings, but also the complete definition of all available settings. This means it is a completely extensible format and you can in fact even build a separate user interface for the server by reading the available settings data and build the controls accordingly. Before you start this on your own now though, this will be part of the remote server configuration feature coming with NexusDB V2 Enterprise.

When deploying the Server to clients, NexusDB customers often need to setup Aliases, Users and default settings for the server. This is best done by integrating this into the application setup procedure. The initialisation file is called nxServer.nxdbworksettings and has to be created in or copied into the same directory as the nxServer executable. The file works just like a normal windows INI file, thus it's easily readable and also changeable with most setup creation tools.

Each section holds the settings for the according component or sub-engine NexusDB. Most of these settings comply with the property names of the Delphi components and their possible values can be found in the Reference section of the manual. You do not need to supply all the settings information, as this will be filled (and potentially changed or extended when used with later server versions!) by the server when he first reads the file.

©2004 Nexus Database Systems Pty Ltd.



6.5.1 How are settings stored?

How are settings stored?



It is easier to explain, when we look at an example. Here are the settings of the Event Log:

```
[EventLog]
Settings=1
Setting0=FileName
FileName_Default=
FileName_ValueList=
FileName_PropertyName=FileName
FileName_Hint=
FileName_SettingType=0
FileName_EnforceValues=0
Setting1=MaxSize
MaxSize_Default=50
MaxSize_ValueList=
MaxSize_PropertyName=MaxSize

MaxSize_Hint=
MaxSize_SettingType=1
MaxSize_EnforceValues=0
FileName_Value=
MaxSize_Value=50
```

The first thing to look at is the category name. It has to be the Name of the actual Delphi component in the server, in this case `Event Log`.

Next we have the `Settings=1` line, which indicates that the component has 2 (! it is 0 based, 1 being the highest index) settings.

According to that you will find a `Setting0` and a `Setting1` line, which return the name of the setting.

Now we look at the actual setting itself. We take a closer look at the `FileName` setting. You'll find a number of entries starting with it's name follow by an underscore and a name for each of it's properties. Here's the list of the **currently supported** properties. This list might change at any time as each component (e.g third-party) can (and a few do) save any information it wants to with this schema.

<code>_Value</code>	The actual value of the setting, which will be stored as a string representation of the <code>_SettingType</code> property.
<code>_Default</code>	The default value of the setting, if <code>_Value</code> is not defined
<code>_ValueList</code>	A comma delimited list of possible values for the setting.
<code>_EnforceValues</code>	This works together with <code>_EnforceValues</code> . If set to 1 the user only values of the list should be accepted as actual values.
<code>_Hint</code>	A hint to be shown in a UI.
<code>_SettingType</code>	The type of the setting. Currently there are the following types defined:

```
TnxSettingType =
(
    { string - 0 }
    nxstString,
    { integer -
1 }
    nxstInteger,
    { Boolean -
2 }
    nxstBoolean,
    { Stream - 3 }
    nxstStream,
    { Selection List
- 4 }
    nxstList,
    { Thread
Priority - 5}
    nxstThreadPriority,
    { Radio Button -
6 }
    nxstSingleOption
,
    { Checkboxes - 7
}
    nxstMultiOption
);
;
```

<code>_PropertyName</code>	The name of actual property of the implementing component. This can be used to use reflection to apply the settings.
<code>_Label</code>	the label that the UI should show.

©2004 Nexus Database Systems Pty Ltd.



6.5.2 Example of configuration file

Example of configuration file



Here's an example of a complete server configuration file:

```
[ServerEngine_AliasHandler]

Settings=1
Setting0=Aliases
Aliases_Default=
Aliases_ValueList=
Aliases_PropertyName=Aliases

Aliases_Hint=
Aliases_SettingType=0
Aliases_EnforceValues=0
Aliases_Value=Test=c:
\data;

[nxBlowfishRC4SecuredTranspo
rt1]
Settings=4
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName
=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=
2
EventLogEnabled_EnforceValue
s=0
Setting2=RespondToBroadCasts

RespondToBroadCasts_Default=
false
RespondToBroadCasts_ValueLis
t=
RespondToBroadCasts_Property
Name=RespondToBroadCasts
RespondToBroadCasts_Hint=
RespondToBroadCasts_SettingT
ype=2
RespondToBroadCasts_EnforceV
alues=0
Setting3=Key
Key_Default=
Key_ValueList=
```

```
Key(PropertyName=Key  
Key_Hint=  
Key_SettingType=0  
Key_EnforceValues=0  
EventLogEnabled_Value=False  
  
Active_Value=True  
RespondToBroadcasts_Value=Fa  
lse  
Key_Value=  
  
[RegisteredCOMTransport]  
Settings=3  
Setting0=Active  
Active_Default=false  
Active_ValueList=  
Active_PropertyName=Active  
Active_Hint=  
Active_SettingType=2  
Active_EnforceValues=0  
Setting1=EventLogEnabled  
EventLogEnabled_Default=fals  
e  
EventLogEnabled_ValueList=  
EventLogEnabled_PropertyName  
=EventLogEnabled  
EventLogEnabled_Hint=  
EventLogEnabled_SettingType=  
2  
EventLogEnabled_EnforceValue  
s=0  
Setting2=RespondToBroadCasts  
  
RespondToBroadCasts_Default=  
false  
RespondToBroadCasts_ValueLis  
t=  
RespondToBroadCasts_Property  
Name=RespondToBroadCasts  
RespondToBroadCasts_Hint=  
RespondToBroadCasts_SettingT  
ype=2  
RespondToBroadCasts_EnforceV  
alues=0  
EventLogEnabled_Value=False  
  
Active_Value=True  
RespondToBroadcasts_Value=Tr  
ue  
  
[ServerCommandHandler]  
Settings=2  
Setting0=Active  
Active_Default=false  
Active_ValueList=  
Active_PropertyName=Active
```

```
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=2
EventLogEnabled_EnforceValues=0
EventLogEnabled_Value=False

Active_Value=True

[EventLog]
Settings=1
Setting0=FileName
FileName_Default=
FileName_ValueList=
FileName_PropertyName=FileName
FileName_Hint=
FileName_SettingType=0
FileName_EnforceValues=0
FileName_Value=c:\test.log

[ServerEngine_ServerSettings]
Settings=6
Setting0=BringUpServerOnStart
BringUpServerOnStart_Default=True
BringUpServerOnStart_ValueList=
BringUpServerOnStart_PropertyName=BringUpServerOnStart
BringUpServerOnStart_Hint=
BringUpServerOnStart_SettingType=2
BringUpServerOnStart_EnforceValues=0
Setting1=MinimizeOnStart
MinimizeOnStart_Default=False
MinimizeOnStart_ValueList=
MinimizeOnStart_PropertyName=MinimizeOnStart
MinimizeOnStart_Hint=
MinimizeOnStart_SettingType=2
MinimizeOnStart_EnforceValues=0
```

```
Setting2=ResetsStatsOnActivat
e
ResetsStatsOnActivate_Defaul
t=False
ResetsStatsOnActivate_ValueL
ist=
ResetsStatsOnActivate_Proper
tyName=ResetsStatsOnActivate

ResetsStatsOnActivate_Hint=

ResetsStatsOnActivate_SettingType=2
ResetsStatsOnActivate_Enforc
eValues=0
Setting3=AutoSaveConfig
AutoSaveConfig_Default=True

AutoSaveConfig_ValueList=
AutoSaveConfig_PropertyName=
AutoSaveConfig
AutoSaveConfig_Hint=
AutoSaveConfig_SettingType=2

AutoSaveConfig_EnforceValues
=0
Setting4=CloseServerFromTray
Only
CloseServerFromTrayOnly_Defau
lt=False
CloseServerFromTrayOnly_Valu
eList=
CloseServerFromTrayOnly_Prop
ertyName=CloseServerFromTray
Only
CloseServerFromTrayOnly_Hint
=
CloseServerFromTrayOnly_Sett
ingType=2
CloseServerFromTrayOnly_Enfo
rceValues=0
Setting5=HiddenUIPages
HiddenUIPages_Default=True
HiddenUIPages_ValueList=
HiddenUIPages_PropertyName=H
iddenUIPages
HiddenUIPages_Hint=
HiddenUIPages_SettingType=2

HiddenUIPages_EnforceValues=
0
BringUpServerOnStart_Value=F
alse
MinimizeOnStart_Value=False

ResetsStatsOnActivate_Value=
False
AutoSaveConfig_Value=True
```

```
CloseServerFromTrayOnly_Value
e=False
HiddenUIPages_Value=


[NamedPipeTransport]
Settings=7
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName
=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=
2
EventLogEnabled_EnforceValue
s=0
Setting2=RespondToBroadcasts

RespondToBroadcasts_Default=
false
RespondToBroadcasts_ValueLis
t=
RespondToBroadcasts_Property
Name=RespondToBroadcasts
RespondToBroadcasts_Hint=
RespondToBroadcasts_SettingT
ype=2
RespondToBroadcasts_EnforceV
alues=0
Setting3=CompressLimit
CompressLimit_Default=512
CompressLimit_ValueList=
CompressLimit_PropertyName=C
ompressLimit
CompressLimit_Hint=
CompressLimit_SettingType=1

CompressLimit_EnforceValues=
0
Setting4=Port
Port_Default=16000
Port_ValueList=
Port_PropertyName=Port
Port_Hint=
Port_SettingType=1
Port_EnforceValues=0
Setting5=ConcurrentIOCPThrea
ds
```

```
ConcurrentIOCPThreads_Default=0
ConcurrentIOCPThreads_ValueList=
ConcurrentIOCPThreads_PropertyName=ConcurrentIOCPThreads

ConcurrentIOCPThreads_Hint=

ConcurrentIOCPThreads_SettingType=1
ConcurrentIOCPThreads_EnforceValues=0
Setting6=ServerThreadPriority
ServerThreadPriority_Default=3
ServerThreadPriority_ValueList=
ServerThreadPriority_PropertyName=ServerThreadPriority
ServerThreadPriority_Hint=
ServerThreadPriority_SettingType=5
ServerThreadPriority_EnforceValues=0
EventLogEnabled_Value=False

Active_Value=True
RespondToBroadcasts_Value=True
CompressLimit_Value=512
Port_Value=16000
ConcurrentIOCPThreads_Value=0
ServerThreadPriority_Value=3

[nxNamedPipeTransport1]
Settings=7
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=2
EventLogEnabled_EnforceValue
s=0
```

```
Setting2=RespondToBroadCasts
RespondToBroadCasts_Default=
false
RespondToBroadCasts_ValueList=
RespondToBroadCasts_Property
Name=RespondToBroadCasts
RespondToBroadCasts_Hint=
RespondToBroadCasts_SettingT
ype=2
RespondToBroadCasts_EnforceV
alues=0
Setting3=CompressLimit
CompressLimit_Default=512
CompressLimit_ValueList=
CompressLimit_PropertyName=C
ompressLimit
CompressLimit_Hint=
CompressLimit_SettingType=1

CompressLimit_EnforceValues=
0
Setting4=Port
Port_Default=16000
Port_ValueList=
Port_PropertyName=Port
Port_Hint=
Port_SettingType=1
Port_EnforceValues=0
Setting5=ConcurrentIOCPThrea
ds
ConcurrentIOCPThreads_Default
=0
ConcurrentIOCPThreads_ValueL
ist=
ConcurrentIOCPThreads_Proper
tyName=ConcurrentIOCPThreads

ConcurrentIOCPThreads_Hint=
ConcurrentIOCPThreads_Settin
gType=1
ConcurrentIOCPThreads_Enforc
eValues=0
Setting6=ServerThreadPriorit
y
ServerThreadPriority_Default
=3
ServerThreadPriority_ValueLi
st=
ServerThreadPriority_Property
Name=ServerThreadPriority
ServerThreadPriority_Hint=
ServerThreadPriority_Setting
Type=5
ServerThreadPriority_Enforce
Values=0
```

```
EventLogEnabled_Value=False

Active_Value=True
RespondToBroadcasts_Value=True
CompressLimit_Value=512
Port_Value=17000
ConcurrentIOCPThreads_Value=0
ServerThreadPriority_Value=3

[nxSecuredCommandHandler1]
Settings=2
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=2
EventLogEnabled_EnforceValues=0
EventLogEnabled_Value=False

Active_Value=True

[ServerEngine]
Settings=8
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=2
EventLogEnabled_EnforceValues=0
```

```
Setting2=ServerName
ServerName_Default=NexusDB
ServerName_ValueList=
ServerName_PropertyName=ServerName
ServerName_Hint=
ServerName_SettingType=0
ServerName_EnforceValues=0
Setting3=MaxRam
MaxRam_Default=-1
MaxRam_ValueList=
MaxRam_PropertyName=MaxRam
MaxRam_Hint=
MaxRam_SettingType=1
MaxRam_EnforceValues=0
Setting4=TempStoreSize
TempStoreSize_Default=-1
TempStoreSize_ValueList=
TempStoreSize_PropertyName=TempStoreSize
TempStoreSize_Hint=
TempStoreSize_SettingType=1

TempStoreSize_EnforceValues=0
Setting5=TempStorePath
TempStorePath_Default=
TempStorePath_ValueList=
TempStorePath_PropertyName=TempStorePath
TempStorePath_Hint=
TempStorePath_SettingType=0

TempStorePath_EnforceValues=0
Setting6=JournalEngineClass

JournalEngineClass_Default=
JournalEngineClass_ValueList=
JournalEngineClass_PropertyName=JournalEngineClass
JournalEngineClass_Hint=
JournalEngineClass_SettingType=0
JournalEngineClass_EnforceValues=0
Setting7=Options
Options_Default=
Options_ValueList=ReadOnly,ForceFailSafe,CloseInactiveFolders,CloseInactiveTables,InMemOnly,IsSecure
Options_PropertyName=Options

Options_Hint=
```

```
Options_SettingType=7
Options_EnforceValues=0
EventLogEnabled_Value=False

Active_Value=True
ServerName_Value=NexusDB
MaxRAM_Value=-1
TempStorePath_Value=
TempStoreSize_Value=-1
JournalEngineClass_Value=Tnx
1xRollbackJournalEngine
OptionsCount_Value=0

[ServerInfoPlugin]
Settings=2
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName
=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=
2
EventLogEnabled_EnforceValue
s=0
EventLogEnabled_Value=False

Active_Value=True

[ServerInfoPluginCommandHand
ler]
Settings=2
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName
=EventLogEnabled
EventLogEnabled_Hint=
```

```
EventLogEnabled_SettingType=
2
EventLogEnabled_EnforceValue
s=0
EventLogEnabled_Value=False

Active_Value=True

[SqlEngine]
Settings=2
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName
=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=
2
EventLogEnabled_EnforceValue
s=0
EventLogEnabled_Value=False

Active_Value=True

[nxSqlTriggerMonitor1]
Settings=2
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName
=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=
2
EventLogEnabled_EnforceValue
s=0
EventLogEnabled_Value=False

Active_Value=False
```

```
[TCPIPv4Transport]
Settings=9
Setting0=Active
Active_Default=false
Active_ValueList=
Active_PropertyName=Active
Active_Hint=
Active_SettingType=2
Active_EnforceValues=0
Setting1=EventLogEnabled
EventLogEnabled_Default=false
EventLogEnabled_ValueList=
EventLogEnabled_PropertyName=EventLogEnabled
EventLogEnabled_Hint=
EventLogEnabled_SettingType=2
EventLogEnabled_EnforceValues=0
Setting2=RespondToBroadCasts

RespondToBroadCasts_Default=false
RespondToBroadCasts_ValueList=
RespondToBroadCasts_PropertyName=RespondToBroadCasts
RespondToBroadCasts_Hint=
RespondToBroadCasts_SettingType=2
RespondToBroadCasts_EnforceValues=0
Setting3=CompressLimit
CompressLimit_Default=512
CompressLimit_ValueList=
CompressLimit_PropertyName=CompressLimit
CompressLimit_Hint=
CompressLimit_SettingType=1

CompressLimit_EnforceValues=0
Setting4=Port
Port_Default=16000
Port_ValueList=
Port_PropertyName=Port
Port_Hint=
Port_SettingType=1
Port_EnforceValues=0
Setting5=ConcurrentIOCPThreads
ConcurrentIOCPThreads_Default=0
ConcurrentIOCPThreads_ValueList=
```

```
ConcurrentIOCPThreads_Proper
tyName=ConcurrentIOCPThreads

ConcurrentIOCPThreads_Hint=

ConcurrentIOCPThreads_Setting
gType=1
ConcurrentIOCPThreads_Enforc
eValues=0
Setting6=ServerThreadPriorit
y
ServerThreadPriority_Default
=3
ServerThreadPriority_ValueLi
st=
ServerThreadPriority_Property
Name=ServerThreadPriority
ServerThreadPriority_Hint=
ServerThreadPriority_Setting
Type=5
ServerThreadPriority_Enforce
Values=0
Setting7=ListenThreadPriorit
y
ListenThreadPriority_Default
=3
ListenThreadPriority_ValueLi
st=
ListenThreadPriority_Property
Name=ListenThreadPriority
ListenThreadPriority_Hint=
ListenThreadPriority_Setting
Type=5
ListenThreadPriority_Enforce
Values=0
Setting8=BroadCastThreadPrio
rity
BroadCastThreadPriority_Defau
lt=3
BroadCastThreadPriority_Value
List=
BroadCastThreadPriority_Prope
rtyName=BroadCastThreadPrio
rity
BroadCastThreadPriority_Hint
=
BroadCastThreadPriority_Sett
ingType=5
BroadCastThreadPriority_Enfo
rceValues=0
EventLogEnabled_Value=False

Active_Value=True
RespondToBroadcasts_Value=Tr
ue
CompressLimit_Value=512
Port_Value=16000
ConcurrentIOCPThreads_Value=
0
```

```
ServerThreadPriority_Value=3  
ListenThreadPriority_Value=3  
BroadcastThreadPriority_Value=3  
  
[nxWinsockTransport1]  
Settings=9  
Setting0=Active  
Active_Default=false  
Active_ValueList=  
Active_PropertyName=Active  
Active_Hint=  
Active_SettingType=2  
Active_EnforceValues=0  
Setting1=EventLogEnabled  
EventLogEnabled_Default=false  
EventLogEnabled_ValueList=  
EventLogEnabled_PropertyName=EventLogEnabled  
EventLogEnabled_Hint=  
EventLogEnabled_SettingType=2  
EventLogEnabled_EnforceValues=0  
Setting2=RespondToBroadCasts  
  
RespondToBroadCasts_Default=false  
RespondToBroadCasts_ValueList=  
RespondToBroadCasts_PropertyName=RespondToBroadCasts  
RespondToBroadCasts_Hint=  
RespondToBroadCasts_SettingType=2  
RespondToBroadCasts_EnforceValues=0  
Setting3=CompressLimit  
CompressLimit_Default=512  
CompressLimit_ValueList=  
CompressLimit_PropertyName=CompressLimit  
CompressLimit_Hint=  
CompressLimit_SettingType=1  
  
CompressLimit_EnforceValues=0  
Setting4=Port  
Port_Default=16000  
Port_ValueList=  
Port_PropertyName=Port  
Port_Hint=  
Port_SettingType=1  
Port_EnforceValues=0
```

```
Setting5=ConcurrentIOCPThreads
ds
ConcurrentIOCPThreads_Default=0
ConcurrentIOCPThreads_ValueList=
ConcurrentIOCPThreads_PropertyName=ConcurrentIOCPThreads

ConcurrentIOCPThreads_Hint=

ConcurrentIOCPThreads_SettingType=1
ConcurrentIOCPThreads_EnforceValues=0
Setting6=ServerThreadPriority
y
ServerThreadPriority_Default=3
ServerThreadPriority_ValueList=
ServerThreadPriority_PropertyName=ServerThreadPriority
ServerThreadPriority_Hint=
ServerThreadPriority_SettingType=5
ServerThreadPriority_EnforceValues=0
Setting7=ListenThreadPriority
y
ListenThreadPriority_Default=3
ListenThreadPriority_ValueList=
ListenThreadPriority_PropertyName=ListenThreadPriority
ListenThreadPriority_Hint=
ListenThreadPriority_SettingType=5
ListenThreadPriority_EnforceValues=0
Setting8=BroadCastThreadPriority
BroadCastThreadPriority_Default=3
BroadCastThreadPriority_ValueList=
BroadCastThreadPriority_PropertyName=BroadCastThreadPriority
BroadCastThreadPriority_Hint=
BroadCastThreadPriority_SettingType=5
BroadCastThreadPriority_EnforceValues=0
EventLogEnabled_Value=False

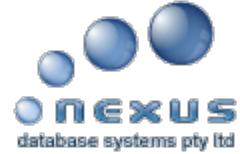
Active_Value=True
```

```
    RespondToBroadcasts_Value=True
    CompressLimit_Value=512
    Port_Value=17000
    ConcurrentIOCPThreads_Value=
    0
    ServerThreadPriority_Value=3
    ListenThreadPriority_Value=3
    BroadcastThreadPriority_Value=3

    [SecurityMonitor]
    Settings=4
    Setting0=Active
    Active_Default=false
    Active_ValueList=
    Active_PropertyName=Active
    Active_Hint=
    Active_SettingType=2
    Active_EnforceValues=0
    Setting1=EventLogEnabled
    EventLogEnabled_Default=false
    EventLogEnabled_ValueList=
    EventLogEnabled_PropertyName
    =EventLogEnabled
    EventLogEnabled_Hint=
    EventLogEnabled_SettingType=
    2
    EventLogEnabled_EnforceValue
    s=0
    Setting2=MaxSessionCount
    MaxSessionCount_Default=-1
    MaxSessionCount_ValueList=
    MaxSessionCount_PropertyName
    =
    MaxSessionCount_Hint=
    MaxSessionCount_SettingType=
    1
    MaxSessionCount_EnforceValue
    s=0
    Setting3=AlwaysLoginAdmins
    AlwaysLoginAdmins_Default=True
    AlwaysLoginAdmins_ValueList=
    AlwaysLoginAdmins_PropertyNa
    me=
    AlwaysLoginAdmins_Hint=
    AlwaysLoginAdmins_SettingTyp
    e=2
    AlwaysLoginAdmins_EnforceVal
    ues=0
    EventLogEnabled_Value=False
```

```
Active_Value=True  
User_Count_Value=0
```

©2004 Nexus Database Systems Pty Ltd.



7 The Enterprise Manager

7.1 Introduction

Introduction



The Enterprise Manager (EM) is the programmer's main tool for table and database maintenance operations.

The EM allows a developer to

- Create tables, indexes, stored procedures, ...
- Perform routine maintenance of NexusDB table structure,
- View and maintain data in tables,
- Maintain and execute SQL scripts,
- Perform live backup and restore operations, and,
- Export and import data in CSV format.

It was completely rewritten for Version 2 and now boast an extensive feature set and an extremely flexible way of presenting data of tables and queries.

The EM is an MDI application allowing multiple table, SQL and restructure views to be opened at the same time. The use of cut and paste operations between all views is fully supported. It also has configurable and dockable toolbars and windows as well as the possibility to register servers for faster access.

This tool is intended for development use only. Due to its ability to allow direct access to the data in NexusDB tables, it is not recommended for distribution to your clients. As the full source for the EM is provided for registered users, it is straightforward for developers to implement their own application specific table utilities.

Due to the use of several third party component sets, it is not possible to recompile the EM out of the box anymore. You need to get your own copy of these third party tools to be able to change or recompile it. A full list of the used tools and the required version numbers are available on request. Just send an email to our support.

Before we investigate connecting to servers, and other actions, we shall first discuss the Global Options settings.

©2004 Nexus Database Systems Pty Ltd.

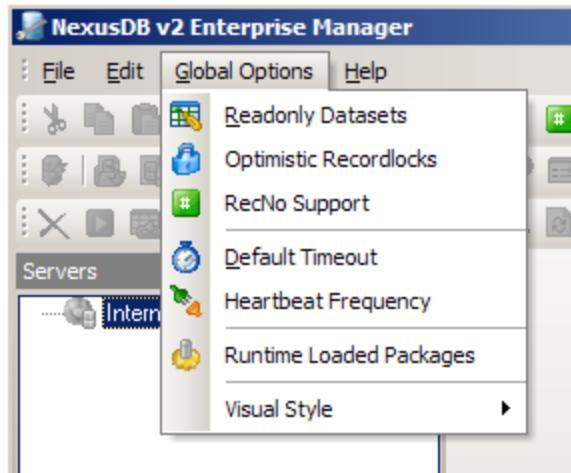


7.2 Global Options

Global Options



There are some global options that can be set in the EM. These options are available via "Global Options" from the main menu when you first boot-up as shown in the figure below.



Most of these options are also accessible through and their current state visible in the global options toolbar. The icons in the toolbar are the same as the ones used in the menu.



Readonly Datasets

This option toggles Read-only mode for all SQL and table browser windows.

NOTE: To protect against inadvertently modifying data, by default (when the EM is first used) this option is on. If you want to modify data whilst in the EM, then you will need to set this option off.

Optimistic Locks

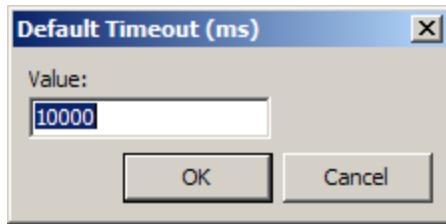
Determines whether table browser windows use optimistic or pessimistic locking when performing edits.

RecNo Support

RecNo support is a special feature to accommodate correct positioning with the thumbs slider on grids. This option needs extra calculations and work on the server side and thus can have a huge impact on the performance. Use it carefully if necessary, and especially avoid its use on large tables.

Default Timeout

The default timeout value is set to 10 seconds as shown in the screen shot below. This value is used as the default when table browsers are opened and for SQL queries.



Note that the value is presented in milliseconds (there are 1,000 milliseconds per second). If you set this value to 0 milliseconds, then the timeout is ignored (has, to all intent purposes, an infinite value). This can be handy when you are testing a particularly complex query which may take many minutes to evaluate however, generally speaking, you would want the timeout to be a finite value to guard against deadlock situations.

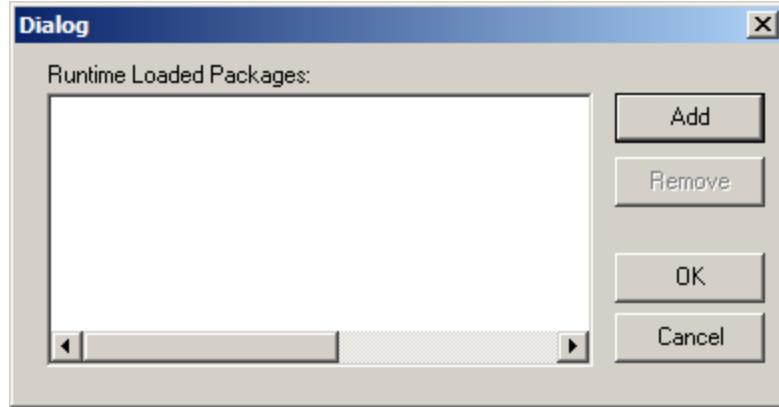
If you set this value to -1, then the value used will be the value of the next highest level component in the connection chain to the currently selected server. For the internal server, that is also 10 seconds. For external servers the value may be different.

Heartbeat Frequency (ms)

How often EM sends heartbeat messages to the connected server(s) to inform them that its still alive. If the connection is lost for the duration of 2 heartbeats, the server closes the connection and frees up server-side resources.

Runtime Loaded Packages

This new feature of EM enables you to load and unload extension packages for EM. An extension package can add new menu items and functionality to EM and is mainly targeted at third-party developers to directly integrate into EM.



Pressing the `Add` button opens a file selector which prompts the user to select a package that he wants to load into EM. The `Remove` button unloads the selected package. If you press `Cancel` the dialog will be closed without actually performing the loading/unloading of packages, `OK` commits the changes and adds/removes the packages chosen.

Visual Style

This option is a purely for the user interface. It has no impact on the functionality or performs any functionality on data. You can select between `OS Default` (default style used by the operating system), `Standard` (the normal Windows 2000 and lower look), `.NET` (the new .NET tools look), `XP` (Windows XP look) and `Office 11` (the look of MS Office 11/2003).

©2004 Nexus Database Systems Pty Ltd.



7.3 Using the Servers Window

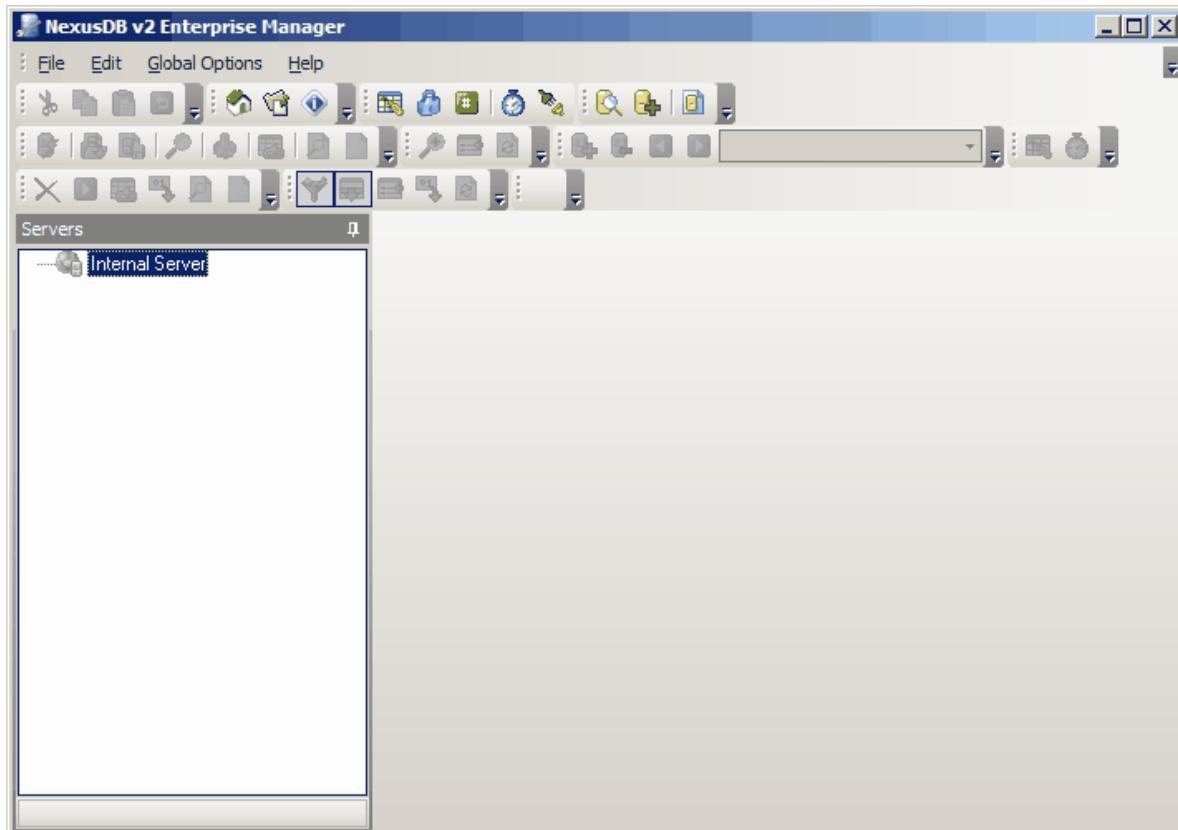
Using the Servers Window



When started, the EM will come up with the Servers window already open as shown below. The Server window is the central place to connect to servers, create and manage databases and even tables. It is by default docked to the left side of the main window and is the window where most tasks can be initiated or performed.

Internal Server

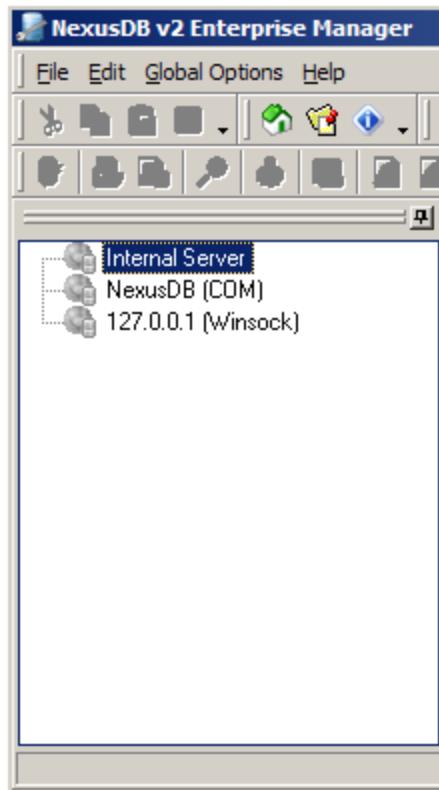
When the EM is booted up and there are no NexusDB Servers broadcasting (or registered) the screen below is displayed.



The server displayed is the embedded server that is compiled into the EM. Being grayed indicates that the server is currently inactive. To activate the server, just double-click on the icon or server name (in this case, Internal Server) label itself, or press Enter when the server is the active selection.

Remote Servers

The EM scans for servers using broadcast and lists all servers found, on all available transports. Found servers are shown either by their name, their address or a combination of both. In any case (except of the Internal Server) the transport type is added in brackets after the name.



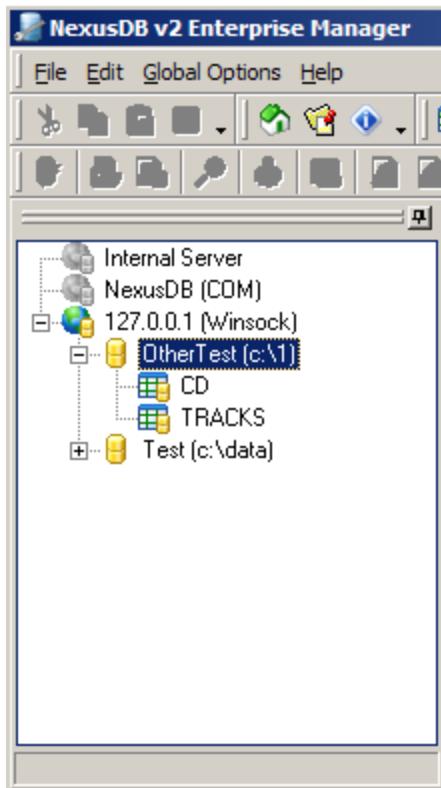
The screen shot above shows a typical screen on startup of the EM in which a server has been found at 127.0.0.1 using Winsock transport and another or the same server named NexusDB using the COM transport. All the servers are grayed because no connection is yet established to any server.

Activating a Server

To connect to one of the servers simply double click the name or icon, or use the right mouse button to access the context menu and select `Attach`. The following screen shot shows an active connection to the local (127.0.0.1) Winsock Server. For each active server the server window displays a hierarchy of objects available on this server with the following structure:

Servername
Alias
Table

As you can see in the example below, the local (127.0.0.1) server has 2 aliases (`Test` and `OtherTest`). One of those aliases (`OtherTest`) has been expanded to show the tables in that alias.



Each of the hierarchy levels (Servers, Aliases, Tables) have their own context menus, which we will enable you to perform global tasks or actions on the selected object. We have a look them now.

©2004 Nexus Database Systems Pty Ltd.

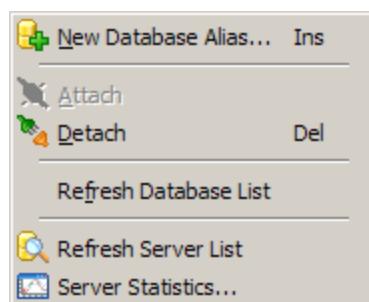


7.3.1 Server Context Menu

Server Context Menu

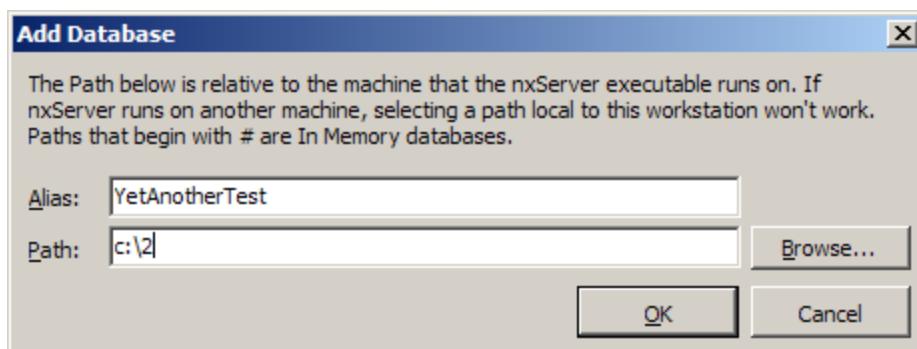


Once the focus is set on a particular server in the main window area, a context menu to manipulate that particular server is accessible by pressing the right mouse button. The items available in this popup menu are shown below. This menu set allows you to manipulate database aliases and work on the currently active server.



New Database Alias

To create a new alias on the active server in the list, enter an alias name and a directory path in the dialog as shown in the next figure. Keep in mind that the directory path is always relative to the machine the NexusDB Server is running on (which could be another machine). If the path is not on the local machine, then the EM can't create it for you. As you can see on the screen shot you can also create InMemory only databases. These are volatile and once the server is disabled, closed or restarted all tables and their data are discarded.



Attach

Use this command to open the connection to a server (if you right-click to open the pop-up menu, the EM will automatically try to establish a connection). If the connection is made active, the server icon in the tree will display a green "link active" line on its right-hand side. If no connection exists, the server icon shows a red "x" at the right-hand side.

Detach

Use this command to close a connection to a specific server.

Refresh Database List

If the alias list is changed outside of the EM (that is, directly on a NexusDB Server), this option will refresh the list here.

Refresh Server List (F5)

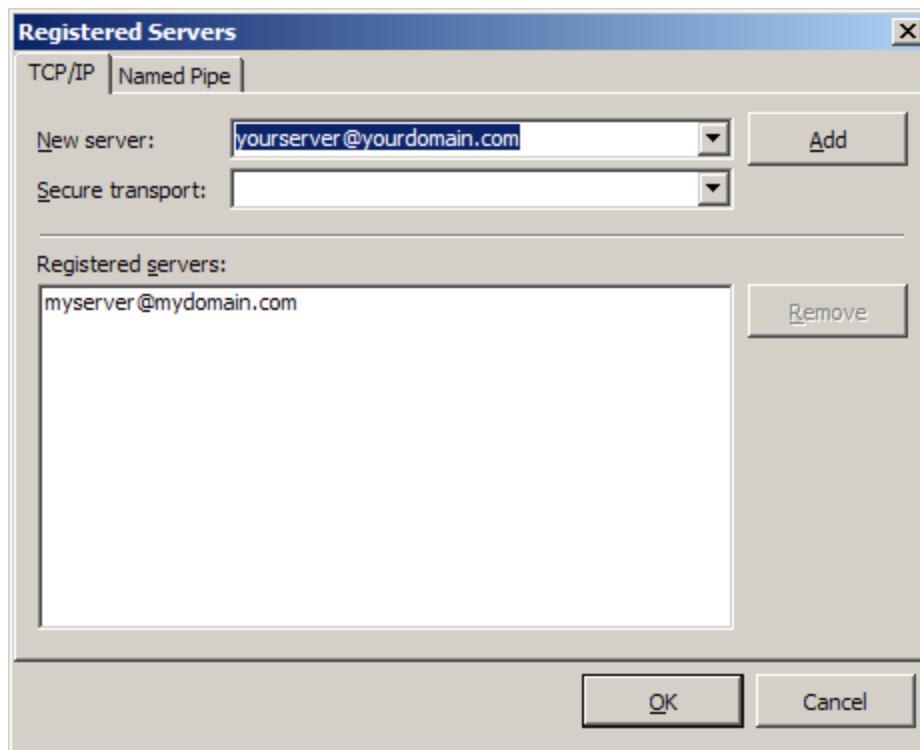
If you start up an external NexusDB Server, this will refresh the list to make it appear. Similarly with closing down servers and refreshing aliases in external servers.

Register Server

Opens a dialog where you can add servers manually. This must be done if the server can't be reached by the normal broadcast method (for instance, if the server is on a different subnet, or only reachable via the Internet).

Use this option to register a server with the EM that can not be found by broadcast, for instance if want to access a server over the internet or simply different sub networks. Enter the details required and click the Add button.

To remove a registered server you need only select it and then click the Remove button.



To use a BlowFish secured server, enter the server name into the New Server box, and select the secure transport in the combo box below it. Press Add, and you have created a securely wrapped transport setup.

When you exit the dialog by pressing OK, any existing connections are closed, and the tree of servers in the Servers window is regenerated.

Server Statistics ...

??? [todo]

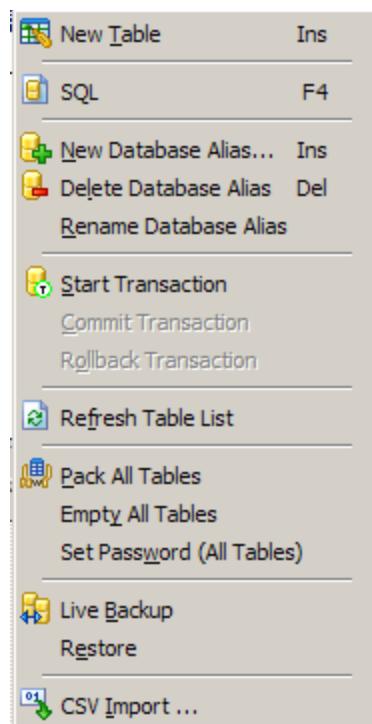


7.3.2 The Database Context Menu

The Database Context Menu



On setting the focus to a particular alias in the currently selected Server, a further context menu to manipulate that alias and tables therein is available for use. The items in this popup menu are shown in the following screen shot. This menu set allows you to manipulate the selected database alias and to work on the tables within that alias.



New Table (Ins)

Open the Create Table Dialog dialog. See the according chapter for details about creating a new table.

SQL (F4)

Opens a SQL window for querying on the active database. See more info on the query window in the chapter The SQL Window.

New Database Alias

Create a new alias on the server this database belongs to. This is the same dialog linked to the similar menu option for the popup menu associated with servers.

Delete Database Alias

Remove an alias. The command only removes the alias from the server's list of aliases, **no tables or directories are physically deleted.**

Rename Database Alias

Give the alias another name.

Start Transaction

Starts a transaction on the database. The database node in the tree will get a circled "T" marker to indicate that a transaction is in progress: All normal table- and SQL operations will now operate in the context of this transaction. **Caution:** keep in mind that using transactions on a database where other people are working, will lock others out from doing their work. Use with extreme caution in production environments!

Commit Transaction

Commits all edits done since the Start transaction command.

Rollback Transaction

Rolls back all edits done since the Start transaction command

Refresh Table List

Update the list of tables owned by the database, in case something external to EM changes the list.

Pack All Tables

Compact and reindex all tables in the current database.

Empty All Tables

Deletes all data from all tables. Be careful when using it, as this is not undo-able (except you're working with transactions).

Set Password (All Tables)

Allows you to give all tables in a database the same password. Note carefully: this password feature **only** works within the EM. It is only intended to stop end-users who have the EM executable from using it to view the content of your tables. Keep in mind that any person with a NexusDB license can simply disable the password checking in the EM source, and view your tables.

Live Backup

Run a backup from one server/database to another, even while the database is in use. See the TnxBackupController ??? for more info on how the backup operates.

Restore

Copy back tables that was previously backed up. This command also works if you just want to copy tables from one server/database to another.

CSV Import

Import CSV data into a new NexusDB table. Supports most variants of CSV and date formats. See section The CSV Import Wizard for detailed info on the import wizard.

©2004 Nexus Database Systems Pty Ltd.

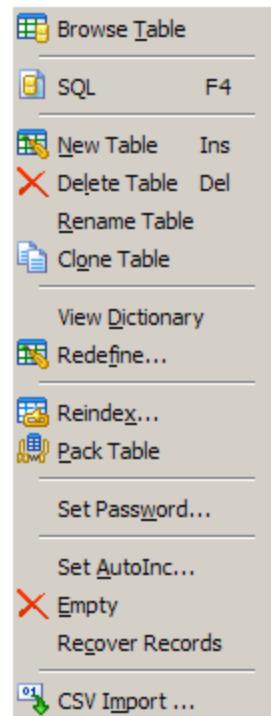


7.3.3 The Table Context Menu

The Table Context Menu



Drilling down even further in our focus, we can now concentrate on individual tables within an alias on a particular Server. Setting the focus to a particular table in the currently active alias (in the selected Server), takes us to a further context menu to manipulate that table. The items in this menu are shown in the picture below.



Browse Table

Open the Table browser window. See section The Table Browser Window for details about this window.

SQL (F4)

Opens a SQL window for querying on the active database. See more info on the query window in section The SQL Window.

New Table (Ins)

Open the Create table window. See section Managing and viewing table structures for details about this window.

Delete Table (Del)

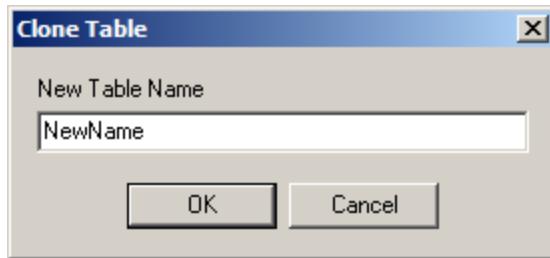
Deletes table permanently from disk. You need exclusive access to the table to the table for performing this task.

Rename Table

Changes the name of a table. You need exclusive access to the table to the table for performing this task.

Clone Table

Creates a new table, with the exact same structure as the selected one. On selecting this menu you will be shown the following dialog. Just enter the name for the new table and press OK.



View Dictionary

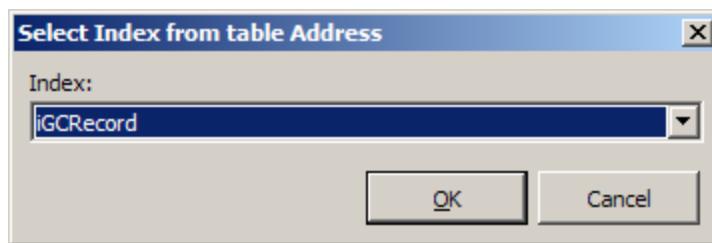
Open the Create/Restructure window in read-only mode, with the Fields tab active. See section Managing and viewing table structures for details about this window.

Redefine

Opens the Create/Restructure window with the current contents of the table. See section Managing and viewing table structures for details about this window.

Reindex

Choose an index in the dialog that you want to reindex,



and press OK. If you want to reindex all Indices at once it's better to use the Pack Table command. You need exclusive access to the table to perform this task.

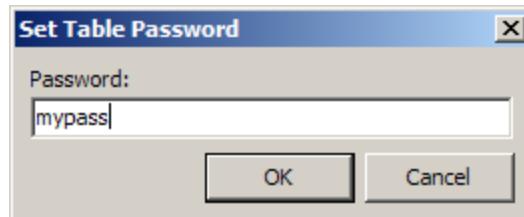
Pack Table

Compact and reindex all Indices. This is achieved by creating a clone of the current table and batch copying all records. Although other databases need to regularly perform a pack table, NexusDB is a maintenance free database. You usually don't need to pack a table, as NexusDB reuses space of deleted records. If for some reason the line got pulled on a server and a table doesn't seem to be right, your first action should be to try to pack the table, as in most cases this will fix the problem. Exclusive access to the table is needed to perform this task.

If that does not help please take a look at the Recover Records command below.

Set Password

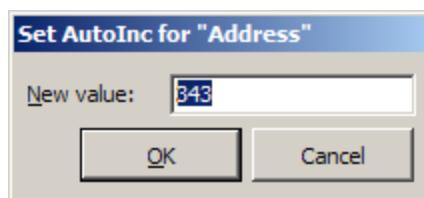
Allows you to give the table a password. **Note carefully:** this password feature only works within the EM. It is only intended to stop end-users who have the EM executable from using it to view the content of your tables.



Keep in mind that any person with a NexusDB license can simply disable the password checking in the EM source, and view your tables.

Set AutoInc

Allows you to change the last used autoinc value for a table.



The next insert operation will generate the value (newvalue+1) in the autoinc field.

Empty

Delete all the records in the table. This command is not reversible and you need exclusive access to the table.

Recover Records

If you have a table whose internal structure has been damaged, this function will attempt to extract all recoverable data into a new "recovered" table. Records that were not fully recoverable are copied into a "failed" table. Again, you need exclusive access to the table.

CSV Import

Import CSV data into a new NexusDB table. Supports most variants of CSV and date formats. See section The CSV Import Wizard for detailed info on the import wizard.

©2004 Nexus Database Systems Pty Ltd.



7.4 Creating and Restructuring Tables

Creating and Restructuring Tables



To create new Tables, please first activate the server in the servers window. Select the database you want the table to create in and bring up the database context menu by pressing the right mouse button. Select the `New Table` menu item and the Table Structure Window will open.

If you want to restructure a table bring up the Table Context Menu in a similar way to above and select `Redefine` and it again brings up the table structure window, this time with the current table structure preset.

Before we describe the various sections of this window we take a short look on regular identifiers and NexusDB field types.

©2004 Nexus Database Systems Pty Ltd.



7.4.1 Regular Identifiers

Regular Identifiers



NexusDB V2 only allows code page neutral ANSI characters in identifiers (table names, field names, ...), that is characters having the same ordinal value regardless of a code page. The following table shows the valid characters that can be used in regular identifiers:

Character	Code	Symbol
-----------	------	--------

Exclamation mark	#33	!
Pound sign	#35	#
Dollar sign	#36	\$
Percent	#37	%
Ampersand	#38	&
Left parenthesis	#40	(
Right parenthesis	#41)
Plus sign	#43	+
Minus sign	#45	-
Digit	#48..#57	0..9
At sign	#64	@
Upper case letter	#65..#90	A..Z
Left bracket	#91	[
Right bracket	#93]
Circumflex	#94	^
Underscore	#95	_
Lower case letter	#97..#122	a..z
Left brace	#123	{
Right brace	#125	}
Tilde	#126	~

- A regular identifier is an unquoted string of characters.
- Regular identifiers cannot contain spaces.
- Regular identifiers cannot begin with a decimal digit.
- Regular identifiers are case-insensitive.
- The maximum length of regular identifiers is 128 characters.

NOTE: There might be some issues for a few customers with converting tables that do not comply with this rules.



7.4.2 Field Types

Field Types



Nexus supports the following field types:

Type	Capacity
Autolnc	32 bit unsigned integer
BCD	Precision 20, scale 4
BLOB	4 GB
BLOB Graphic	4 GB
BLOB Memo	4 GB
Boolean	8 bit boolean flag
Byte	8 bit unsigned integer
Byte Array	64 KB
Char	8 bit character
Currency	Precision 20, scale 4
Date	4 bytes
DateTime	8 bytes
Double	5.0E-324 .. 1.7E308
Extended	3.6E-4951 .. 1.1E4932
GUID	16 bytes
Int8	8 bit signed integer
Int16	16 bit signed integer
Int32	32 bit signed integer
Int64	64 bit signed integer
NullString	8192 characters
RevRev	32 bit unsigned integer
ShortString	255
Single	1.5E-45 .. 3.4E38
Time	4 bytes
WideChar	16 bit character
WideString	32767 characters

Word16	16 bit unsigned integer
Word32	32 bit unsigned integer

©2004 Nexus Database Systems Pty Ltd.



7.4.3 Table Structure Window

Table Structure Window



Use the Table Restructure window to view and edit the table dictionary. If you select the View Fields or View Indices options, this screen is in read-only mode. The table can only be restructured by the **Redefine** menu option.

The NexusDB Data Dictionary stores all meta information about tables. For each information EM presents the user with a separate category. If you're using the EM shipped with NexusDB the following categories are available.

- Main Table Name
 - ... Locale Descriptor
 - ... File Descriptors
 - ... Encryption Engine
 - ... Autolinc Engine
 - ... Field Descriptors
 - ... Index Descriptors
 - ... Record Engine
 - ... Child Tables

Let's take a closer look at each of them. First we see is the Table name edit.

 A screenshot of a software interface showing a single-line text input field. The placeholder text "Table Name:" is visible to the left of the cursor. The word "NewTable" is typed into the field.

While in NexusDB V1 Tablename could be any valid windows filename and it was limited only by the OS in size and characters, NexusDB V2 restricts the filename to be in sync with the SQL:2003 standard's Regular Identifier definition. This is necessary to ensure full compatibility with this standard.

©2004 Nexus Database Systems Pty Ltd.



7.4.3.1 Locale Descriptor

Locale Descriptor



The Locale descriptor for a table is telling NexusDB how the (string) data in a table should be handled.

Locale:
LOCALE_USER_DEFAULT (1024)

Ignore KanaType
 Ignore NonSpace
 Ignore Symbols
 Ignore Width
 Use String Sort

Storage Codepage:
0 - Default

Add Descriptor Remove Descriptor

Use the options to set the Windows OS collation for character and national character types. If you leave the options empty, then the plain byte order is used in sort and comparison operations. Please refer to the Windows manual for further information on collation, locales and code pages and see also the Win32API help file in Delphi – topic : Compare String.

©2004 Nexus Database Systems Pty Ltd.



7.4.3.2 File Descriptors

File Descriptors



NexusDB allows the developer to split the data of tables into different physical files. Each of these files is defined by a File Descriptor.

File Descriptor:
File 2 New Remove

File Extension: NX1 Description: Storage for Blobs

Block Size: 4kb Initial Blocks: 4 1

You will always find a File1 descriptor in this dialog, which is the main file. It can't be deleted and always uses the default file extension (the according field is empty), which is set by the database developer in the source codes.

File Extension

For every file except File1 (the main file) you can set your own file extension. Take care to not give any two files the same extension as this will lead to a "Can't open tablename" error when trying to create the table. **Please note** that in the shipped version the default file extension is NX1 and thus the example above would trigger exactly this error.

Block Size

Blocksize is restricted to values of 4, 8, 16, 32 and 64 kilobytes. The server engine will ensure that each block can hold at least one record and conforms to index block page requirements as well. As a heuristic you should bear in mind that larger block sizes take longer to read/write to disk.

Initial Blocks

Initial Blocks is the number of blocks the table is initially created with. The default for Initial Size is 4, which is the minimum number of blocks NexusDB needs to totally define a table.

Grow Blocks

Grow Blocks is the number of blocks it grows by on disk when more space is needed in the file. To reduce disk fragmentation you should use a large value for Grow Size, especially if you have a table with a large number of indices and/or is being continuously appended to. The default is nonetheless set to 1 to ensure minimum file size.

©2004 Nexus Database Systems Pty Ltd.



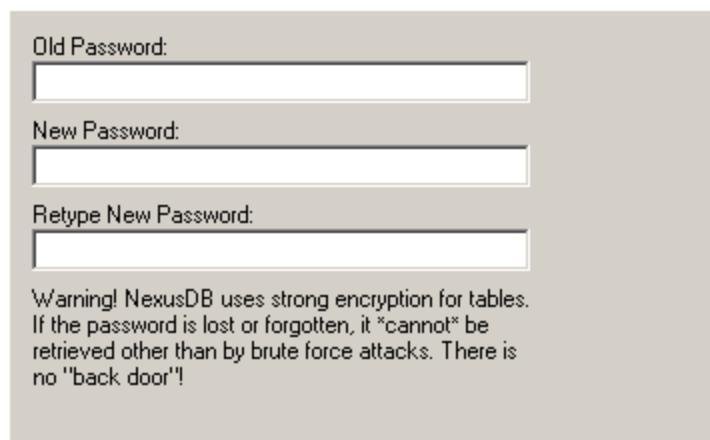
7.4.3.3 Encryption Engine

Encryption Engine



By default NexusDB V2 does not encrypt tables and the Encryption engine combo box will thus be empty.





Selecting the nx1xDefault encryption engine instructs the server engine to encrypt the table on disk. Tables with Encrypted checked instructs the server engine to encrypt each block as it is written to disk. Blocks read will also be decrypted. The encryption is controlled server side.

Unlike NexusDB V1 which was a very simple protection schema, NexusDB V2 uses strong encryption for tables. This means you have to provide a password for encrypting the table and re-enter it when you want to access a table. Due to the nature of strong encryptions, a password can **not** be retrieved again and brute force would be the only way to crack this schema. Take care to remember the password used, otherwise you can not get access to your data anymore.

Leaving the combobox empty means the table will not be encrypted.

©2004 Nexus Database Systems Pty Ltd.

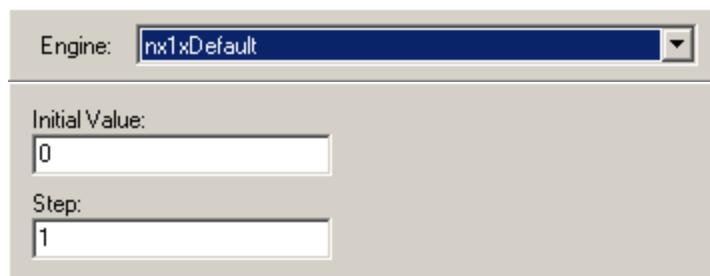


7.4.3.4 Autolnc Engine

Autolnc Engine



NexusDB V2 has the option to select a specific Autolnc engine which internally handles the assignment of unique values.



For the developer version, there's only 1 engine included, nx1xDefault, which is a simple Autolnc generator that starts with an Initial Value (defaults to 0) and increases it by Step 1 (default) for each record inserted. It does not reuse values of deleted records.

The Enterprise Version of NexusDB V2 will add an alternative generator engine, that is also capable of reusing values of deleted records.

If the combo box is left empty, the nx1xDefault engine is implicitly used with a and Initial Value of 0 and a Step 1.

©2004 Nexus Database Systems Pty Ltd.



7.4.3.5 Field Descriptors

Field Descriptors



Name	Type	Units	Decimals	Required
Index	AutoInc	10	0	<input type="checkbox"/>
Name	WideString	200	0	<input type="checkbox"/>
Country	WideString	50	0	<input checked="" type="checkbox"/>

Below the table is a horizontal toolbar with buttons: Insert, Append, Delete, Move Up, Move Down, and Clear. To the right of the table is a detailed view of the selected "Country" field's descriptor settings:

Default Value Descriptor	
Default Descriptor	TnxConstDefaultValueDescriptor
Default Value	Australia
ApplyAt	Both
ApplyOnInsert	<input checked="" type="checkbox"/>
ApplyOnUpdate	<input type="checkbox"/>
OverWriteNotNull	<input type="checkbox"/>

Name

A field Name is required. Please note that the field name has to comply with the SQL:2003 regular identifier restrictions.

Type

A field Type is required. Take a look at the Field Types section for a list of NexusDB types and their capacity.

Units

For most types this is implicit and can't be set, but some types (strings, byte array) also require a value for size which is entered in the Units column. Again please take a look at the Field Types section if you're not sure about the capacity of a certain type.

Decimals

The Decimals column can be used for floating point fields. It is not, however, mandatory for these fields.

Required

Check the Required tick box if the field must have a non-null value.

Map From Field

Map from Field is only visible when you restructure tables. It displays the field map (on a field by field basis as shown) used when the table is restructured.

Data Loss Action

Like Map From Field this column is only visible if you're restructuring a table. While trying to restructure a table, it might be possible that some field values can't be mapped without a loss of data. By setting this option you can tell NexusDB what to do if this happens. The possible actions are **dlaFail** (the restructure is stopped and the original state is restored), **dlaNULL** (set the field where the data loss occurred to NULL) and **dlaBestFit** (copy the best fitting value that NexusDB can determine e.g. truncate a string field to fit a new column length).

©2004 Nexus Database Systems Pty Ltd.



7.4.3.5.1 Default Values

Default Values



In NexusDB V2 we've introduced a much more flexible way of defining/setting a default value for a field. Actually we've gone even a bit further and allow to also apply Default values when a record is updated. This can be very handy for example to enforce a certain value if the column is set to NULL by a user.

Default Descriptor

Again, NexusDB offers different types of default descriptors. The currently available ones are

- **TnxAutoGuidDefaultValueDescriptor**,
- **TnxCurrentDateTimeDefaultValueDescriptor**,
- **TnxEmptyDefaultValueDescriptor** and
- **TnxConstDefaultValueDescriptor**.

The first two are special generators: TnxAutoGuidDefaultValueDescriptor creates a new unique GUID every time it is applied. TnxCurrentDateTimeDefaultValueDescriptor sets the field to the current date and time, while TnxEmptyDefaultValueDescriptor always sets the field to NULL.

TnxConstDefaultValueDescriptor allows the user to specify a certain value as Default Value that will be applied.

Default Value

You can only specify a value in the **Default Value** row for the TnxConstDefaultValueDescriptor (or better, the others ignore a value). Please make sure that the value complies with the data type chosen for the field and does not violate constraints set elsewhere (e.g. Referential Integrity).

Apply At

Due to the nature of NexusDB being a real Client/Server system, you've the option to apply the default setting on the **Server**, **Client** or on **Both**. For best consistency it is advisable to have always Both selected here.

ApplyOnInsert

Checking this option makes sure that NexusDB applies the default for every newly inserted record.

ApplyOnUpdate & OverwriteNonNULL

You can force NexusDB to apply the default value when a record is updated too. This can e.g. be handy to enforce a certain value in a column, instead of NULL. Set OverWriteNonNULL if you want to **always** apply the default value.

©2004 Nexus Database Systems Pty Ltd.



7.4.3.6 Index Descriptors

Index Descriptors



The index descriptors section is where maintenance of the indices is performed.

Main				
Name	Unique	File	Description	Key Length
Index	<input checked="" type="checkbox"/>	File 1	Primary Index	5
▶ Name	<input type="checkbox"/>	File 1	Lookup Value	403
Sequential Access Index	<input checked="" type="checkbox"/>	File 1	Sequential Access Index	8

Main		Locale Settings		
Field Name	Ascending	Case Insensitive	Locale	
Key Descriptor	Override Char C	Override Byte Count	Null Behaviour	Ignore Kanatype Ignore Nonspace Ignore Sym Ignore W String
I Name	<input checked="" type="checkbox"/>	<input type="checkbox"/>	LOCALE_SYSTEM_DEFAULT (2048)	
Localized Text Sort	200	402	nbTop	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

Main				
Main		Locale Settings		
Field Name	Ascending	Case Insensitive	Locale	
Key Descriptor	Override Char C	Override Byte Count	Null Behaviour	Ignore Kanatype Ignore Nonspace Ignore Sym Ignore W String
I Name	<input checked="" type="checkbox"/>	<input type="checkbox"/>	LOCALE_SYSTEM_DEFAULT (2048)	
Localized Text Sort	200	402	nbTop	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

The Indices tab allows you to set and define indices for your table from the fields entered. This consists of 2 parts for each index. The index header information is entered into the top grid on this tab. The fields for each index are entered in the lower grid. Note that you must enter at least one field for each index as you proceed. (That is, you can not simply enter all the indices and then come back and add the fields for each index as a 'second pass'.)

Index Header

Each index must be given a **name** (again, refer to the Regular Identifiers section for compliance) as a minimum entry in the top grid. If all keys in this index are unique, then check the Unique column.

To save the index in secondary **file** select it from the drop down (a file has to be defined in the File Descriptors part first). The default selection is File1 which means the main table file.

The **Description** column is an optional place for you to enter whatever text you would normally like to put comments related to each index. This is a virtually size-unlimited field.

Index Fields

The fields in each index contain the following values:

A valid (existing) **field name** must be entered. This is selectable from a dropdown downlist.

NexusDB allows you to set a field based sort order. The default is **Ascending**, but you are not restricted to having all fields in an index as either ascending or descending.

Check the **Case Insensitivity** field if you don't want case sensitivity in your sorting.

The value entered into the **Key Descriptor** describes which sorting engine you will be using for this field. The default is `Byte Compare`, but you can also select `Simple Text Sort` (which is basically just a byte by byte comparison) and `Localized Text Sort`. The last option needs the additional selection of **Locale Settings** and it's sub options to define which collation is used for sorting. Please refer to the Windows manual for further information on collation, locales and code pages and see also the Win32API help file in Delphi – topic : `Compare String`.

Other options available are **Override Char** (??? [what is this]), **Override Byte** (??? [what is this]) and finally **Null Behaviour**. The latter defines where in an index records with a NULL value in the field are sorted in. The available selections are `nbTop`, `nbBottom`, `nbAsAscend`, `nbAsDescend`, `nbFilterNull` and `nbFilterNonNull` ??? [need more info on these].

©2004 Nexus Database Systems Pty Ltd.



7.4.3.7 Record Engine

Record Engine



NexusDB V2 comes with a number of Record Engines, that store data in a slightly different way each.



In the current version you have three engines to choose from:

nx1xDefault

Selecting this option will use the standard option as the developer of an application has defined it. It can either be the static or the variable one. The actual selection is done at compile time of the application used.

Static

This is an optimized record engine that stores fields in the static original length. It is the fastest engine.

Variable

The variable record engine is storing fields with variable length. It cuts off the fields at their actual length and thus for example saves in a country field with length 50 for a value "Australia" only 9 characters instead of 50. Technically it needs a bit more space but I think it illustrates its purpose. This engine will keep the file size to a minimum as more records can be saved in each block.

If you leave this combo box empty, the nx1xDefault engine will be used implicitly.

©2004 Nexus Database Systems Pty Ltd.



7.5 Browsing Tables and Queries

Browsing Tables and Queries



NexusDB knows two different concepts to access data. The first is the direct cursor access method, for most Delphi user simply Table or TTable access. The other one is using SQL to define and access data. EM can handle both types and for this purpose has two different windows to work with: the table browser and the SQL window.

Both allow detailed browsing and viewing of the data in certain tables, and share some functionality in the data grid that is used to present the data. Let's first take a look at its options.

©2004 Nexus Database Systems Pty Ltd.



7.5.1 Using the Data Grid

Using the Data Grid



The data grid displays all fields in table or query result set, with default formatting. For tables and live queries the data is directly editable, unless the main menu Live Datasets option is off (it is off by default).

Here are the main features of the grid:

Sorting Data

Click one of the column headers to sort all your data by this column.

DiskSerialNumber	Artist	Title
11626136	Abba	Abba

Be aware that this can take a long time on large tables, depending if you a suitable index is available or not.

Grouping Data

Drag & drop a column header to the dark gray area just above them to group by this field. You can also group by more than one field by dropping different fields.

The screenshot shows a table browser interface with a header row containing columns for 'Artist', 'Title', 'DiskSerialNumber', 'Title', and 'AlbumCover'. Below the header is a message 'Click here to add a new'. A list of artists is displayed in a tree view: '+ Artist : Abba' (selected), '+ Artist : Accept', '+ Artist : Aerosmith', and '+ Artist : Aha'. Under 'Artist : Abba', there are three rows of data: '13065458 Balls to the wall', '12664006 Breaker', and '12081530 Restless and wild'.

Again this can be time consuming for large tables.

Rearrange Columns

The columns of the grid can be rearranged by dragging the header of a column along the header row.

The screenshot shows the same table browser interface as above, but with the 'Artist' column header being dragged from its original position to the right of the 'DiskSerialNumber' column header. The header row now shows 'DiskSerialNumber', 'Artist', 'AlbumCover', 'Title', 'Artist', and 'new row'.

Resize Column Widths

You can resize the width of each individual column by dragging the separator line at its end between columns.

The screenshot shows the table browser with the 'Artist' column width being resized. The 'Artist' column header is highlighted, and its rightmost separator line is being dragged to the left, increasing the width of the adjacent column.

©2004 Nexus Database Systems Pty Ltd.



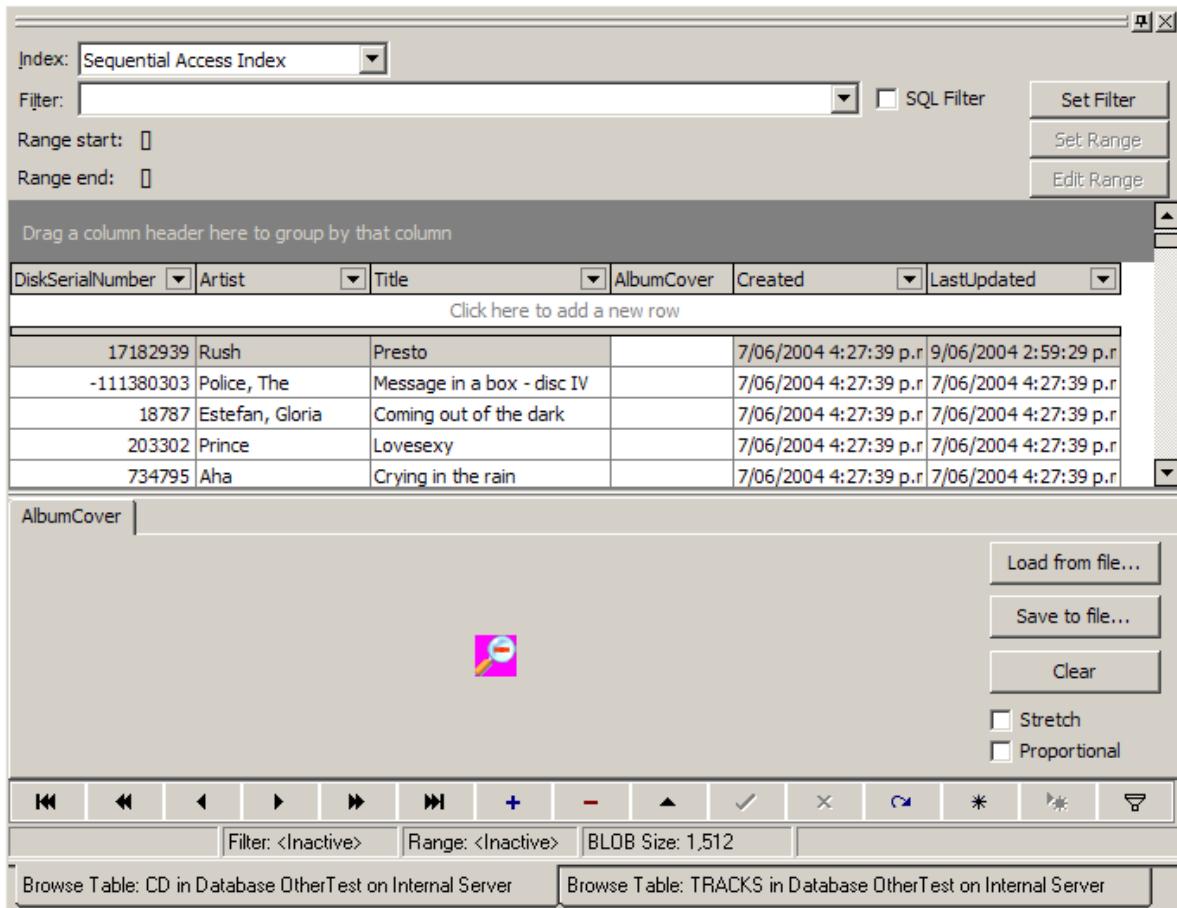
7.5.2 Browsing a Table

Browsing a Table



When you want to view or edit the data in your tables, or check results of filters, ranges etc, the Table Browser window will do this and more.

The first thing to note is that the table browser is a dockable window and can be floated if you like so. By default the window is docked to the full client area of the main window and each open table will be tabbed on the bottom.



Data Grid

For finding out about how to manage the data in the data grid please refer to the Data Grid section.

Indices

By selecting an Index from the list, you define which Index is used for the viewing order and to which index a range is applied to. To use the Find Nearest and Set Range functions, which both require a searchable index, select an index other than the Sequential Access Index.

Find Nearest

This option is only visible if you have an Index different from the Sequential Access Index selected. Enter the value you want to look for, press the Find button and the data grid will be positioned on the first record matching the value.

Index:	ArtistIdx	Find nearest:	Acc	Find
--------	-----------	---------------	-----	------

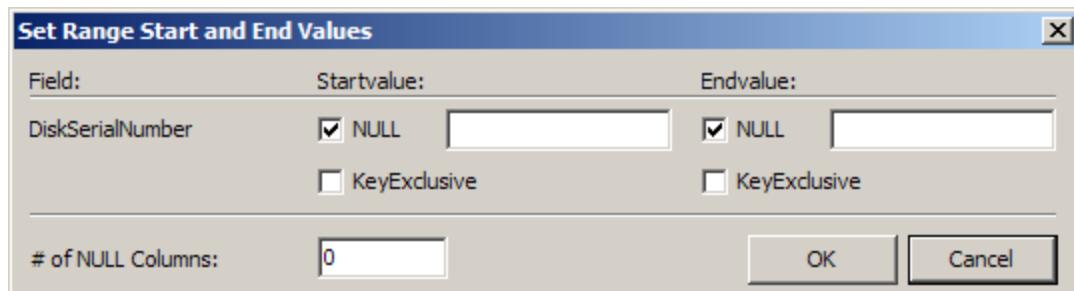
The search is performed by looking up the data in the index and thus it will only find index keys matching the value.

Ranges

Ranges allow you to very quickly restrict the data grid view to a subset of records only by specifying a lower and upper indexed value.

Use the Set Range/Clear Range value to enable a Range and Edit Range to change a previously set range.

The Set Range dialog that is popped up, is dynamically configured to have one line of settings for each field in an index. Set start and end values for each field to be ranged, and press OK. The fields and options correspond to the SetRange() and KeyExclusive parameters that you would use in normal table-handling code.



Please note: You need to have an Index different from the Sequential Access Index selected to use ranges.

Filters

??? [to be done]

BLOB Fields

If the table contains Blob fields, then the bottom area of the window will display the blobs in the active record (see picture above, the Blob field contains an image). This can be turned off in the View menu for browsing speed.

New Menus

When made the active window, the Table Browser window adds a few menus to the main menu. These are discussed in the next section.

©2004 Nexus Database Systems Pty Ltd.

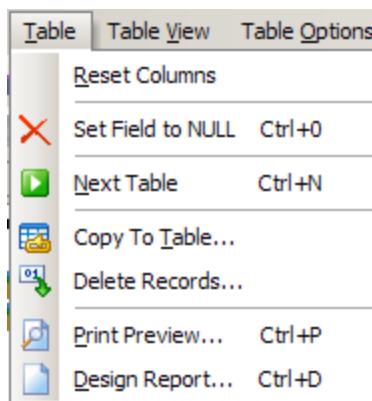


7.5.2.1 The Table Menu

The Table Menu



The Table menu is only visible if a table is active in the browsing window.



Most of its options are also available via the



Table Menu toolbar. Again the icons in the menu and toolbar refer to each other.

Reset Columns

Ensures that columns have reasonable min/max widths.

Set Field to Null (Ctrl+0)

Clears the value of the active field in the grid.

Next Table (Ctrl+N)

Closes the active table, and opens the next table in the database (in alphabetical order) in the same window.

Copy To Table...

Copies the contents of the table to another table, respecting any active filters and ranges.

Delete Records...

Deletes the records in the table, respecting any active filters and ranges. Records not selected by filters or ranges remain.

Print Preview (Ctrl-P) ...

Opens the table in print preview. See section Printing Reports for more on the printing.

Design Report (Ctrl-D) ...

If the Report Engine DLL is installed, opens the report designer. See section Printing Reports for more on the printing.

©2004 Nexus Database Systems Pty Ltd.

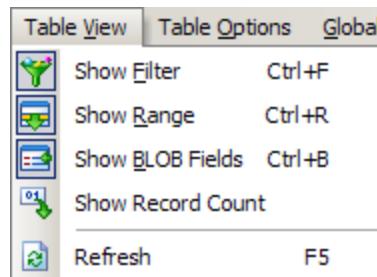


7.5.2.2 The Table View Menu

The Table View Menu



The Table View menu, just like the Table menu is only visible, if you're browsing a table. Most of its options allow you to minimize clutter in the browser window and maximize the amount of data you can see.



Again, there is a toolbar with the same functionality available



and as usual the icons tell you what is what.

Show Filter (Ctrl-F)

Shows or hides the Filter area of the Table Browser.

Show Range (Ctrl-R)

Shows or hides the Range area of the Table Browser.

Show Blob Fields (Ctrl-B)

Shows or hides the Blob display area of the Table Browser.

Show Record Count

Shows or hides the display of RecordCount in the status bar. If you've a filter or range enabled that can have a huge impact on display speed, so only switch it on if really needed.

Refresh (F5)

Refreshes the browser window. This has in most cases no effect on the Internal Server, but will make sure on remote connections that you see the current data.

©2004 Nexus Database Systems Pty Ltd.

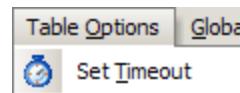


7.5.2.3 The Table Options Menu

The Table Options Menu



The Table Options menu too, is only visible while browsing a table.

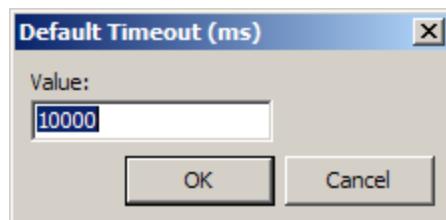


And again it has a corresponding toolbar:



Set Timeout

For operations on large table, it might be necessary to increase this setting. The default timeout value is set to 10 seconds as shown in the screen shot below.



Note that the value is presented in milliseconds (there are 1,000 milliseconds per second). If you set this value to 0 milliseconds, then the timeout is ignored (has, to all intent purposes, an infinite value).

If you set this value to -1, then typically the value of the Global Options will be used.

©2004 Nexus Database Systems Pty Ltd.



7.5.3 Using SQL

Using SQL



This is where SQL queries can be tested and executed. For a full reference of SQL:2003 and the various NexusDB SQL extensions please take a look at the according SQL Reference Book of this

manual.

The SQL script input area accepts scripts of any length. Multi-statement scripts must have each single statement terminated with ';'. If the statement is a single SELECT statement, or the last statement in a multi-statement query is a SELECT, then the query result set displays in the grid after execution.

The screenshot shows the Oracle Enterprise Manager interface. At the top left, there is a SQL script input area containing the command: `SELECT * from CD`. To the right of the script, a message window displays the status: "Simplifying: SELECT ALL FROM CD; no change" and "Executing SELECT ALL FROM CD;". Below the script area is a data grid with the following columns: DiskSerial, Artist, Title, AlbumCover, Created, and LastUpdated. The data grid contains the following records:

DiskSerial	Artist	Title	AlbumCover	Created	LastUpdated
17182939	Rush	Presto		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.
-111380303	Police, The	Message in a box - disc IV		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.
18787	Estefan, Gloria	Coming out of the dark		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.
203302	Prince	Lovesexy		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.
734795	Aha	Crying in the rain		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.
925236	Haza, Ofra	Galbi		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.
939861	Toto	Stop loving you		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.
993152	Houston, Whitney	Didn't we almost have it all		7/06/2004 4:27:39 p.m.	7/06/2004 4:27:39 p.m.

Below the data grid is a section for managing album covers, featuring buttons for "Load from file...", "Save to file...", and "Clear", along with checkboxes for "Stretch" and "Proportional". At the bottom of the interface, there is a toolbar with various navigation and search icons, and a status bar displaying: "Query retrieved", "Queries = 1", "Records: 701", "Execution time = 30 ms", and "BLOB Size: 1,512".

Data Grid

To find out more about the possibilities the data grid provides please check out the chapter Customizing the Data Grid.

Query Window

If the last statement is not a SELECT, the grid will be empty and instead the status bar will display how many records were affected by the last non-SELECT statement. If live (editable) queries is turned on in the local and global options, and the query syntax conforms to what is supported for live queries, the grid will be editable, otherwise it will be read-only. If you want to change values in a non-live query, either use UPDATE statements, or open the table in a table browser window.

The query browser window allows you to have several SELECT statements open on the same database at once. This is a handy way to compare different variants of a query to see which records

they return. Use the Create New Connection function to add statements. The list of statements is displayed in the combo box in the toolbar.

The Log/Execution Plan Window

To view NexusDB's SQL execution plan for a query, a log window can be activated. Note that in order to return the necessary log, the query must include the #L+ option. It is auto inserted at the top of the query text when you turn on the log via the menu.

Blob Fields

If the query result set contains Blob fields, then the bottom area of the window will display the blobs in the active record. This can be turned off in the View menu for browsing speed.

New Menus

The following menus are added to the main menu when a SQL Browser window is active: the Query menu, the Query View menu, the Query Connections menu and the Query Options menu. We now take a closer look at each of them.

©2004 Nexus Database Systems Pty Ltd.



7.5.3.1 The Query Menu

The Query Menu



The Query menu is only visible if a query window is active. Its main purpose is to let the user load, save and execute SQL queries.

Query		
	Query View	Query Connections
	<u>Execute</u>	Ctrl+E
	<u>Open</u>	Ctrl+O
	<u>Save</u>	Ctrl+S
	<u>Find</u>	Ctrl+F
	<u>Parameter Values</u>	Ctrl+M
	<u>Copy To Table...</u>	
	<u>Print Preview</u>	Ctrl+R
	<u>Design Report</u>	Ctrl+T

Alternatively you can also use the



Query toolbar.

Execute (Ctrl+E or F9)

Opens or runs an SQL statement. If the statement contains parameters, a dialog will pop up to get parameter values before execution starts.

Open (Ctrl+O)

Read in SQL scripts stored on disk.

Save (Ctrl+S)

Save SQL scripts to disk.

Find (Ctrl+F)

Search for a particular word in the current SQL script.

Parameter values (Ctrl+M)

Open the parameter values dialog manually.

??? [dialog to be done]

Copy To Table

Copy the active result set to a new table.

??? [dialog to be done]

Print Preview (Ctrl+R)

Opens the current active result set in the Report Engine print preview. See section Printing Reports chapter for more information.

Design Report (Ctrl+T)

Opens the Report Engine design view. See section Printing Reports chapter for more information.

©2004 Nexus Database Systems Pty Ltd.

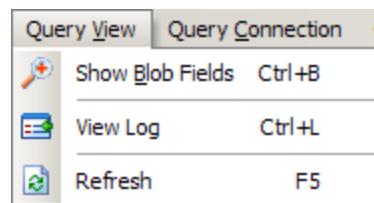


7.5.3.2 The Query View Menu

The Query View Menu



The Query View too menu is only visible if a query window is active. As with the Table View menu, most of it's options allow you to minimize clutter in the browser window and maximize the amount of data you can see.



Again there's a corresponding toolbar available:



Show Blob Fields (Ctrl+B)

Shows or hides the Blob display area of the SQL Browser.

View Log (Ctrl+L)

Shows or hides the display of the log (also called execution plan).

??? [some information on the execution plan from Per needed].

Refresh (F5)

Reruns the active query and thus completely updates the data in the grid.

©2004 Nexus Database Systems Pty Ltd.

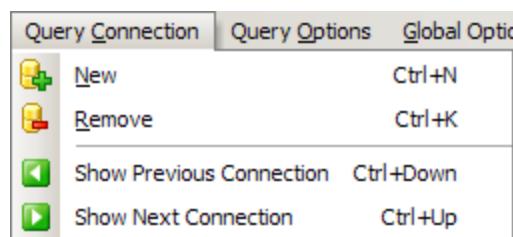


7.5.3.3 The Query Connections Menu

The Query Connections Menu



As with all other query depending menus, this one is only visible while you're working in a Query window. Its main purpose is to allow the user to work with several queries in one query window.



All the options can also be accessed through the



Query Connection toolbar

New (Ctrl+N)

Opens a new SQL query, leaving the existing one in the list of open queries.

Remove (Ctrl+K)

Closes the currently active SQL query.

Show Previous Connection (Ctrl+Down)

If several SQL queries are available, you can use this option to go back to the one you used before.

Show Next Connection (Ctrl+Up)

If several SQL queries are available, you can active the next one in the window list.

©2004 Nexus Database Systems Pty Ltd.

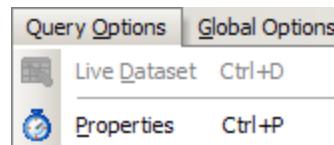


7.5.3.4 The Query Options Menu

The Query Options Menu



The Query Options menu lets the user switch live datasets on and set query specific options.



As with most other menus, there's a corresponding toolbar:



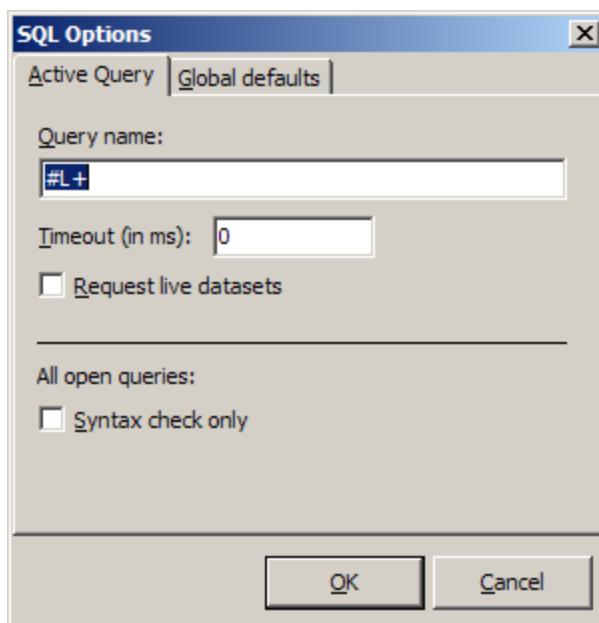
Live Dataset (Ctrl+D)

Determines whether the next SELECT statement attempts to return a live (editable) dataset. Turning it on does not guarantee that the result set is editable.

Properties (Ctrl+P)

Opens the query property dialog.

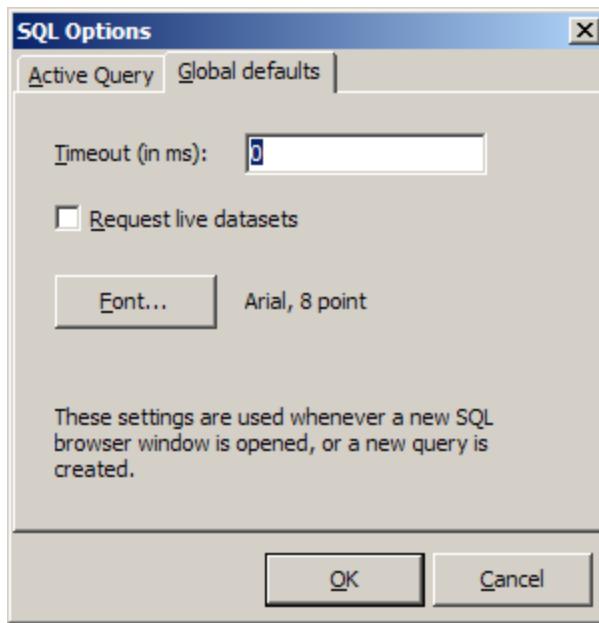
The Query property dialog has 2 tabs.



In the first tab `Active Query` you can change the live dataset, timeout and the query name for the currently active query. Note that the value for timeout is presented in milliseconds (there are 1,000 milliseconds per second). If you set this value to 0 milliseconds, then the timeout is ignored (has, to all intent purposes, an infinite value). This can be handy when you are testing a particularly complex query which may take many minutes to evaluate however, generally speaking, you would want the timeout to be a finite value to guard against deadlock situations.

If you set this value to -1, then typically the value of the Global Options will be used.

You can also switch syntax highlighting on and off (this affects **all** open windows). In the second tab `Global Defaults`



you can set the timeout, live dataset and Font used for all subsequently opened queries.

©2004 Nexus Database Systems Pty Ltd.



7.6 Importing CSV Files

Importing CSV Files



This wizard allow you to import a wide range of text files, both delimited and fixed width formats. First you are presented a select file dialog to point to the file you want to import.

After that a wizard with 3 steps pops up, which will help you to set all options for the import of the file.

©2004 Nexus Database Systems Pty Ltd.

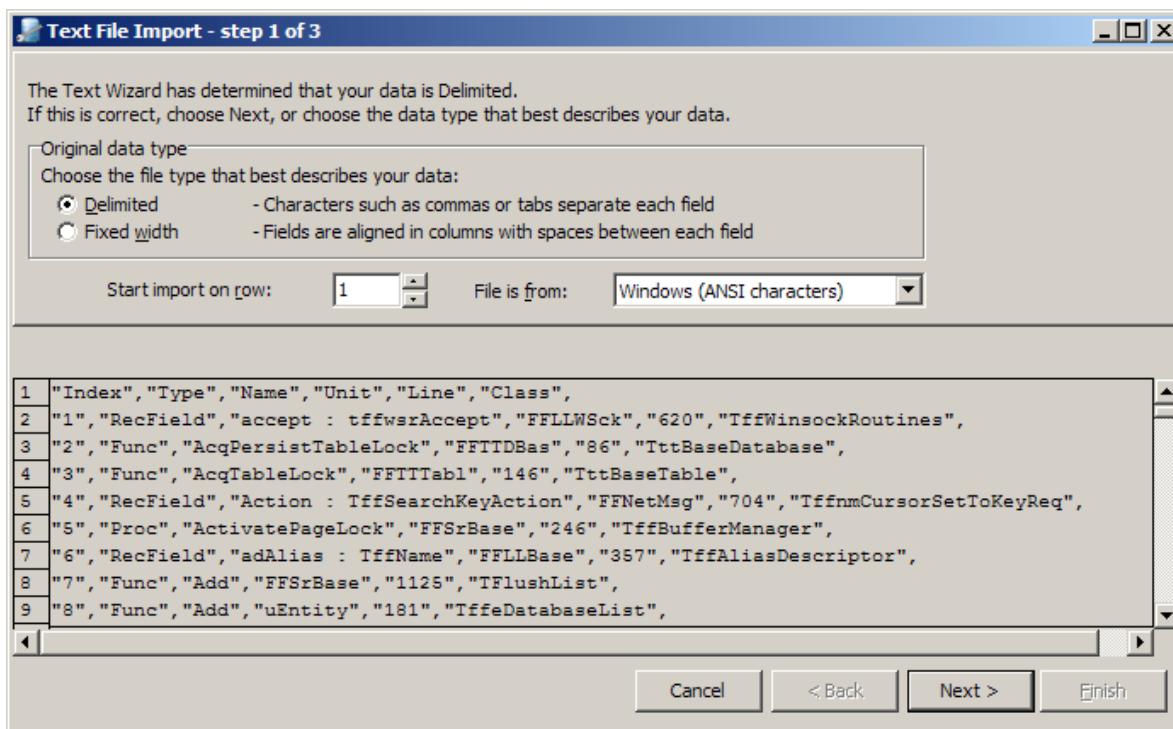


7.6.1 Step One - File Type

Step One - File Type



The first page of the wizard displays the unformatted contents of the start of the file. The wizard also tries to analyze the start of the file to determine what type of file it is. Most of the time it will guess correctly, but you should check that the Delimited or Fixed width selection matches the file contents.



Also, make sure to change the "Start import on row" setting if there are blank lines or descriptive column names at the top of the file, to avoid importing blank or erroneous data.

Note that if "Start import on row" is larger than 1, then the column descriptions (if any) will be used to create field names in the "New table" function in Step 3.

©2004 Nexus Database Systems Pty Ltd.

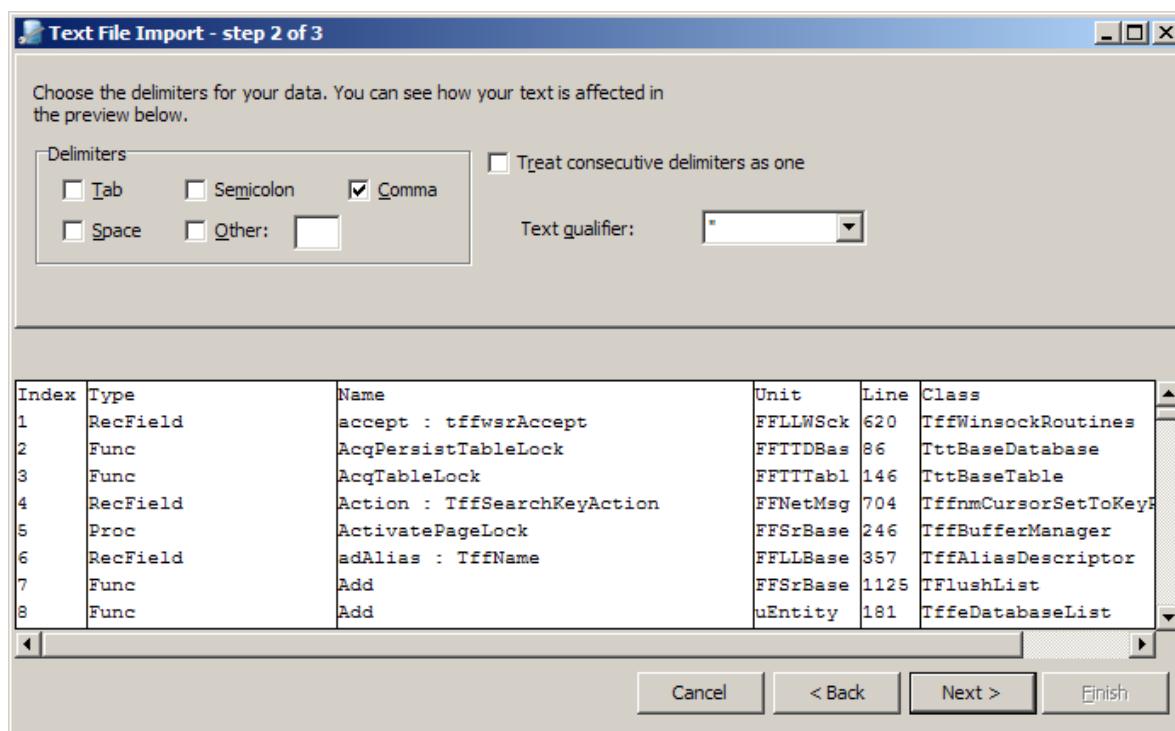


7.6.2 Step Two - Delimiters

Step Two - Delimiters



In the second wizard page, select all delimiters used in the file you are importing from. The Other setting can be used if an uncommon delimiter has been used. The preview of the columns changes to indicate how the options affect the import.



©2004 Nexus Database Systems Pty Ltd.

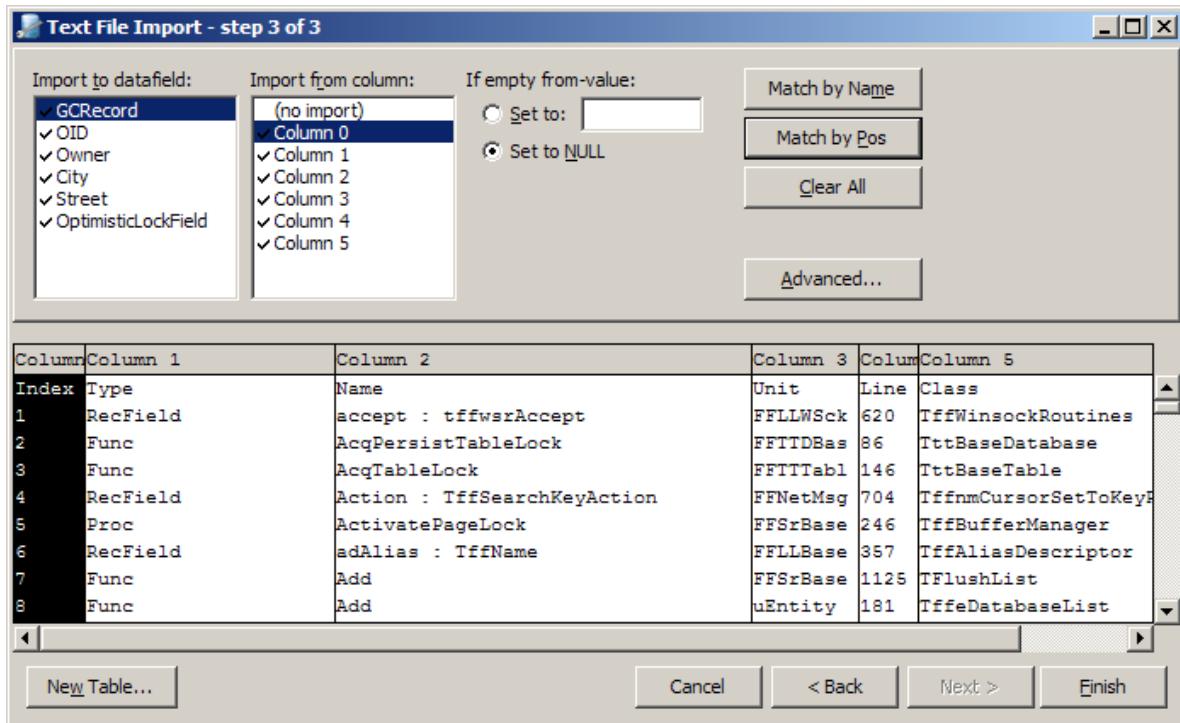


7.6.3 Step Three - Field Formats and Advanced Options

Step Three - Field Formats and Advanced Options



On the third and last wizard page, set formatting options, and link table fields with import columns.



The Import to data field list contains the list of fields in the table you are importing to, or a generic numbered name (field 1 etc) if you haven't created a table yet.

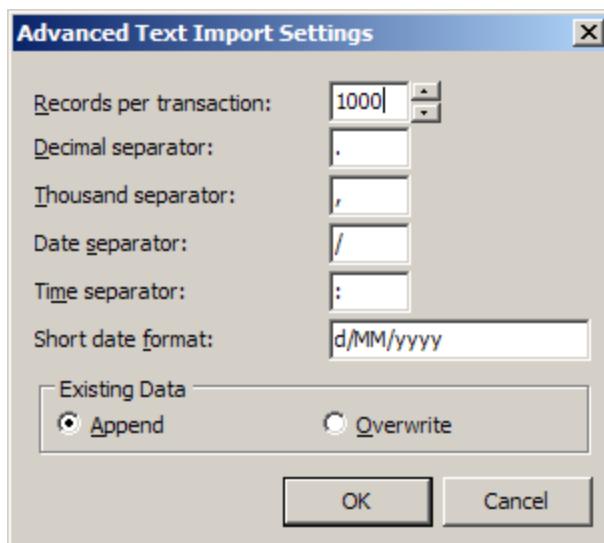
The Import from column lists the columns in the text file. If there are column descriptions in the first line of the file, and you chose "Start import on row" to be larger than 1 on the first wizard page, then the descriptions are used, otherwise generic numbered column names are displayed.

To link a field in the table with a column, click the field in the field list first, and then click the column you want to import into the field. If you make a wrong selection and want to remove the column, click the (no import) item at the top of the column list. Check marks appear next to fields and columns that have been selected.

The column preview will display the chosen column and invert its colour. You may select columns by clicking either the Import from column list, or directly in the preview.

The New Table button will bring up the Create Table dialog, with fields, field types and units preset from the values detected in the CSV file. Ensure that text fields are large enough to hold the longest text you have in a column etc. The detection process only looks at the start of a file, so there might be longer text strings in a column than what is detected from the first part of the file. If you successfully Create the table, the new table is now the one being imported into.

Before linking individual fields and creating a new table, you should open the Advanced settings dialog and set the options to match what is contained in the CSV file. Also, to speed up importing, it is a good idea to use a higher number for Records per transaction (recommend 1.000-10.000).



To ensure proper handling of date/time formats, make sure to set the date/time options in the Advanced settings dialog, before pressing the "New Table" button on that wizard page.

©2004 Nexus Database Systems Pty Ltd.



7.7 Printing Reports

Printing Reports



EM includes an option to print reports from the following dialogs.

1. The Table Browser – Print Preview and Design Report menu options will be enabled
2. SQL Query Browser – same options as above are available under Query
3. Dictionary Browser – the Print button on the lower left-hand side of the window is enabled

Enterprise Manager uses FastReports for this purpose. Here's a short overview of how to use it. For an extensive documentation of the vast possibilities please follow the link.

??? [screenshots and quick intro to fastreports needed]

©2004 Nexus Database Systems Pty Ltd.



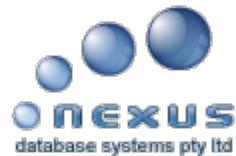
7.8 Runtime Loaded Packages

Runtime Loaded Packages

??? [to be done]



©2004 Nexus Database Systems Pty Ltd.



8 The Importer

The Importer



©2004 Nexus Database Systems Pty Ltd.



8.1 Introduction

Introduction



The "Importer Suite" is a set of utilities to convert tables from one format to another.

Program	Source Engine	Destination Engine
ADO2FF	Microsoft ActiveX Data Objects	FlashFiler 2
ADO2NX	Microsoft ActiveX Data Objects	Nexus DB
BDE2FF	Borland Database Engine	FlashFiler 2
BDE2NX	Borland Database Engine	Nexus DB
DAO2FF	Microsoft Data Access Objects	FlashFiler 2
DAO2NX	Microsoft Data Access Objects	Nexus DB
FF2NX	FlashFiler 2	Nexus DB
Importer	Any supported engine	Any supported engine

The Importer utility is designed as a test environment for the implementation.

©2004 Nexus Database Systems Pty Ltd.



8.2 Design

Design



The basic approach is to use a TTable compatible toolbox as wrapper around the various source engines and build the structure of the destination tables from the Delphi internal descriptions. After conversion of the table structure the data is imported with a simple loop of single record inserts wrapped into transactions of configurable size. The utilities try to convert as many tables as possible. Therefore failures during the import operation are logged but do not stop the operation itself. The captions might not be the most intuitive ones, but the need of unique shortcuts introduces some restrictions on the possible descriptions.

©2004 Nexus Database Systems Pty Ltd.



8.2.1 Known Issues

Known Issues



This approach adds an additional abstraction layer. Numeric types with a fixed number of decimals are mapped onto integer values or onto floating point values. But since neither FlashFiler nor Nexus DB support numeric values with a fixed number of decimals, which can be freely chosen by the user, this loss of meta data seems to be tolerable.

The converters only support compound indexes, i.e. indexes that are created by the value of one or more fields. Expression indexes and/or indexes containing unknown fields are ignored.

The importers are designed for the use by developers. This means that it is assumed that the user has enough privileges for the operation and knows how to use the source environment to determine the reason for import failures.

©2004 Nexus Database Systems Pty Ltd.



8.3 Command Line Usage

Command Line Usage



Besides the interactive definition of parameters the utilities support command line parameters. These can either been added explicitly to the program calls or implicitly via a configuration file.

`ff2nx @ff2nx_parameters.txt`

The parameter "auto" switches each specific importer to batch mode. The parameter "nowait" removes the need to leave the programs by pressing a button and allows the usage of the utilities as part of an installation script. Thus the following calls can be used in case of simple conversions.

```
ff2nx "-fd=<Directory
containing the FlashFiler
tables>" "-nd=<Directory
containing the Nexus
tables>" -auto -nowait
```

```
bde2nx  "-ba=<BDE Alias>" "-  
nd=<Directory containing the  
Nexus tables>" -auto -nowait
```

If a parameter which does not allow more than one value is used the last definition wins. The configuration files created by the utilities always set the auto and the nowait parameter to false. For an automatic run the files have to be patched or the auto and nowait parameters have to be set again after the introduction of the parameter file. The created configuration files are documented and are considered as description of the possible parameters.

©2004 Nexus Database Systems Pty Ltd.



8.3.1 Configuration file example

Configuration file example



Example of a configuration file

```
# Nexus Import Parameters  
Version 1.00  
# === Transfer Manager  
=====  
=====  
# Automatic Mode  
-auto-  
# Leave progress display  
automatically  
-nowait-  
# Transfer Type (nttMerge,  
nttReplace, nttDuplicate,  
nttLayout)  
-ot=nttDuplicate  
# String Handling  
(nshSource, nshNull,  
nshEmpty)  
-os=nshNull  
# Transaction block size  
#     Value  
#     Unit  
-ob=2;Record(s)  
# Attributes for text fields  
in indexes  
#     Language  
#     Sorting  
#     Locale  
#     Ignore Case  
#     Ignore Width  
#     Use String Sort  
#     Ignore Kana Type
```

```
#      Ignore Symbols
#      Ignore Non Space
-oi=German Germany;German
Phone Book
order;$00010407;False;False;
True;False;False;False
# Substitutes for indexes
without name
#      Primary index
#      Secondary index(es)
-op=NX$PRIMARY;NX$INDEX
# === Borland Database
Engine
=====
=====
# Directory Mode
-bd+
# Alias Name
-ba=C:
\Workbnch\nxtransfer\BDE
# Table Name(s)
-bt=master.dbf
# Field Name(s)
-bf=SYMBOL
-bf=CO_NAME
-bf=EXCHANGE
-bf=CUR_PRICE
-bf=YRL_HIGH
-bf=YRL_LOW
-bf=P_E_RATIO
-bf=BETA
-bf=PROJ_GRTH
-bf=PRICE_CHG
-bf=SAFETY
-bf=RANK
-bf=RCMNDATION
-bf=RISK
# Range Usage
-rr+
# Range Index
#
RangeStartKeyExclusive
#      RangeEndKeyExclusive
#      Index Name
-ri=False;False;SYMBOL
# Range start
#     FieldName
#     FieldType
#      Null
#      Value
-rs=SYMBOL;ftString;False;HS

# Range end
#     FieldName
#     FieldType
```

```

#      Null
#      Value
-re=SYMBOL;ftString;False;WM

# Filter usage
-rf+
# Filter definition
-rd=RISK = 'LOW'
# === Nexus Engine
=====
=====

# Server Name
-ns=Directory Mode
(Embedded)
# Database Name
-nd=C:\nexus
# Table Name(s)
-nt=submaster
# Field Name(s)

```

©2004 Nexus Database Systems Pty Ltd.



8.3.2 Known Issue

Known Issue



The security handling needs to be improved. At the moment the triggered login prompts block the usability for scripting. But storing the user/password combination is a security risk. Suggestions are welcome.

©2004 Nexus Database Systems Pty Ltd.



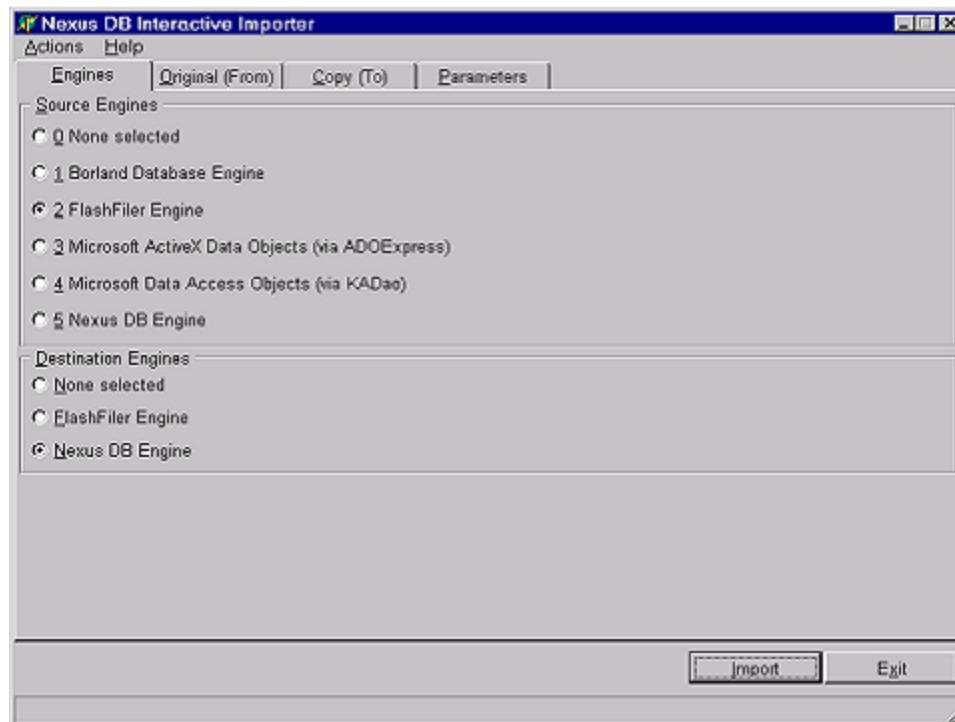
8.4 Interactive Usage

8.4.1 Engine Selection

Engine Selection



Since only the combination FlashFiler - Nexus is always available, the importer starts with this combination. The "None selected" are activated to indicate that the real selection requested an engine that is not available on the used PC.



©2004 Nexus Database Systems Pty Ltd.



8.4.2 Configuration of the Source Engines

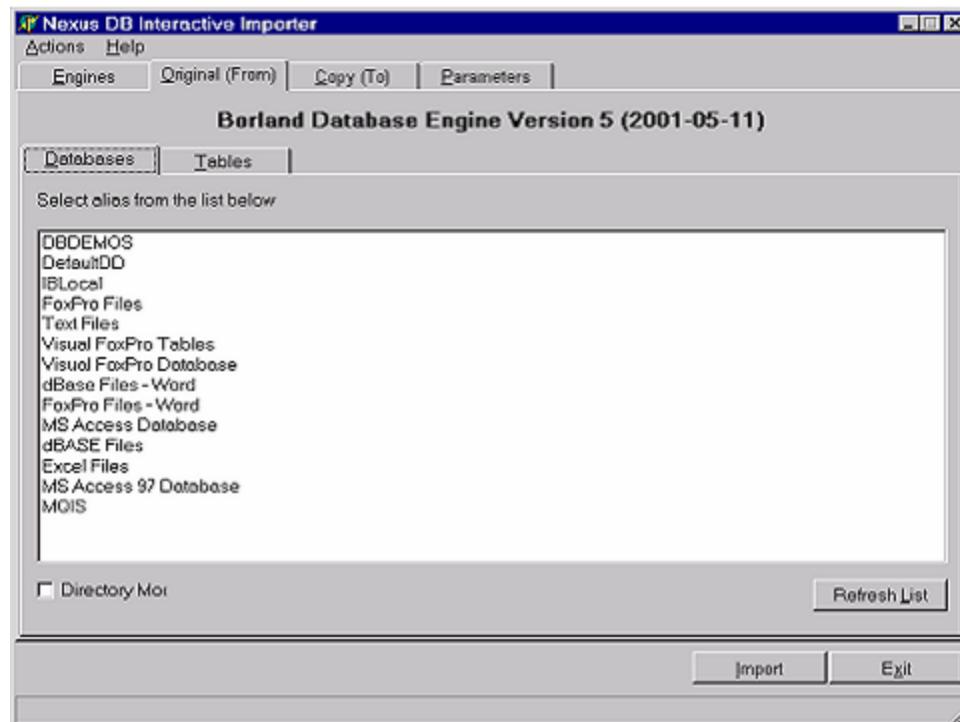
8.4.2.1 Borland Database Engine

Borland Database Engine

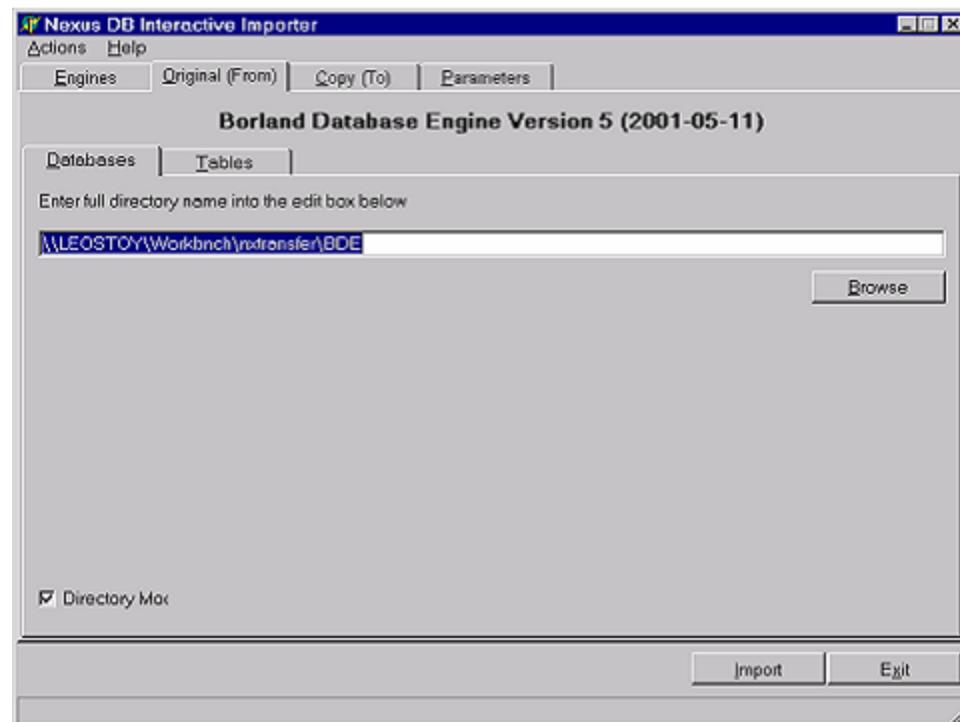


This engine is the one for which the importers are designed. All other engines have to be compatible with a TTable.

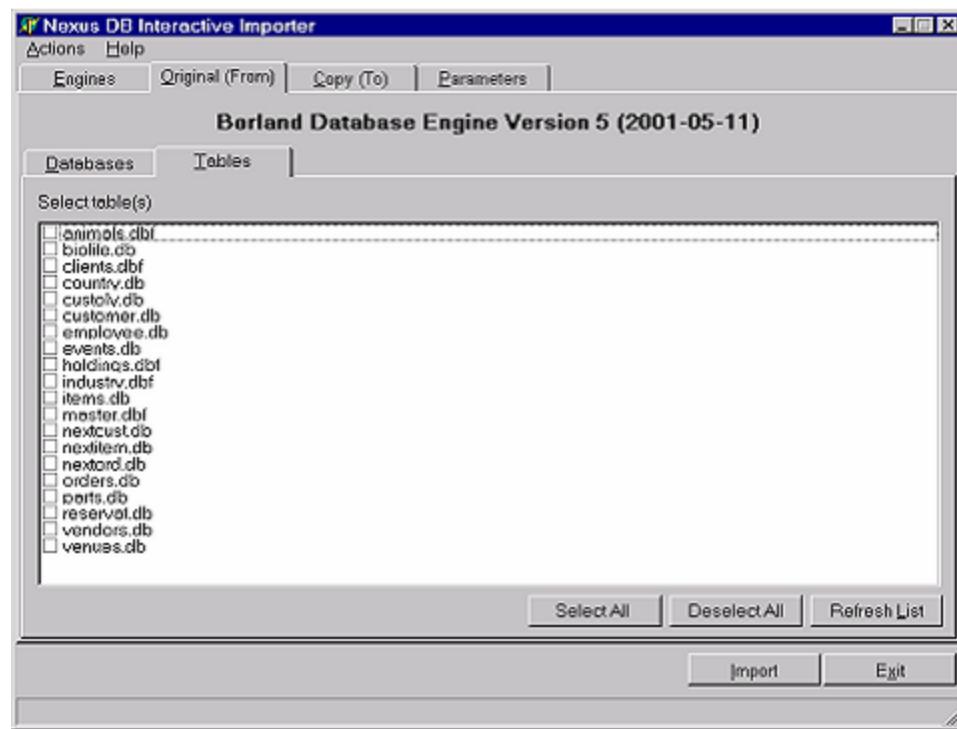
The header on the source engine tab always shows a description of the engine used. Since the BDE only returns the major version the time stamp of the ISAPI DLL is also given.



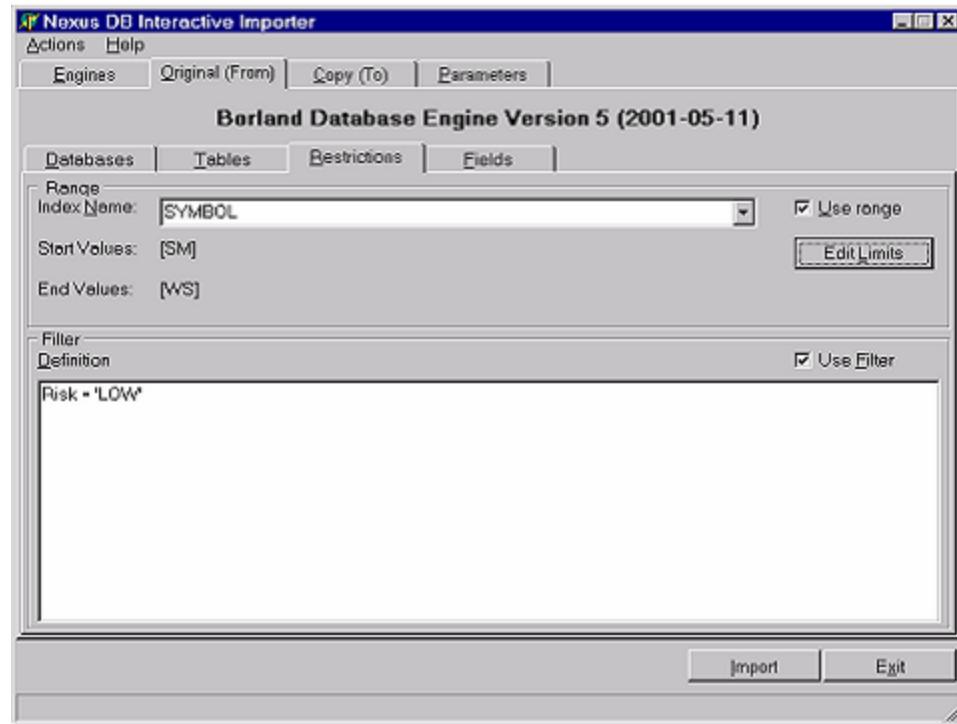
The first tab shows a list of available Alias names. Choose the one you want to use. If you want to import dBase or Paradox tables you can also switch to directory mode.



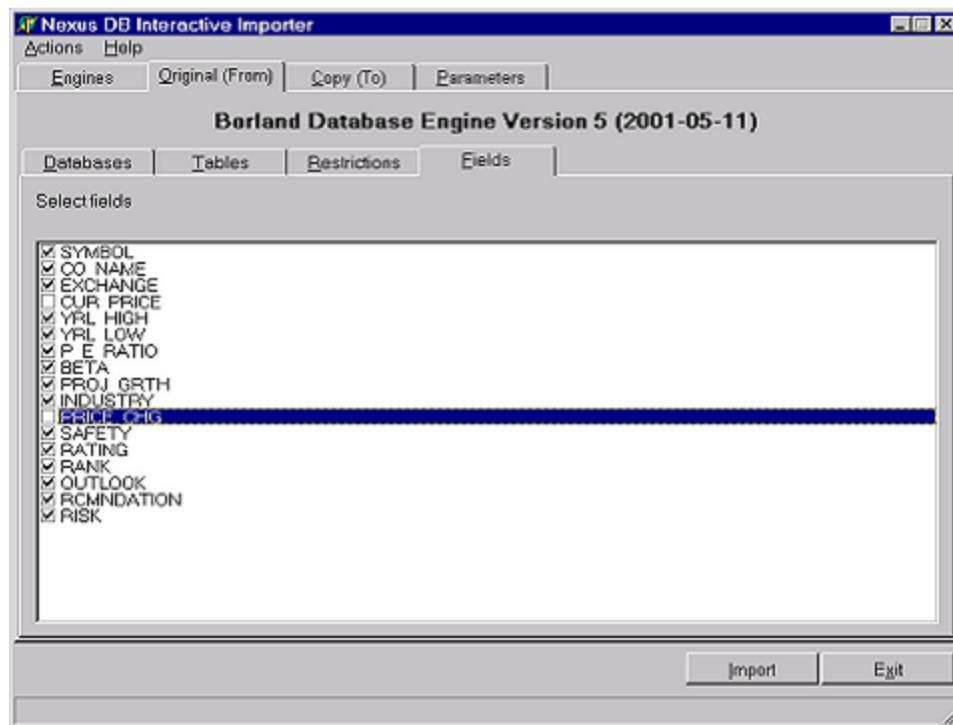
The selection of the tables uses the same views for all engines. Therefore it is only described for the BDE.



As first step you select one ore more tables.



If you import only one table, you can restrict the included records in the same way as in the data display of the Enterprise Manager.



Additionally you can define to import only a subset of the available fields.

©2004 Nexus Database Systems Pty Ltd.

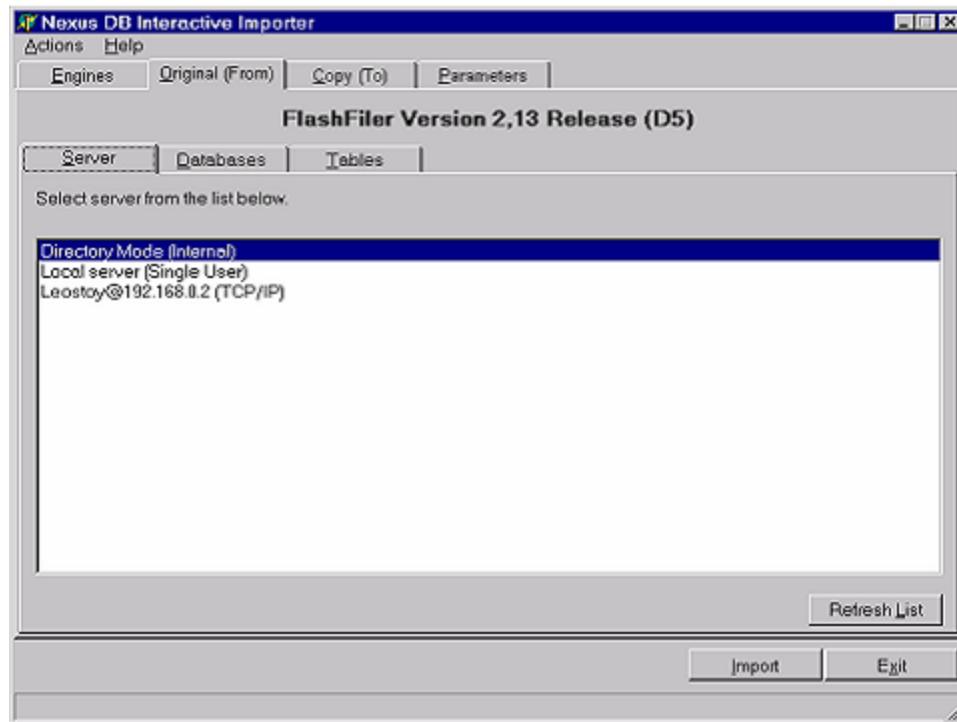


8.4.2.2 FlashFiler 2

FlashFiler 2

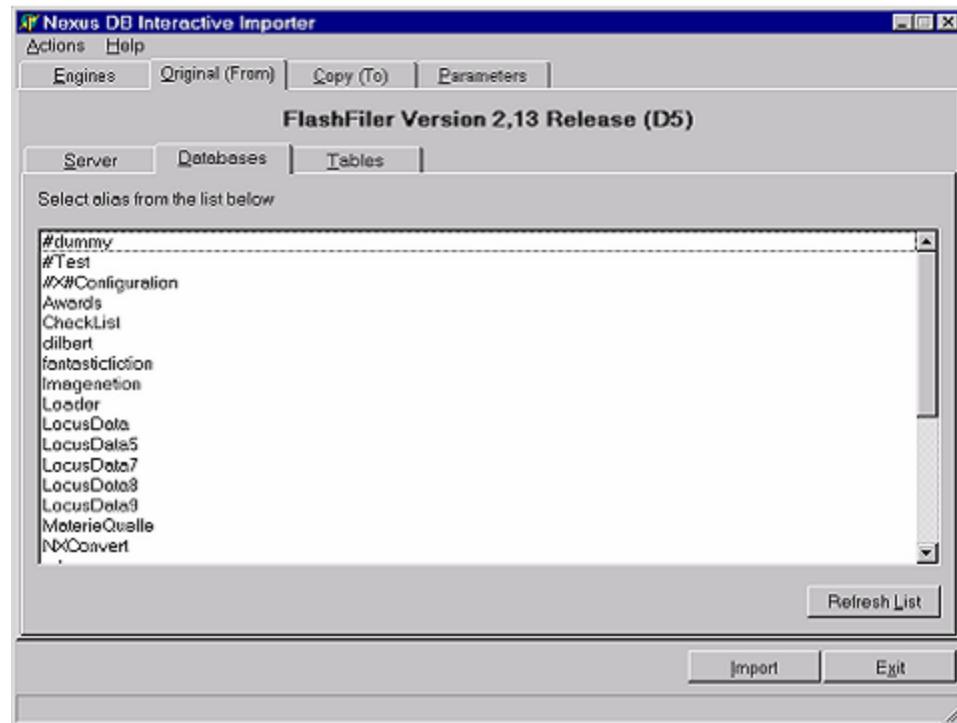


Both the commercial version and the Open Source version from SourceForge can be used. A specialized converter allows transferring more features of the source table than possible with the other engines. This includes the definition of additional files for BLOBs and indexes and their block size.

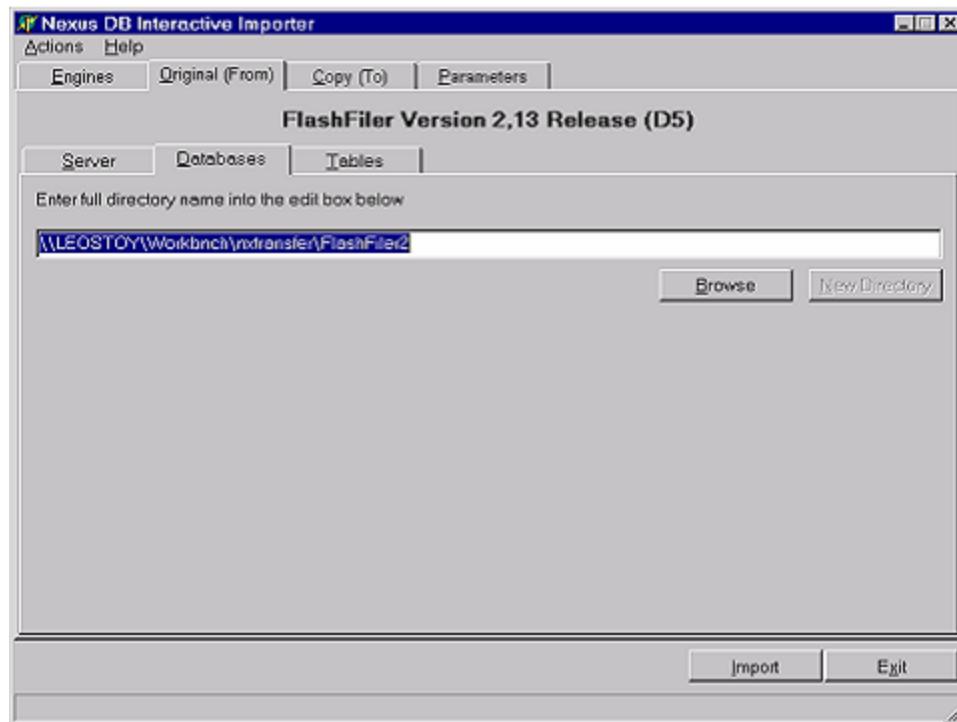


The importer can either use the embedded server or an external one.

The Local server (Single User) entry is always available, even if there is no external FF Server available, since the TffLegacyTransport has no real check for its presence.



If an external server was selected the destination database is chosen from the list of known aliases.



For the internal server the name of a directory is entered.

©2004 Nexus Database Systems Pty Ltd.

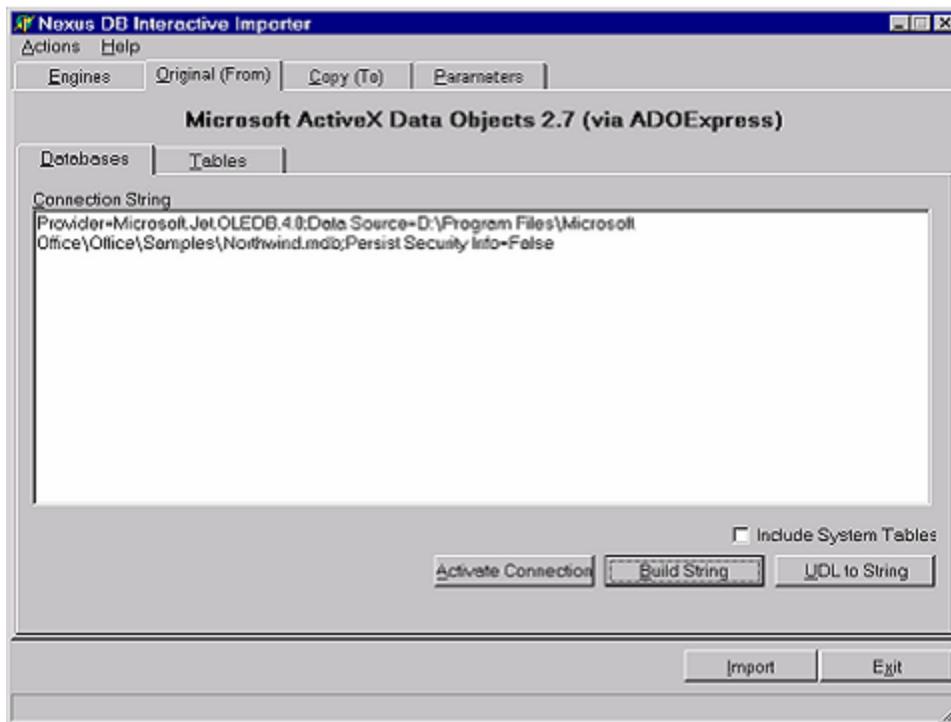


8.4.2.3 ADO

ADO



The used toolbox is the ADO wrapper, which is part of Delphi itself. For owners of Delphi 5 Professional the ADOExpress toolbox is a standalone product and might not be available. Its successor dbGO is part of all Delphi 6 and Delphi 7 versions, which support the BDE.



The ADO identifies the connection by a list of parameters. The easiest version is just a reference to a prepared definition file.

©2004 Nexus Database Systems Pty Ltd.

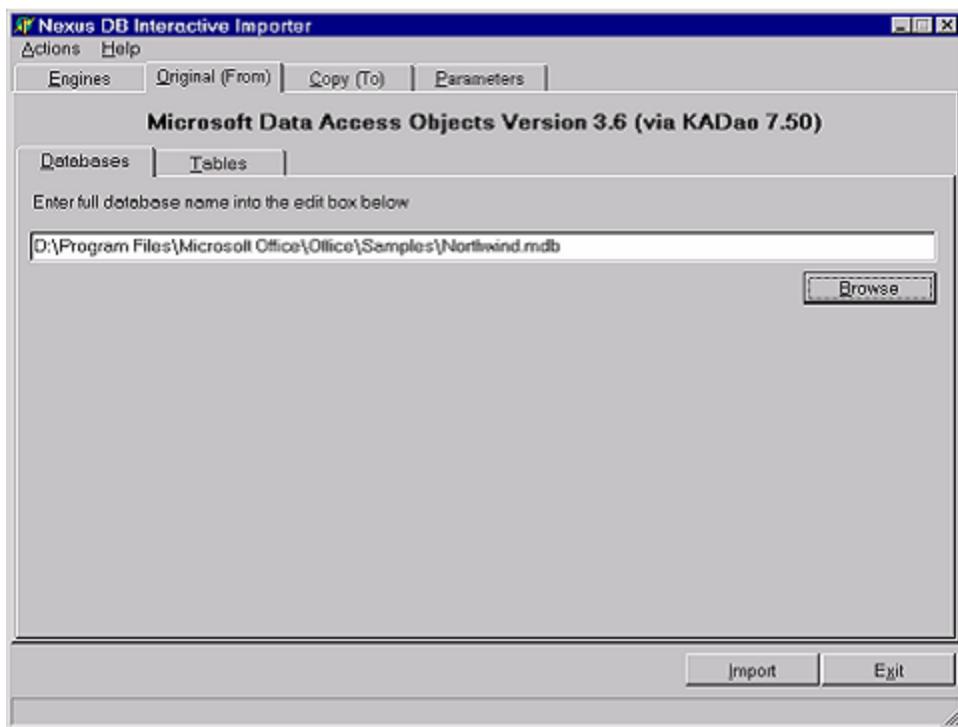


8.4.2.4 DAO

DAO



The used toolbox consists of the freeware KADao components from <http://kadao.dir.bg/>. These components need the Jet Engine, which is either installed as part of Access or the Microsoft MDAC package. The latest MDAC versions included in Windows 2000 and Windows XP do not longer contain the Jet engine, but it is still available as separate download on the Microsoft site. The BDE needs the DAO 3.5 version whereas Access 2000 and Access XP use the DAO 3.6 version. These importers are provided for the users of Delphi 5 Professional, which cannot switch to ADO to import data from databases created with these versions.



In this case the name of an Access table has to be entered.

The used freeware version of KADAO has limitations in the index support. Since the IndexDefs structure is empty, indexes are lost during the conversion. Thus use the ADO version if possible.

©2004 Nexus Database Systems Pty Ltd.

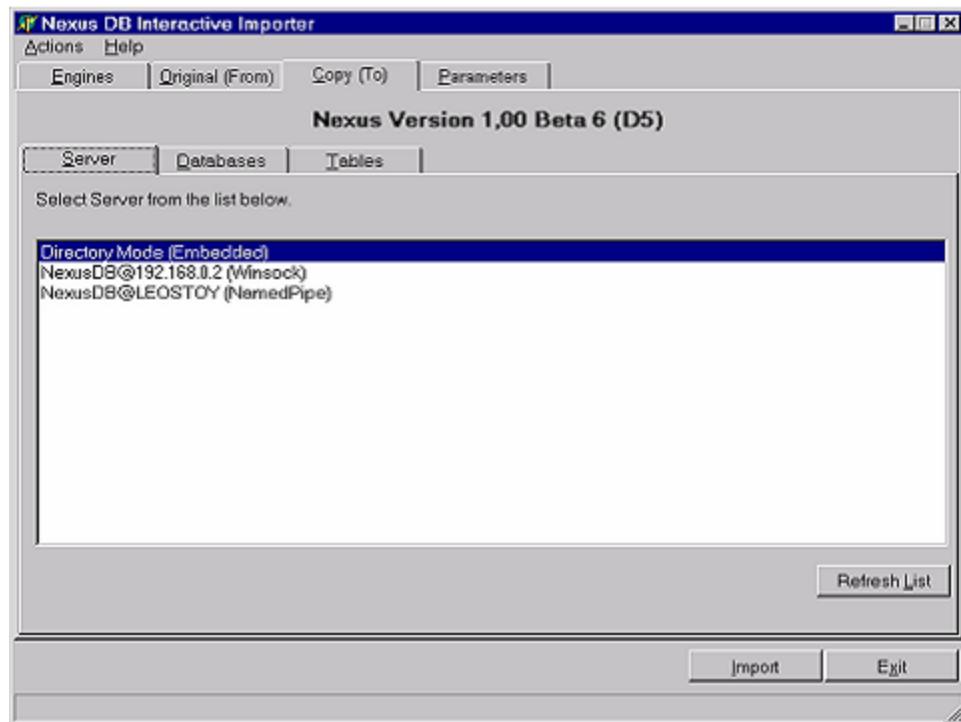


8.4.3 Configuration of the Destination Engines

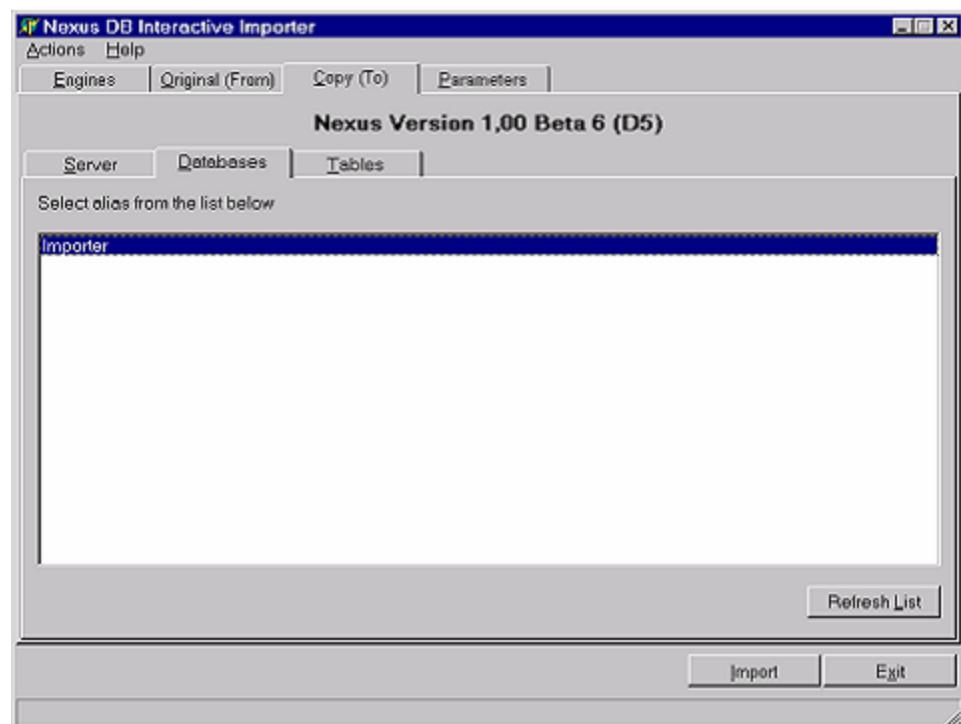
8.4.3.1 NexusDB

NexusDB

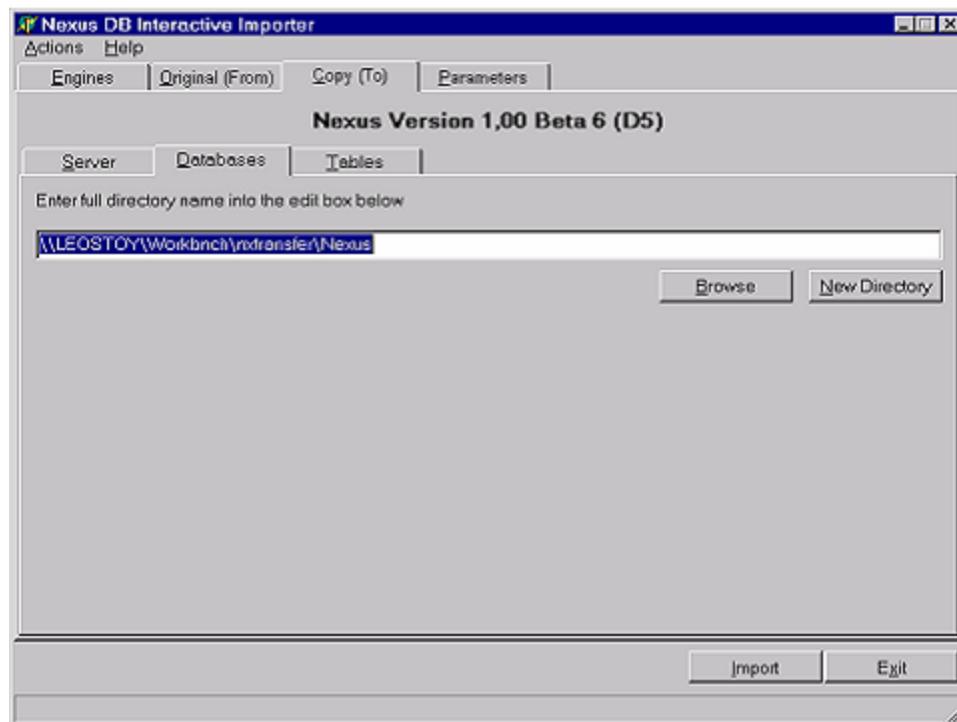




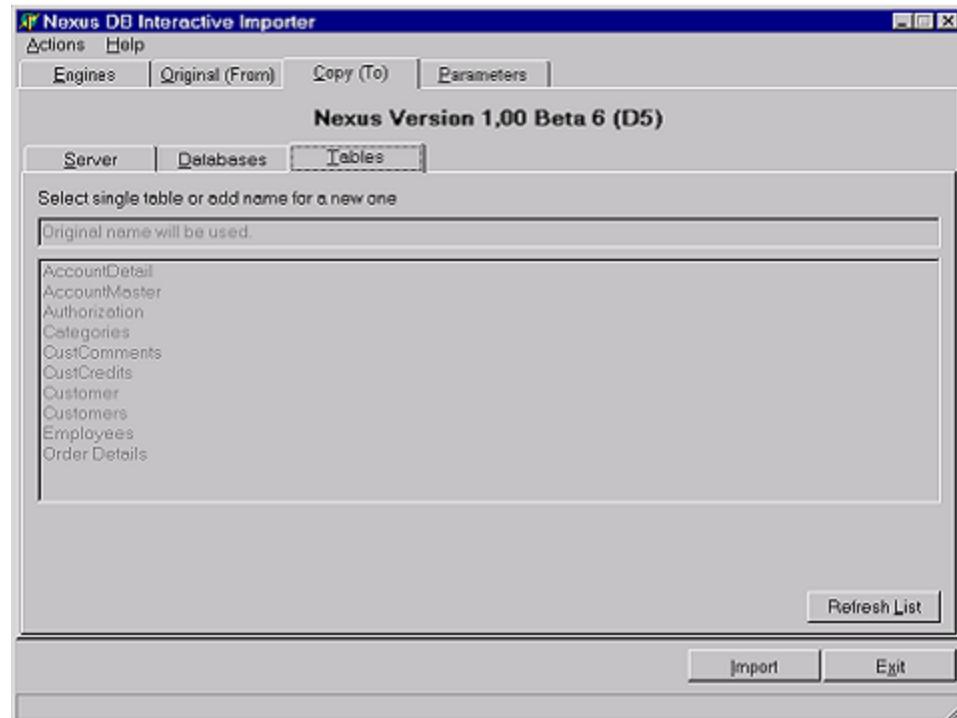
Select the server from the given list.



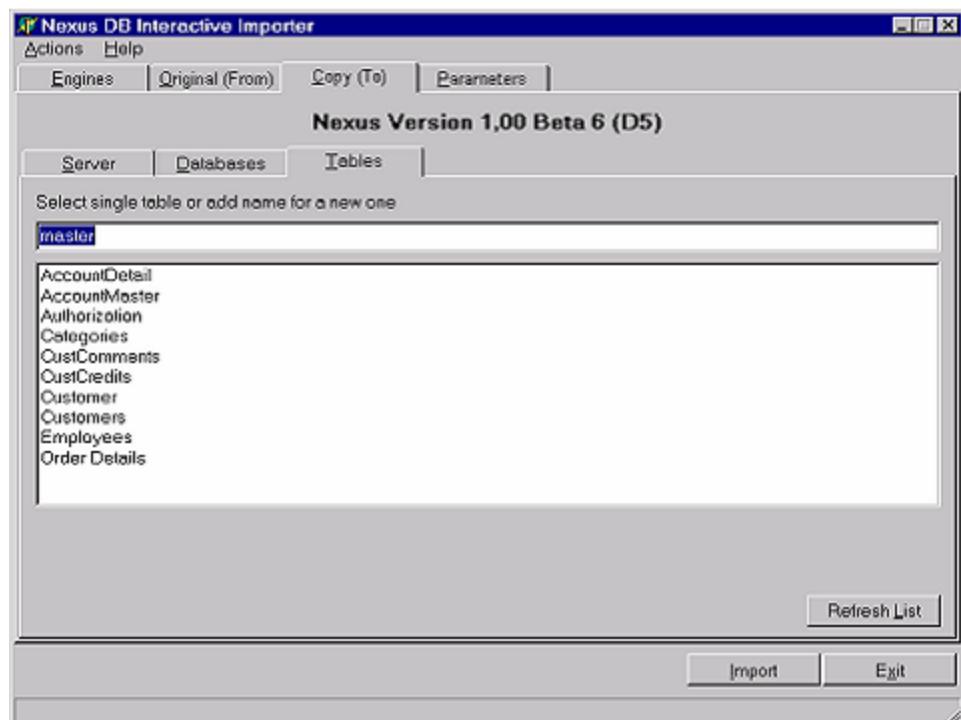
For external servers select an alias name.



For the embedded server enter an alias path.



If more than one source table has been selected, the name of the destination table is automatically created from the name of the source table.



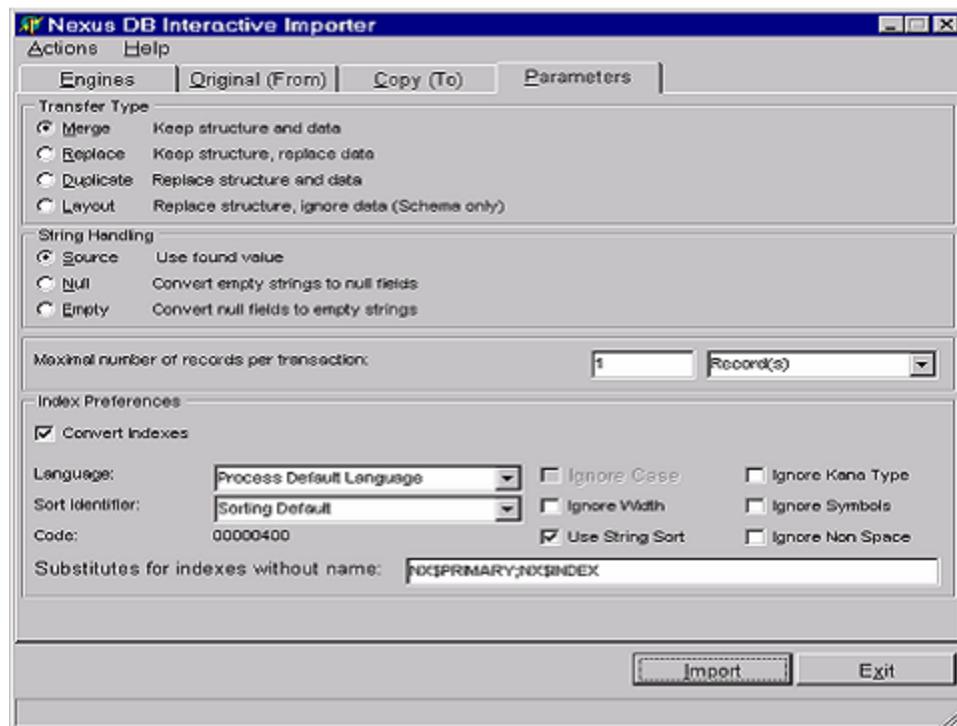
©2004 Nexus Database Systems Pty Ltd.



8.4.4 Configuration of the Conversion Process

Configuration of the Conversion Process





Transfer Type

- **Merge** - Only new tables are created. The data of the source table is appended to the data of the destination table. If the fields of the source table are no subset of the fields of the destination table or when the import creates duplicate values for a unique key the import operation fails.
- **Replace** - Only new tables are created. The data of the source table replaces the data of the destination table. If the fields of the source table are no subset of the fields of the destination table the import operation fails.
- **Duplicate** - Each table is created and the data of the source table is copied.
- **Layout** - Each table is created but no data is copied.

String Handling

- **Source** - Empty strings are copied as empty strings, Null fields are copied as null fields
- **Null** - Null fields are copied as empty strings
- **Empty** - Empty strings are copied as Null fields

Index Preferences

- The first parameter allows you to ignore all indexes. This might speed up the import operation, but adds the need to add the indexes manually.
- The next block defines the parameters for a string field inside an index. Sort direction and case sensitivity are set according to the source index definition, but for all other values a default can be chosen.

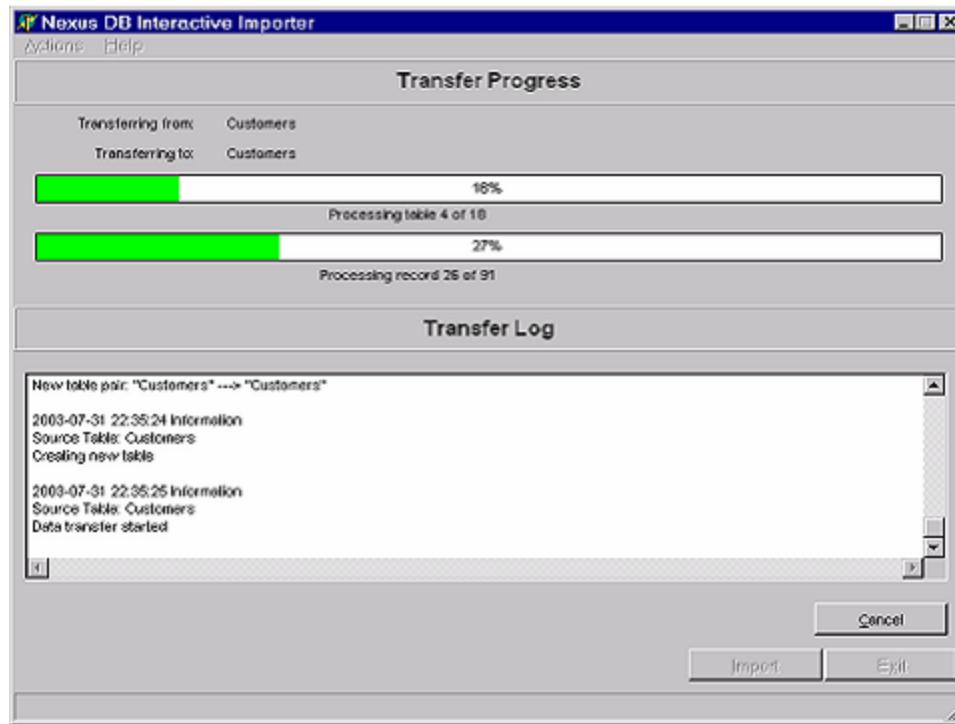
- The last entry controls the generation of index names for indexes without name in the source table. The first value is the name for the primary index, which will be promoted to default index in Nexus DB and the prefix for alternate indexes.

©2004 Nexus Database Systems Pty Ltd.

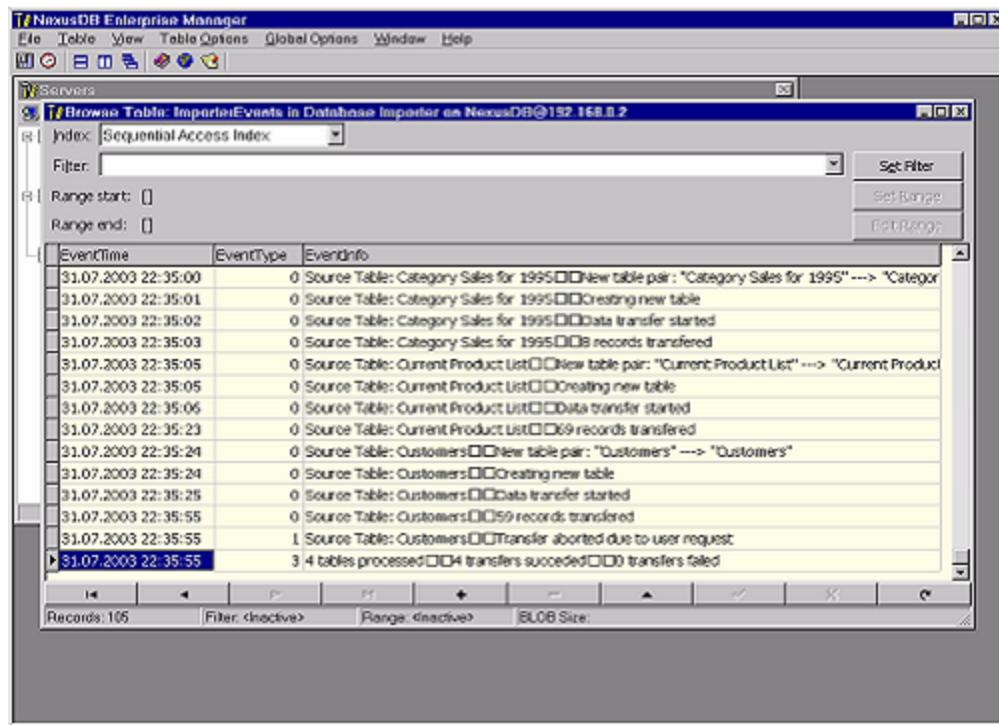


8.4.5 Transfer Process

Transfer Process



During the import operation the progress is shown and all events are logged. The Event log remains visible until either the Back button or the Exit button is pressed. The need for this user interaction can be removed with the nowait command line parameter.



The events are not only added to the list on the progress view, but also added to the table ImporterEvents in the same directory than the importer application. This allows the unattended usage of the utilities by an installation script. The event info contains line breaks that are not displayed by the Enterprise Manager.

©2004 Nexus Database Systems Pty Ltd.



8.5 Gotcha's

8.5.1 Side Effects of the "Directory Mode"

Side Effects of the "Directory Mode"



Some engines allow the usage of multi-file tables. In this case the additional files might use the same extension in both the source and the destination engine. In case of a FlashFiler to Nexus conversion this situation is guaranteed. Therefore the source directory and the destination directory must be different.

©2004 Nexus Database Systems Pty Ltd.



8.5.2 Misinterpretation of the Table Types

Misinterpretation of the Table Types



If an engine tries to determine the type of a table by its extension misinterpretations can happen. The most common situations are engines that use dBase compatible main files, but optimized index files and/or BLOB files with a different extension. The existence of these files is announced by bytes in the table header. The BDE supports FoxPro databases (CDX engines) but without a well-configured alias they are sometimes considered to be original dBase files. If this happens the BDE can't open the table and the importer utilities fail. The only possible solution is to remove the indexes with the utilities of the source engine or to patch the table headers.

©2004 Nexus Database Systems Pty Ltd.



8.5.3 Long Import Durations

Long Import Durations



The utilities create table with indexes if the source table has indexes defined. If the source tables contain a larger number of records the conversion of an index less copy and recreation of the indexes with the Enterprise Manager might be a better approach.

©2004 Nexus Database Systems Pty Ltd.



8.5.4 Long File Problems

Long File Problems



The different header signatures have a side effect on the used event log table "ImporterEvents". It should be deleted before the first call of a utility of a new beta version. Since it is created by code it can also be deleted before each run to save space and to minimize the number of records to be checked.

©2004 Nexus Database Systems Pty Ltd.



8.6 Implementation Details

Implementation Details



The following table lists the source code units of the project.

Unit	Specific Engines	Specific Wrapper	Remarks
fcnxiADOControlView	Microsoft ActiveX Data Objects	ADOExpress aka dbGO	Engine Control View
fcnxiBDEControlView	Borland Database Engine		Engine Control View
fcnxiDAOControlView	Microsoft Data Access Objects	KADao (Freeware)	Engine Control View
fcnxiFlashFilerControlView	FlashFiler 2.x		Engine Control View
fcnxiNexusControlView	Nexus DB 1.0		Engine Control View
fcnxiTransferManagerControlView			Transfer Manager Control View
fdnxImportProgress			Progress Dialog/Progress View
fdnxSetRange			Range Dialog
fmnxMain			Main Form (Specific Importers)
fmnxMainAll			Main Form (Combined Importer)
ftnxTemplateControlView			Engine Control View (Base Form)
ucnxiADOEngine	Microsoft ActiveX Data Objects	ADOExpress aka dbGO	Data Engine
ucnxiBasicDataEngine			Data Engine (Base Class)
ucnxiBasicDataMover			from Delphi TDataSet to Delphi TDataSet plus engine wrapped transactions
ucnxiBasicDefinitions			Structure Converter (Base Class)
ucnxiBasicStructureConverter			Data Engine
ucnxiBDEEngine	Borland Database Engine		Command Line Parser
ucnxiCommandLine			Data Engine
ucnxiDAOEngine	Microsoft Data Access Objects	KADao (Freeware)	Structure Converter (from FlashFiler to Nexus)
ucnxiDictionaryConverter	FlashFiler 2.x, Nexus DB 1.0		Structure Converter (from Delphi to FlashFiler)
ucnxiFlashFilerConverter	FlashFiler 2.x		Data Engine
ucnxiFlashFilerEngine	FlashFiler 2.x		Structure Converter (from Delphi to Nexus)
ucnxiNexusConverter	Nexus DB 1.0		Data Engine
ucnxiNexusEngine	Nexus DB 1.0		Transfer Manager
ucnxiTransferManager			
upnxiUtilities			
urnxiCommon			
urnxiConfiguration			

©2004 Nexus Database Systems Pty Ltd.



8.6.1 User Interface

8.6.1.1 Command Line Parameters

Command Line Parameters



A specific parser object manages the command line parameters. The classes which need their values store them in a set of "Default" parameters which are copied to the "View" set for usage in the control views and/or automatic import operations.

©2004 Nexus Database Systems Pty Ltd.



8.6.1.2 Graphical User Interface

Graphical User Interface



The specific importers all use the same main form, which contains a page control with four tabs.

1. Source Engine Parameters.
2. Destination Engine Parameters.
3. Transfer Manager Parameters
4. Dummy containing the startup message.

The combined importer uses a main form inherited from the previous one, which adds an additional tab sheet, which allows switching between the different engines.

The contents of the first three tab sheets are provided in form of a container form (called view in the above list), which is created by the active data engines or the transfer managers, which decouples the main form from the supported engines. The importer used the old approach of using forms as lightweight containers instead of frames because some components have problems with form inheritance and frame usage. In case of the importer suite, the FlashFiler components show this behavior, since they use a name base linking scheme, which does not allow you to reuse names across different forms in one application.

©2004 Nexus Database Systems Pty Ltd.



8.6.2 Importer Classes

8.6.2.1 Transfer Manager

Transfer Manager



The Transfer Manager is responsible for the program flow of the import operations. It gets its parameters either from the command line or via its control view.

©2004 Nexus Database Systems Pty Ltd.



8.6.2.2 Data Engines

Data Engines



The Data Engines provide a unified interface to the different database engines. Due to the lack of a common ancestor classes that provide both FieldDefs and IndexDefs properties, the TDataSet is the base class for the data transfer and all needed extensions have to be provided as methods.

©2004 Nexus Database Systems Pty Ltd.



8.6.2.3 Structure Converters

Structure Converters



The Structure Converter is responsible for the creation of a new Nexus table based on the table descriptions provided by the Delphi FieldDefs and IndexDefs structures. Therefore a TTable compatible wrapper is needed for each database engine of client to be used.

©2004 Nexus Database Systems Pty Ltd.



8.6.2.4 Data Movers

Data Movers



The Data Movers are responsible for the data transfer between the source tables and the generated new tables. The provided one copies the data between the TField objects of two Delphi TDataSet instances having the same name. For transaction control it uses methods of the used data engines.

©2004 Nexus Database Systems Pty Ltd.



8.6.3 How to add a new importer

How to add a new importer



For each new importer two classes are needed a data engine and its control view.

Thus the first step is to copy the BDE2NX project under a new name and then to rename the ucnxiBDEEngine and fcnxiBDEControlView units.

The second step is to replace the BDE specific components in the data engine by its compatible counter parts. This should be described in a chapter about adopting of a BDE application to the engine of your choice.

©2004 Nexus Database Systems Pty Ltd.



8.6.4 Using the Importer Classes in own applications

Using the Importer Classes in own applications



The importer classes can easily be used in own applications by using the CommandLineParser class as the central point of operation. Each parameter that can be set with the command line parser is a public property of the class, but in this case you have to read the methods, which deal with the macros to find out which ones to set.

So it is easier to create a parameter file with the standalone importers in interactive mode and then load it from code. Here's a simple example:

```

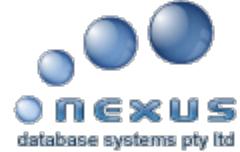
procedure
ImportFlashFilerFiles(const
ParameterFileName:
AnsiString);
begin
{ Clear command line parser
values }
    CommandLineParser.ClearValues;
{ Read parameter file }
    CommandLineParser.ProcessParameterFile(ParameterFileName);
{ Source engine:
FlashFiler }
    ucnxiFlashFilerEngine.CreateSourceEngine;
{ Destination engine:
Nexus }
    ucnxiNexusEngine.CreateDestinationEngine;
{ Structure converter:
Dictionary }
    ucnxiDictionaryConverter.CreateStructureConverter;
{ Data mover: Standard }
    ucnxiBasicDataMover.CreateDataMover;
{ Transfer Manager: Standard
}
    ucnxiTransferManager.CreateTransferManager;
{ Run }
    TransferManager.RunBatch;

{ Free used objects to avoid
problems with the
finalization sequence }
    FreeAndNil(DataMover);
    FreeAndNil(StructureConverter);
    FreeAndNil(DestinationEngine);
    FreeAndNil(SourceEngine);

```

```
FreeAndNil(TransferManager  
);  
end;
```

©2004 Nexus Database Systems Pty Ltd.



9 Introduction

Introduction



Welcome to the NexusDB Delphi Developer Reference Manual. This manual is organized into a number of sections which describe all aspects of the operation, maintenance and development of NexusDB applications written in Delphi.

This book should be seen as reference and Help for Delphi developers. If you want to know more about NexusDB concepts and Architecture please consult the according manual book.

If you should experience any issues with NexusDB we strongly encourage you to contact us via any of the methods listed on our webpage. If you feel you have uncovered an issue with NexusDB, then kindly ask for you to provide (where possible) a sample project demonstrating the issue you have found. This is to help us isolate and track down the issue to resolve it for you.

For the latest information on Nexus products be sure to regularly visit the website at www.nexusdb.com and, while you are there, add the various newsgroups (at news.nexusdb.com) to your reader.



10 Installation Guidelines

10.1 Embedded Server Version

Embedded Server Version



???todo



10.2 (Delphi) Developer Version

(Delphi) Developer Version



???todo



10.3 (Delphi) Enterprise Version

(Delphi) Enterprise Version



The installation guideline for the Enterprise Version of NexusDB V2 will be update when the product is available.



10.4 Trial Versions

Trial Versions



???todo



10.5 Upgrading from V1

Upgrading from V1



NexusDB is delivered as a source only installation to keep the download size for our customers as small as possible.

- Uninstall any previously installed NexusDB packages (for all Delphi Versions) on your machine
- Ensure that you have closed down any running instances of Delphi. (The installer detects this in order to be able to properly install the components into your IDE)
- Execute either the NexusDBSetup.exe or NexusTrialSetup.exe file
- The install wizard will then lead you through the installation in the usual fashion

This should now have successfully installed NexusDB into your version of Delphi and be ready for use in your projects.

???update



10.6 Recompiling the Packages and Binaries

Recompiling the Packages and Binaries



If you wish or have to recompile the various packages and/or the binaries please follow these instructions

- Remove all NexusDB packages (and possibly installed dependent packages) from the list of loaded packages. (Delphi menu: Components | installed package)
- Delete all package binaries (NexusDB*.bpl) from the machine. (This is optional and if you are sure that you have just one instance in the correct target directory you can skip this step)
- Start Delphi, close whatever project that might be loaded, and go to Open Project
- Select the Nexus2PackageX0.bpg package from the correct DelphiX subdirectory (x stands for your version of Delphi, for e.g., Nexus2Package70.bpg from the \Delphi7 subdirectory)
- After the project group is loaded please go to the Project menu and do a "Build all projects" - this will compile all the NexusDB packages
- After compilation open the Project Manager (CTRL-ALT-F11) and select the Nexus200dv70.dpk project
- Right mouse click over this project to bring up the popup menu and select "Install"
- A message box should appear to inform you that NexusDB components have been added to the NexusDB palette tab
- Select File/Close All (save changes)
- Due to a bug in Delphi, you should now close down and restart Delphi. If you do not do this, then the executables in the next step will be built with runtime packages, even though the project options specify otherwise.
- Open Project Nexus2ApplicationsX0.bpg from the same directory as step 7 above
- Again do a Project/Build All to create Server, Enterprise Manager and Tools executables which should end up in the <install directory>\Bin subfolder.



11 VCL Component Overview

VCL Component Overview



This section contains a listing of all the components available on the NexusDB VCL component tab. Most of the components are grouped here in a logical manner according to usage and functionality. For instance, all the transports and command handlers are grouped together. There are, however, some properties and events common to a number of components. Some of these are discussed in the Common Attributes section.

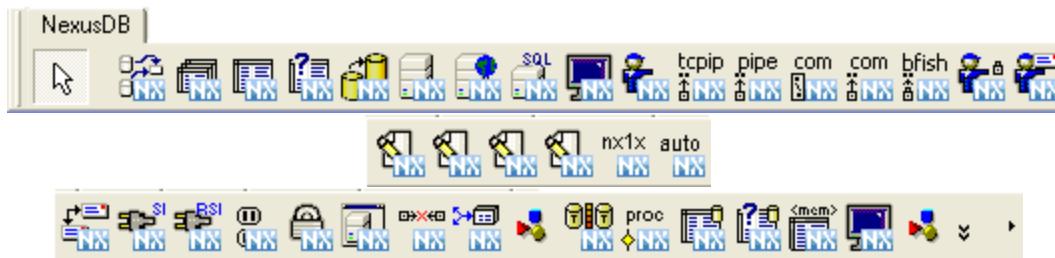


11.1 Components on the VCL Palette

Components on the VCL Palette



When the NexusDB design time package is installed into the IDE the palette contains a tab that will look like this.



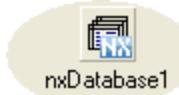
Here's a short description of what each component is and what its place is in NexusDB. All components and their key properties are described in the following sub sections. If you're looking for a full reference please see the Source Reference book.

TnxSession



The NexusDB session component. Required component to connect your database/table/query components to a remote or embedded server engine component.

TnxDatabase



The NexusDB equivalent of TDatabase. Provides alias and database path handling. Not required, although for manageability reasons it's recommended to use one.

TnxTable



The NexusDB equivalent of TTable. Used for navigational access to tables by a NexusDB server.

TnxQuery



The NexusDB TQuery equivalent. Extensive support for ANSI SQL/DDL provides strong access and data manipulating capabilities.

TnxBackupController



The backup controller allows backups with full data integrity at any time, even if the server is running 24/7.

TnxServerEngine



The server engine contains the core NexusDB engine. Drop one on a form or data module in your application to create a single user database application. Connect with transports, SQLEngine etc to create custom stand-alone database servers.

TnxRemoteServerEngine



Client-side server engine. Together with a transport, it provides remote access to data on a stand-alone server.

TnxSQLEngine



SQL engine Required for single-user applications that use SQL statements (TnxQuery components), or in standalone servers.

TnxSimpleMonitor



Used to develop server-side modules that monitors events (table open, record update, etc) and performs actions based on those events. Use to extend the server functionality.

TnxServerCommandHandler



The server command handler is responsible for decoding client requests, translation and forwarding of this request to the server engine and issuing the results back to the client.

TnxWinsockTransport



This component connects two machines via TCP/IP (Winsock). At least one of the transports is required to create a client application that connects to a remote server.

TnxNamedPipeTransport



This component connects two machines via Named Pipes. At least one of the transports is required to create a client application that connects to a remote server.

TnxCustomComTransport



This component is intended for in-process use (EXE and DLLs), or where direct RPC calls can be performed.

TnxRegisteredComTransport



This component connects client and server processes on the same machine. It does not require that any network is installed.

TnxBlowfishRC4SecuredTransport



This is a wrapper transport that can be used to make other transports (TCP, Named Pipes etc) secure.

TnxSecuredCommandHandler



???todo

TnxSimpleCommandHandler



Command handler that does not depend on any database components. Use if you need to create a message system or similar that doesn't require database functionality.

TnxSimpleSession



Session component that does not depend on any database components. Use if you need to create a message system or similar that doesn't require database functionality.

TnxServerInfoPlugin



Server-side plug-in example component that implements a few nice to have utility functions (get server-side time, get unique server ID).

TnxRemoteServerInfoPlugin



The client-side plug-in example component for accessing the TnxServerInfoPlugin functions from a remote client.

TnxServerInfoPluginCommandHandler



nxServerInfoPluginCommandHandler1

Use this on the server side to route messages from remote clients to the TnxServerInfoPlugin.

TnxSecurityMonitor



nxSecurityMonitor1

???todo

TnxServerManager



nxServerManager1

This component manages a server engine (and all its attached server modules) and their settings.

TnxCustomConnectionFilter



nxCustomConnectionFilter1

???todo

TnxDatabaseMapper



nxDatabaseMapper1

???todo

TnxSessionPool



nxSessionPool1

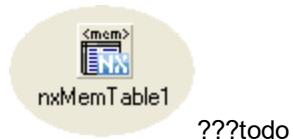
???todo

TnxTransContext



nxTransContext1

???todo

TnxStoredProc**TnxCachedDataSet****TnxSqlUpdateObject****TnxMemTable****TnxSqlTriggerMonitor****TnxSharedMemoryTransport****TnxEventLog**

TnxEventLogDispatcher



The event log is a centralized way for handling log entries. All the other components can be connected to a single event log.

TnxEventBasedLog



???todo

TnxWinEventLog



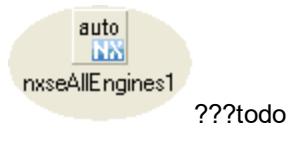
???todo

Tnx1xAllEngines



???todo

TnxSeAllEngines



???todo



11.2 Common Properties and Events

Common Properties and Events



The properties and events listed below are used in most of the components with essentially the same meaning (with only slight variations for some components).

Most of the NexusDB components are stateful components. Consequently, nearly all have the category and display name properties and the state events listed below.

Key Properties

ActiveDesignTime	Component's Active property will be set to this value at design time under the IDE. For the server engine component only, if this is set to true, then it will be set temporarily to false if the app is run under the IDE.
ActiveRunTime	Component's Active property will be set to this value at runtime.
DisplayCategory	Used by the nxServer interface to show where in the tree-view to put the component. (See nxServer.exe)
DisplayName	This is the name to be used in the tree-view for this component.
Enabled	Enables/disables the component.
EventLog	Connects to a TnxEventLog. Log events will be written to the file specified in the attached TnxEventLog component.
EventLogEnabled	Enables/disables the writing of log information if a TnxEventLog component is attached.

Events

You would rarely use these state events. They are included here merely so that you are aware of them and that you should not, in the normal course of a project, need to hook into them. Certainly only for the most advanced of users.

OnStateChanged	Allows you to hook into when the State Engine of the component has changed.
OnStateTransChanged	Allows the user to hook even deeper into the actual change in a state transition.



11.3 Server Engines and Monitors

11.3.1 TnxSQLEngine

TnxSQLEngine



Supported in products: All

An SQL Engine component must be placed on the main form with a TnxServerEngine for any embedded server application that uses SQL. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
DisplayCategory	
DisplayName	nxSqlEngine1
Enabled	True
EventLog	
EventLogEnabled	False
Name	nxSqlEngine1
Tag	0
Version	2.0000 Public Beta 4 (D7)

Event	Description
EventLog	
OnStateChange	
OnStateTransC	

Purpose

The TnxSQLEngine component is required to execute any query (as well as a TnxServerEngine). It is not required, however, when using a Remote Server. When placed on the form, the TnxSQLEngine component is linked to a server engine with the link being from the server engine component.

If this component is omitted, a "capability not supported" error message will occur when trying to execute a query. This will only occur with embedded server applications as nxServer has a correctly connected TnxSQLEngine component.

Please also see Common Properties and Events for more details.



11.3.2 TnxServerEngine

TnxServerEngine



Supported in products: All

A Server Engine component must be placed on the main form for any application using a "local" (embedded) server. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Two screenshots of the VCL Object Inspector dialog box, both titled "Object Inspector" and showing "nxServerEngine1 TnxServerEngine".

Left window (Properties tab):

Name	nxServerEngine1
Options	[]
ServerName	
SqlEngine	
TableExtension	nx1
Tag	0
TempStorePath	
TempStoreSize	-1
Version	2.0000 Public Beta 4 (D7)

Right window (Events tab):

EventLog	
OnStateChange	
OnStateTransC	
SqlEngine	

Purpose

The TnxServerEngine component contains all of the database server functionality. It is only required on the form, however, when using an embedded server or creating a custom server. To create a database application, either a TnxServerEngine or TnxRemoteServerEngine component needs to be placed on the form.

The TnxServerEngine is responsible for all navigational database access and manipulation as well as opening and closing files, folders etc. Any access to the database using queries, needs an additional TnxSQLEngine included and linked to via the SQLEngine property.

Only one TnxServerEngine or TnxRemoteServerEngine component is needed per application, regardless of how many threads are used. These engines are threadsafe for multithreaded applications.

Key Properties

MaxRAM	The maximum amount of RAM to be reserved for use by the buffer manager. If this value is set to 0 or -1, then half the available physical RAM will be used. Otherwise, the amount specified, in MegaBytes, will be used.
Options	???todo
SQLEngine	Connect to a TnxSQLEngine component. This is necessary if you are using a TnxQuery component in your single exe application.
TempStorePath	System path used for the internal temporary storage. If blank, then the user system default temp path is used.
TempStoreSize	This is the size, in MegaBytes, of the temporary storage area. If blank or -1, then it is set to the minimum of (a) 10 times MaxRAM, (b) one-third the available disk space, or (c) one-third the maximum possible file size of the filesystem. Temp storage is used by the buffer manager if RAM requirements exceed the value of MaxRAM. This is likely to happen if you use large SQL's or very big transactions. In such cases make sure that adequate temp storage has been allocated. It is strongly recommended that you ensure enough space has been set aside for the first two categories.

Note: The temporary storage files are created with a filesystem flag that makes the OS delete them when the creating process terminates. If you have unexpected shutdowns of the machine, OS or your programs, then the temporary storage files might be left on disk and not cleaned up. To see where the temp path of a windows system is, and what files are stored there, issue the following at a command prompt:

```
DIR /AH %TEMP%
```

Please also see Common Properties and Events for more details.



11.3.3 TnxServerManager

TnxServerManager



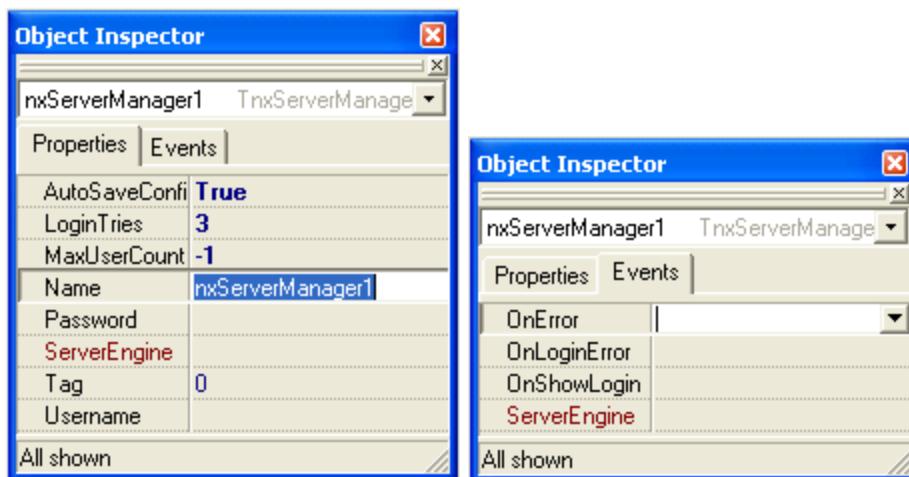
Supported in products: Developer Edition.

NexusDB Server Manager component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

???todo

Key Properties

AutoSaveConfig	???todo
LoginTries	???todo
Password	???todo
ServerEngine	???todo
Username	???todo

Key Events

OnError	???todo
---------	---------

OnLoginError	???todo
OnShowLogin	???todo



11.3.4 Tnx1xAllEngines and TnxseAllEngines

Tnx1xAllEngines and TnxseAllEngines



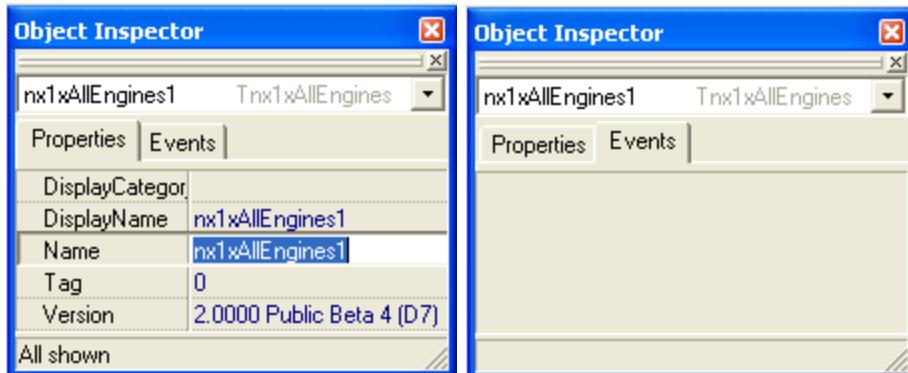
Supported in products: All

Sub Engine units component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The TnxServerEngine uses the sub-engines contained in this unit, and it must therefore be included so that these sub-engines are linked into the application. To achieve this, simply place a Tnx1AllEngines component on the form or, alternatively, include nx1AllEngines in the Uses clause of the unit containing your server engine component.



11.3.5 TnxPascalScriptEngine

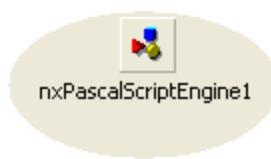
TnxPascalScriptEngine



Supported in products: All.

NexusDB Pascal Script Engine component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Two side-by-side Object Inspector windows for the component 'nxPascalScriptEngine1'. Both windows show the 'Properties' tab selected. The left window shows the full list of properties: DisplayCategory, DisplayName, EventLog, EventLogEnabled, Name (which is selected and highlighted in blue), Tag, and Version. The right window shows the 'Events' tab, which is currently empty.

Purpose



11.3.6 TnxServerScriptEngine

TnxServerScriptEngine



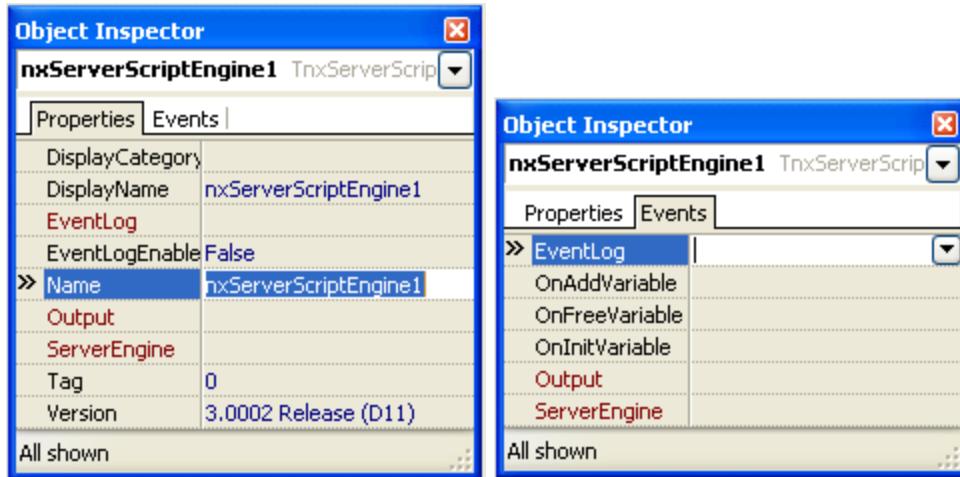
Supported in products: All.

NexusDB Server Script Engine component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose



11.3.7 TnxSimpleMonitor

TnxSimpleMonitor



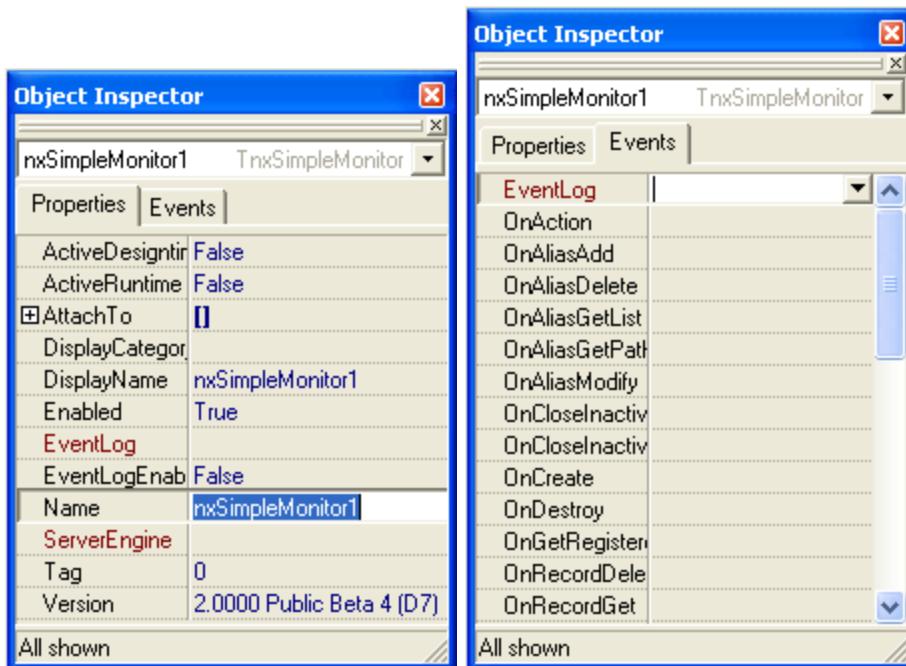
Supported in products: Developer Edition, Embedded Edition.

NexusDB Simple Monitor component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

When attached to the server engine, the TnxSimpleMonitor can be notified of all events in the server. This can be used, for example, to limit the number of connections made to the server. Each event is called at least once and potentially twice when the relevant method is called. Each event takes a Boolean, aBefore, indicating whether the method is being implemented before or after the action takes place. For example, if OnAliasAdd is implemented, it will be run before the AliasAdd action takes place. This enables any actions to be aborted by raising an exception. On successful completion of the AliasAdd method, OnAliasAdd will be called again and is generally used for informational purposes only.

Key Properties

AttachTo	Use this event to tell the monitor to connect to session, databases, cursors or any combination of these.
ServerEngine	Connect this to the a TnxServerEngine component.

Key Events

OnAction	This event is raised for all actions and tells you what occurred. Duplicates all the other individual events.
----------	---

Please also see Common Properties and Events for more details.



11.3.8 TnxSQLTriggerMonitor

TnxSQLTriggerMonitor



Supported in products: Developer Edition, Embedded Edition.

??? todo

Icon



Properties and Events view

Object Inspector	
nxSqlTriggerMonitor1 TnxSqlTriggerMonit	
Properties Events	
ActiveDesigntime	False
ActiveRuntime	False
DisplayCategory	SQL Engine
DisplayName	SQL Triggers
Enabled	True
EventLog	
EventLogEnabled	False
Name	nxSqlTriggerMonitor1
ServerEngine	
SqlEngine	
Tag	0
Version	2.0000 Public Beta 4 (D7)
All shown	

Object Inspector	
nxSqlTriggerMonitor1 TnxSqlTriggerMonit	
Properties Events	
EventLog	
OnStateChanged	
OnStateTransChar	
ServerEngine	
SqlEngine	
All shown	

Purpose

Please also see Common Properties and Events for more details.



11.3.9 TnxSecurity Monitor

TnxSecurity Monitor



Supported in products: All

NexusDB Security Monitor component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Object Inspector	
nxSecurityMonitor1 TnxSecurityMonito ▾	
Properties	Events
ActiveDesignTime	False
ActiveRuntime	False
AlwaysLoginAd	False
DisplayCategory	
DisplayName	nxSecurityMonitor1
Enabled	True
EventLog	
EventLogEnabled	False
MaxSessionCount	-1
Name	nxSecurityMonitor1
ServerEngine	
Tag	0
Version	2.0000 Public Beta 4 (D7)
All shown	

Object Inspector	
nxSecurityMonitor1 TnxSecurityMonito ▾	
Properties	Events
EventLog	
OnStateChange	
OnStateTransC	
ServerEngine	
All shown	

Purpose

The TnxSecurityMonitor links directly to the server engine and is used to authenticate connections (from a session component) and apply access restrictions. These restrictions are issued on a per user, per server basis. Access tokens available are 'amend', 'read' and 'write'. Granting write access

alone will allow the user to enter data but all tables will appear empty. If no TnxSecurityMonitor is used in an application, there will be no security and all rights are granted to every user.

Key Properties

ServerEngine	Assign a TnxServerEngine component to this property.
--------------	--

Please also see Common Properties and Events for more details.



11.4 Transport Group

Transport Group



The transport components are components of a high-level message orientated communication system. All components have a mode (listen mode), which defines if this transport is the server (listen) or client (send). All transport components are threadsafe.



11.4.1 Common Transport Properties and Events

Common Transport Properties and Events



Transports operate synchronously; that is, on the client side they send a request (message) and then block and waits for a reply. On the server side all incoming requests (messages) are handed off to a command handler, which then processes the message and generates a reply (return message). This return message is then passed back to the transport for transmission back to the originating client.

Key Properties

CommandHandler	Used if the transport receives any messages. If the client side has callbacks then all the incoming messages are sent off to the attached command handler.
MaxBytesPerSecond	Limits the bandwidth the transport can send per second. Very useful for testing purposes.
Mode	Mode can be either listen or send. It depends on whether its client or server side.

PingTime	This is to limit the transport for testing purposes and is a value in milliseconds used to simulate ping times.
RespondToBroadcasts	If true the transport answers to broadcasts from the set server.
ServerNameDesignTime	Name of the server to connect to at design time.
ServerNameRunTime	Name of the server to connect to at run time.
Timeout	Value, in milliseconds, used when sending a broadcast.

Key Events

OnAddSession	Called whenever a new session is added to the transport on the server side.
OnChooseServer	This is raised when the broadcast finds more than one available server.
OnConnectionLost	This event is not implemented yet.
OnFindServers	This event is raised twice. First when it begins to look for servers and then when it has finished looking for servers. This allows you to show a dialog box or similar to the end user.
OnRemoveSession	This event is called when a session is removed on the server side of the transport.

Please also see Common Properties and Events for even more details.



11.4.2 TnxCustomCOMTransport

TnxCustomCOMTransport



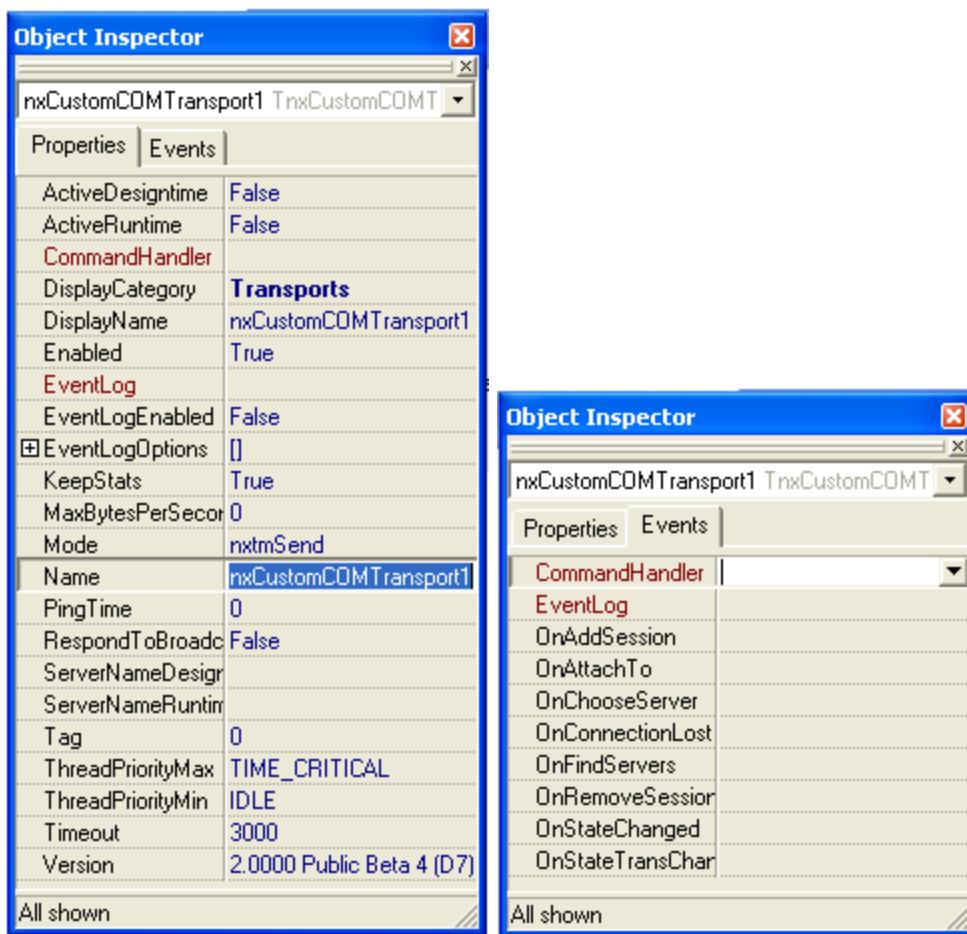
Supported in products: All

NexusDB Custom COM transport component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The servername property for the TnxCustomCOMTransport component has no meaning. The TnxCustomCOMTransport.OnAttachTo event needs to be implemented and generally connects to the AttachRemoteTransport method of the server transport. This transport does not support broadcasts and connections are contained inside a single process. A callback thread count is not needed, as communication is bi-directional.

Key Events

OnAttachTo	If the Custom COM Transport is in send mode then when this event is called, it is your responsibility to pass your TnxTransport to the server side and then process the TnxTransport object returned.
------------	---

Please also see Common Transport Properties and Events for more details.



11.4.3 TnxRegisteredCOMTrasnsport

TnxRegisteredCOMTrasnsport



Supported in products: All

NexusDB Registered COM transport component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Object Inspector	
nxRegisteredCOMTransport1 TnxRegistered	
Properties	Events
ActiveDesignTime	False
ActiveRuntime	False
CommandHandler	
DisplayCategory	Transports
DisplayName	nxRegisteredCOMTransport1
Enabled	True
EventLog	
EventLogEnabled	False
EventLogOptions	[]
KeepStats	True
MaxBytesPerSecond	0
Mode	nxtmSend
Name	RegisteredCOMTransport1
PingTime	0
RespondToBroadcast	False
ServerNameDesignTime	
ServerNameRuntime	
Tag	0
ThreadPriorityMax	TIME_CRITICAL
ThreadPriorityMin	IDLE
Timeout	3000
Version	2.0000 Public Beta 4 (D7)
All shown	

Object Inspector	
nxRegisteredCOMTransport1 TnxRegistered	
Properties	Events
CommandHandler	
EventLog	
OnAddSession	
OnChooseServer	
OnConnectionLost	
OnFindServers	
OnRemoveSession	
OnStateChanged	
OnStateTransChar	
All shown	

Purpose

The TnxRegisteredCOMTransport is similar to the TnxCustomCOMTransport with a few additions. Firstly, the servername property needs to contain the same value in both server-side and client-side

transports. These transports must also reside on the same computer. Secondly, the TnxRegisteredCOMTransport component supports broadcasts as the computer contains a list of registered COM transports.

Please also see Common Transport Properties and Events for more details.



11.4.4 Common to Winsock and Named Pipe Transports

Common to Winsock and Named Pipe Transports



The most commonly used transport, TnxWinsockTransport represents a TCP/IP v4 connection over a network. The servername property is the DNS or alternatively takes the form `servername@ip_address` where `servername@` is optional. Only one server can be assigned to a single port. The Winsock transport can also be connected to a connection filter component.

For a TnxNamedPipeTransport the servername property takes the value `servername@computername`, where `@computername` is optional and defaults to the current computer if not specified. This transport is only available for use on NT based systems and is dependent on the security settings of the operating system.

The TnxNamedPipeTransport provides better performance than TnxWinsockTransport on a single computer, but slower over a network.

Both of these transports are implemented using overlapped IO and IO completion ports to minimise the number of threads required and thread context switches to process incoming requests.

The CallbackThreadCount property defaults to 0 and must be set > 0 to enable messages to be sent from the server back to the client. This property defines the number of concurrent callbacks that can be processed.

Key Properties

BroadcastThreadPriority	Priority of the thread that answers broadcasts. Recommended to leave it alone. Default value (tpNormal) is a good balanced value.
CallbackThreadCount	Set on the client side. If set to greater than 0 then callbacks are possible. The value set is the number of concurrent threads on the client side. If the server has more callbacks than this value, it holds on to them until the number of threads on the client side is reduced.
CallbackThreadPriority	Priority of the thread that sends broadcasts. Recommended to leave it alone. Default value (tpNormal) is an excellent choice.
CompressLimit	This is the minimum message size in bytes before compression kicks in.
CompressType	This is the compression engine used. The number here represents the algorithm implemented.

	<p>0 = no compression. 1-9 are various zip compressions. 10 is the well-known RLE algorithm.</p> <p>If you want to use this, 1 is a good default to use over slow connections. Please note that for the compression to work, you need to include the actual implementation units (<code>nxlZipCompressor</code>, <code>nxlRleCompressor</code>) in the uses clause of both server and client (it is already compiled into the default server).</p>
ConcurrentIOPCThreads	This is the number of threads on the server side processing client requests. If set to 0 then it defaults to the number of CPU's on the server.
HeartbeatInterval	This is the number of milliseconds between sending heartbeat messages to the server from the client side.
HeartbeatThreadPriority	This is the background thread priority that wakes up every HeartbeatInterval to send a message to the server.
Port	Standard TCP/IP port number. Default is arbitrarily chosen as 16000. Must be the same on the server and the clients.
ServerThreadPriority	Priority of the server thread. Recommended to leave it alone. Default value (<code>tpNormal</code>) is not going to get you into too much trouble.
WatchdogInterval	This is the number of milliseconds between checking for heartbeat messages received on the server from clients. Checks all connections and if 2 heartbeats have gone missing for a client then that connection is declared dead and tidied up.
WatchdogThreadPriority	Priority of the server watchdog thread. Recommended to leave it alone. Default value (<code>tpHighest</code>) is a good value.

Please also see Common Transport Properties and Events for more details.



11.4.5 TnxSharedMemoryTransport

TnxSharedMemoryTransport



Supported in products: All

??? todo

Icon



Properties and Events view

The image displays two side-by-side windows of the 'Object Inspector' tool, both titled 'nxSharedMemoryTransport1 TnxSharedMem'. The left window shows the 'Properties' tab, which lists numerous configuration settings for the transport component. The right window shows the 'Events' tab, which lists various event handlers.

Properties Tab (Left Window):

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
CallbackThreadCount	0
CallbackThreadPriority	tpNormal
CommandHandler	
CompressLimit	512
CompressType	0
ConcurrentLOCPTCount	0
DisplayCategory	Transports
DisplayName	nxSharedMemoryTransport1
Enabled	True
EventLog	
EventLogEnabled	False
EventLogOptions	[]
HeartbeatInterval	10000
HeartbeatThreadPriority	tpTimeCritical
KeepStats	True
MaxBytesPerSecond	0
Mode	nxtmSend
Name	redMemoryTransport1
OverlappedClient	False
PingTime	0
Port	16000
RespondToBroadcast	False
ServerNameDesignTime	
ServerNameRuntime	
ServerThreadPriority	tpNormal
Tag	0
ThreadPriorityMax	TIME_CRITICAL
ThreadPriorityMin	IDLE
Timeout	3000
Version	2.0000 Public Beta 4 (D7)
WatchdogInterval	10000
WatchdogThreadPriority	tpHighest

Events Tab (Right Window):

Event	Description
CommandHandler	
EventLog	
OnAddSession	
OnChooseServer	
OnConnectionLost	
OnFindServers	
OnRemoveSession	
OnStateChanged	
OnStateTransChar	

Purpose

Please also see Common to Winsock and Named Pipe Transports for more details.



11.4.6 TnxWinsockTransport

TnxWinsockTransport



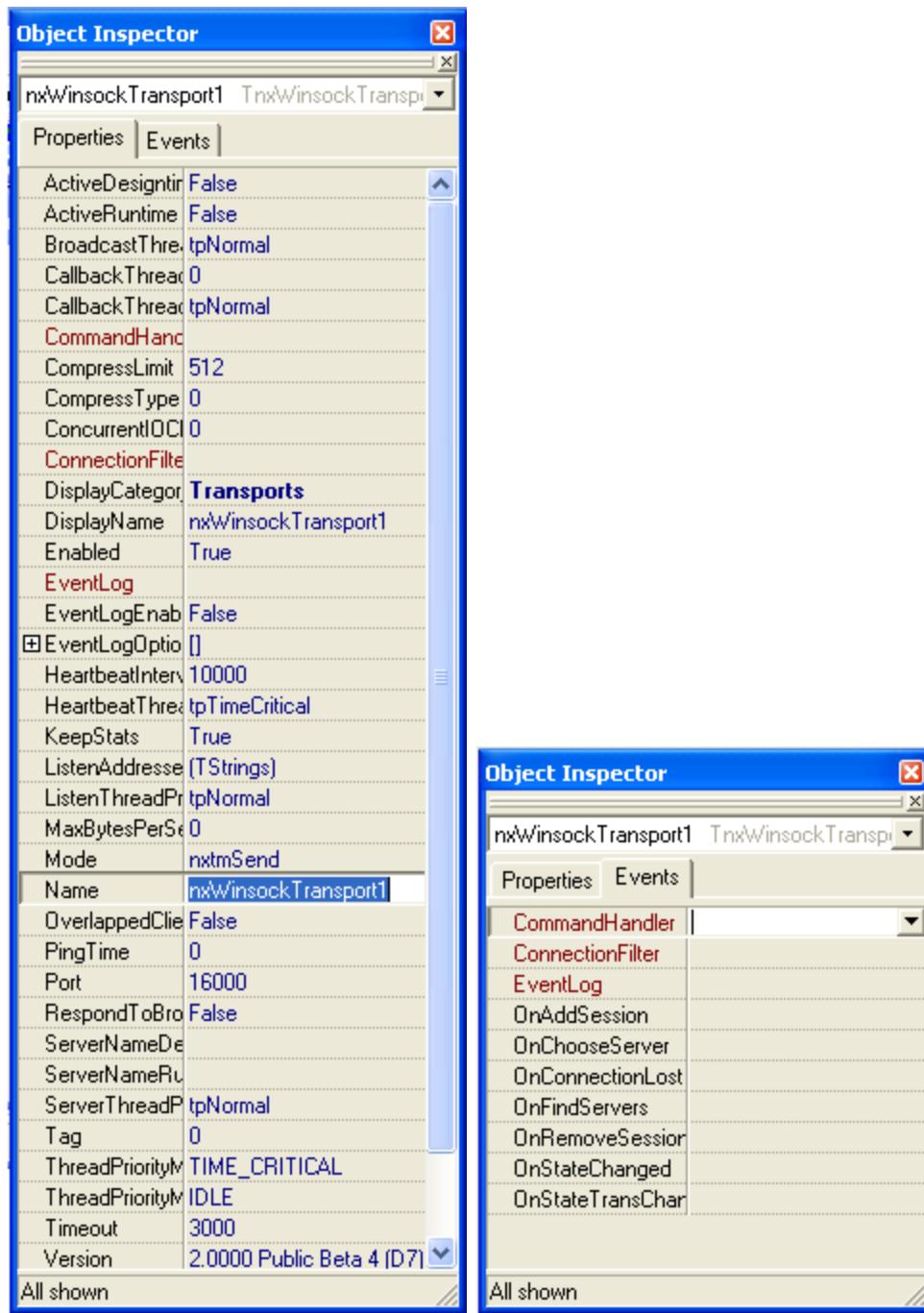
Supported in products: Developer Edition, BDE Replacement Edition.

NexusDB TCP/IP transport component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The most commonly used transport, TnxWinsockTransport represents a TCP/IP v4 connection over a network. The `servername` property is the DNS or alternatively takes the form `servername@ip_address` where `servername@` is optional. Only one server can be assigned to a single port. The Winsock transport can also be connected to a connection filter component.

Key Properties

ConnectionFilter	Connect to a TnxConnectionFilter component to filter out unwanted client connections.
ListenThreadPriority	Priority of the thread that listens for broadcasts. Recommended to leave it alone. Default value (tpNormal) is really quite ok.

Please also see Common to Winsock and Named Pipe Transports for more details.



11.4.7 TnxNamedPipeTransport

TnxNamedPipeTransport



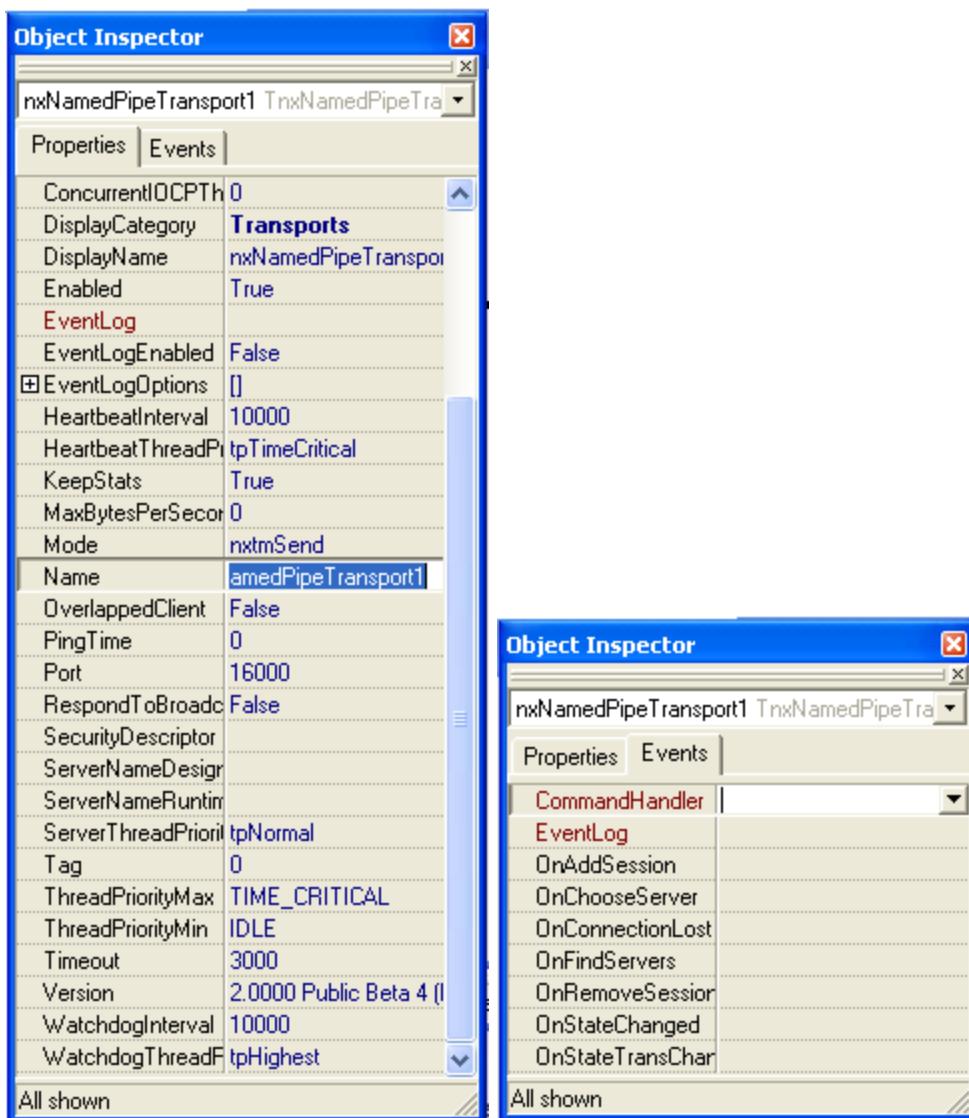
Supported in products: All.

NexusDB Named Pipe transport component. Default values are shown in the object inspector snapshot.

Icon



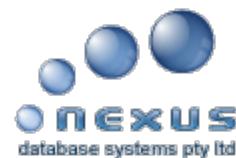
Properties and Events view



Purpose

Uses the MS Windows Named Pipe transport to connect to a server. Only available server side on NT based machines. It is faster than Winsock if your application is on the server machine, but normally slower than Winsock on a LAN.

Please also see Common to Winsock and Named Pipe Transports for more details.



11.4.8 TnxBlowfishRC4SecuredTransport

TnxBlowfishRC4SecuredTransport



Supported in products: All.

NexusDB Blowfish RC4 Secured transport component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

The image displays two windows of the Object Inspector for the component 'nxBlowfishRC4SecuredTransport1'. The left window shows the 'Properties' tab, and the right window shows the 'Events' tab.

Properties Tab (Left Window):

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
CommandHandler	
DisplayCategory	Transports, Secured
DisplayName	nxBlowfishRC4SecuredTransport1
Enabled	True
EventLog	
EventLogEnabled	False
EventLogOptions	[]
KeepStats	True
Key	
MaxBytesPerSecond	0
Mode	nxlmSend
Name	C4SecuredTransport1
PingTime	0
RespondToBroadcast	False
ServerNameDesign	
ServerNameRuntime	
Tag	0
ThreadPriorityMax	TIME_CRITICAL
ThreadPriorityMin	IDLE
Timeout	3000
Transport	
Version	2.0000 Public Beta 4 (D7)

Events Tab (Right Window):

Event
CommandHandler
EventLog
OnAddSession
OnChooseServer
OnConnectionLost
OnFindServers
OnRemoveSession
OnStateChanged
OnStateTransChar
Transport

Purpose

he TnxBlowfishRC4SecuredTransport is a secure transport that uses the Blowfish RC4 algorithm with a static key, which must match on the client and server side. This transport must be used in conjunction with the secure command handler and other transport as shown below:



Key Properties:

Key	This is the key used (a string) by the encryption engine. It must be the same on both client and server side or garbage ensues!
Transport	As shown in the diagram above, connect to an appropriate TnxTransport component. (Usually the Winsock transport.)

Please also see Common Transport Properties and Events for more details.



11.5 Servers and Command Handlers

11.5.1 TnxServerCommandHandler

TnxServerCommandHandler



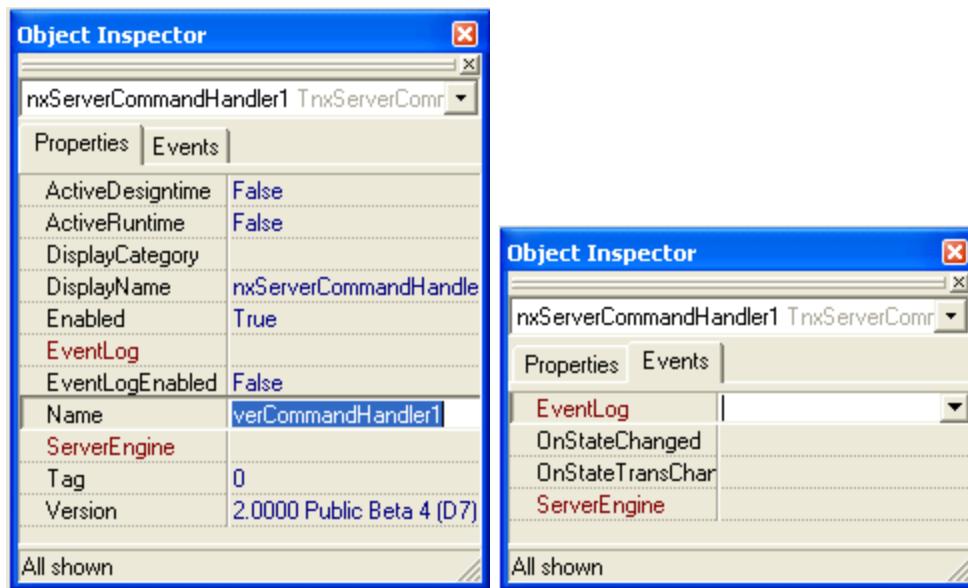
Supported in products: Developer Edition, Embedded Edition.

NexusDB Server Command Handler component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The TnxServerCommandHandler component must be linked to either a TnxServerEngine or TnxRemoteServerEngine component. Any number of transports can be linked to the command handler, which receives messages from the transport and translates it back into method calls to pass to the server engine. Any replies from the server will be passed back through the command handler to the transport.

If an unrecognised message is received, the server command handler will send the message to all attached plugin command handlers looking for a match.

Key Properties

ServerEngine	Connect to a TnxServerEngine component.
--------------	---

Please also see Common Properties and Events for more details.



11.5.2 TnxSecuredCommandHandler

TnxSecuredCommandHandler



Supported in products: All.

NexusDB Secured Command Handler component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Object Inspector	
nxSecuredCommandHandler1 TnxSecuredC	
Properties Events	
ActiveDesigntime	False
ActiveRuntime	False
DisplayCategory	
DisplayName	nxSecuredCommandHandler1
Enabled	True
EventLog	
EventLogEnabled	False
Name	redCommandHandler1
SecuredTransport	
Tag	0
Version	2.0000 Public Beta 4 (D7)
All shown	

Object Inspector	
nxSecuredCommandHandler1 TnxSecuredC	
Properties Events	
EventLog	
OnStateChanged	
OnStateTransChar	
SecuredTransport	
All shown	

Purpose

Part of the secured transport system, a TnxSecuredCommandHandler component is needed as a link between a secure transport component and standard transport. See the diagram under the RC4 component description above.

Key Properties

SecuredTransport	Connect to a secured transport component.
------------------	---

Please also see Common Properties and Events for more details.



11.5.3 TnxSimpleCommandHandler

TnxSimpleCommandHandler



Supported in products: Developer Edition.

NexusDB Secured Command Handler component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
DisplayCategory	
DisplayName	nxSimpleCommandHandle
Enabled	True
EventLog	
EventLogEnabled	False
Name	simpleCommandHandler1
Tag	0
Version	2.0000 Public Beta 4 (D7)

Event	Description
EventLog	
OnStateChanged	
OnStateTransChar	

Purpose

A TnxSimpleCommandHandler receives messages from a transport component but has no dependency on database related code and therefore, cannot process any messages. This component can be used to receive messages from a transport and pass them on to all plugin command handlers for processing. The TnxSimpleSession must be used in place of the database aware TnxSession component.

Please also see Common Properties and Events for more details.



11.5.4 TnxCustomConnectionFilter

TnxCustomConnectionFilter



Supported in products: All.

NexusDB Custom Connection Filter component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
DisplayCategory	
DisplayName	nxCustomConnectionFilter1
Enabled	True
EventLog	
EventLogEnabled	False
Name	nxCustomConnectionFilter1
Tag	0
Version	2.0000 Public Beta 4 (D7)

Event	Description
OnAcceptConnection	
OnStateChanged	
OnStateTransChar	

Purpose

This component is used in conjunction with the Winsock transport. It is used to prevent certain connections being made to the server. For instance, if the server is connected to the internet, you could allow only certain clients to connect based on the logic encoded in the OnAcceptConnection method.

Key Events

OnAcceptConnection	For all physical incoming connections, this event is raised containing ip, port, local, remote and other connection information. Using this data, the connection request can be accepted or rejected based on the filter settings.
--------------------	--

Please also see Common Properties and Events for more details.



11.5.5 TnxRemoteServerEngine

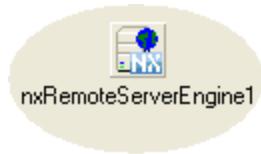
TnxRemoteServerEngine



Supported in products: Developer Edition, BDE Replacement Edition.

Remote Server Engine component. Must be placed on the main form with a transport component for any client application to access a Nexus Server. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Two side-by-side 'Object Inspector' windows for the component 'nxRemoteServerEngine1'.

Left Window (Properties View):

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
CacheAliases	False
DisplayCategory	
DisplayName	nxRemoteServerEngine1
Enabled	True
EventLog	
EventLogEnabled	False
Name	remoteServerEngine1
RemoteThreadPrio	NORMAL
Tag	0
Transport	
Version	2.0000 Public Beta 4 (D7)

Right Window (Events View):

Event	Description
EventLog	
OnStateChanged	
OnStateTransChar	
Transport	

Purpose

A TnxRemoteServerEngine component is the local (client side) representation of a remote TnxServerEngine. It must be connected to a transport component so that it can communicate with a separate server application running either on the same computer or over a network. The TnxRemoteServerEngine receives method calls and translates them into messages/requests passed

to the transport and eventually to the running server. It can also receive replies from the transport if needed.

Key Properties

CacheAliases	If set to true then the alias list from the server is loaded client side when a connection is first established. This will speed up operations. However, if the server changes any alias information, then the client is unaware of such changes until the next time it connects.
Transport	Must be connected to a transport component to connect to a remote server.

Please also see Common Properties and Events for more details.



11.6 Sessions and Databases

11.6.1 TnxSessionPool

TnxSessionPool



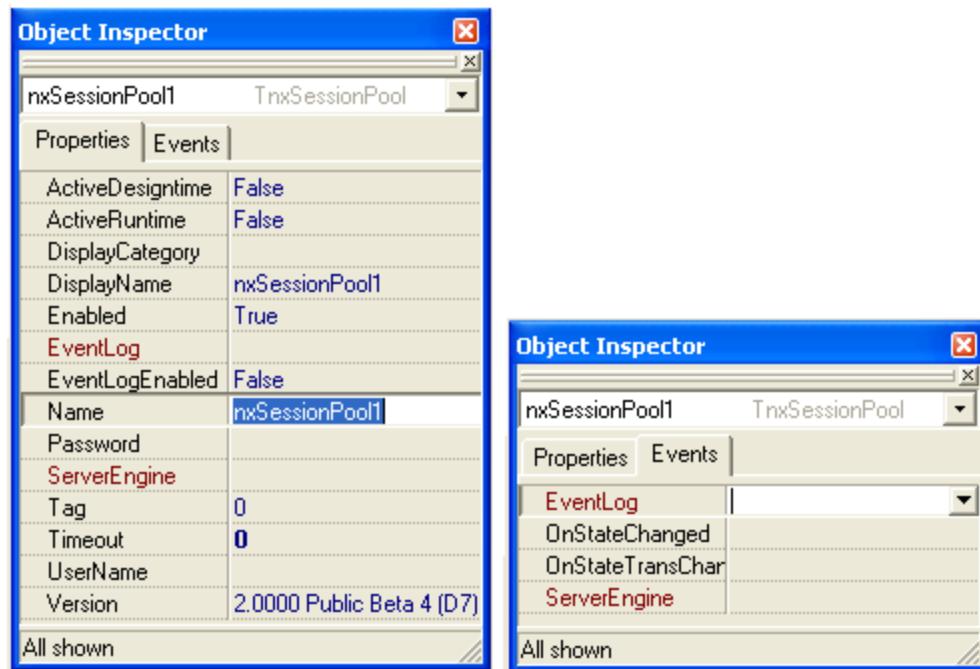
Supported in products: Developer Edition, Embedded Edition.

??? todo

Icon



Properties and Events view



Purpose

Please also see Common Properties and Events for more details.



11.6.2 TnxSession

TnxSession



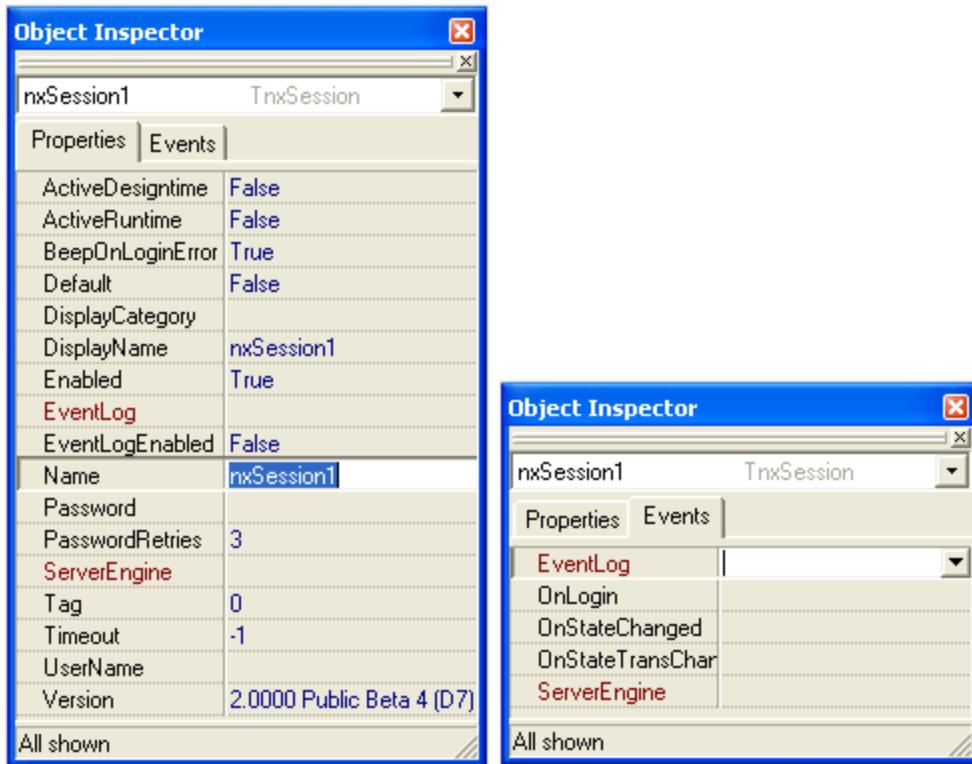
Supported in products: All.

NexusDB Session component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

A session defines the security context for a database connection. As such, a TrxSession component is required for each thread. The TrxSession component must link to either a server engine or remote server engine and is required for all database applications. Multiple sessions are needed to implement several simultaneous logins. Login usernames and passwords, if used, are defined in the TrxSecurityMonitor, which contains secure server functionality.

Key Properties

BeepOnLoginError	Makes noise on a login failure.
Password	Set to a password that will be recognized by the server for the set username.
PasswordRetries	How often it brings up the password dialog and retries login with the server before it gives up and raises an error.
ServerEngine	Connect to a TrxServerEngine component.
UserName	Set to a username recognized by the server.

Key Events

OnLogin	Allows you to show your own custom login dialog.
---------	--

Please also see Common Properties and Events for more details.



11.6.3 TnxDatabase

TnxDatabase



Supported in products: All.

NexusDB database component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Two side-by-side "Object Inspector" windows for the component "nxDatabase1".
Left Window (Properties View):

Property	Value
ActiveDesignime	False
ActiveRuntime	False
AliasName	
AliasPath	
Default	False
DisplayCategory	
DisplayName	nxDatabase1
Enabled	True
EventLog	
EventLogEnabled	False
Exclusive	False
FailSafe	False
Name	nxDatabase1
ReadOnly	False
Session	
Tag	0
Timeout	-1
TransContext	
Version	2.0000 Public Beta 4 (D7)

Right Window (Events View):

Event	Description
EventLog	
OnStateChanged	
OnStateTransChar	
Session	
TransContext	

Purpose

A TxnDatabase component represents a single alias or folder and defines the transaction context if transactions are used. During a transaction, the state of the database is stored and all changes essentially made on commit. As such, you can have several TxnDatabase components linked to the same session and alias, holding different states.

As well as linking the TxnDatabase to a TxnSession component, either an alias name (taken from server alias mappings) or an alias path (relative to the server location) must be specified. Functionality contained includes table management such as pack, destroy, etc and transaction control.

Key Properties

AliasName	Select from the list derived from the session connected to this component. (AliasName and AliasPath are mutually exclusive properties. That is, you select one or the other, but do not set both.)
AliasPath	Use this to directly access the data tables. This is especially useful at designtime with embedded server apps. (AliasName and AliasPath are mutually exclusive properties. That is, you select one or the other, but do not set both.)
Exclusive	Grants you exclusive access to the alias. It can improve performance for embedded server apps which have exclusive access by their very nature.
FailSafe	Sets failsafe transactions.
ReadOnly	Allows readonly access to the alias. This should be set automatically with an alias on a readonly device such as a CD-ROM.
Timeout	If a value of -1 is selected, then it uses the timeout from the session connected. This is the timeout value used for any operations involving the database component.

Please also see Common Properties and Events for more details.



11.6.4 TxnBackupController

TxnBackupController



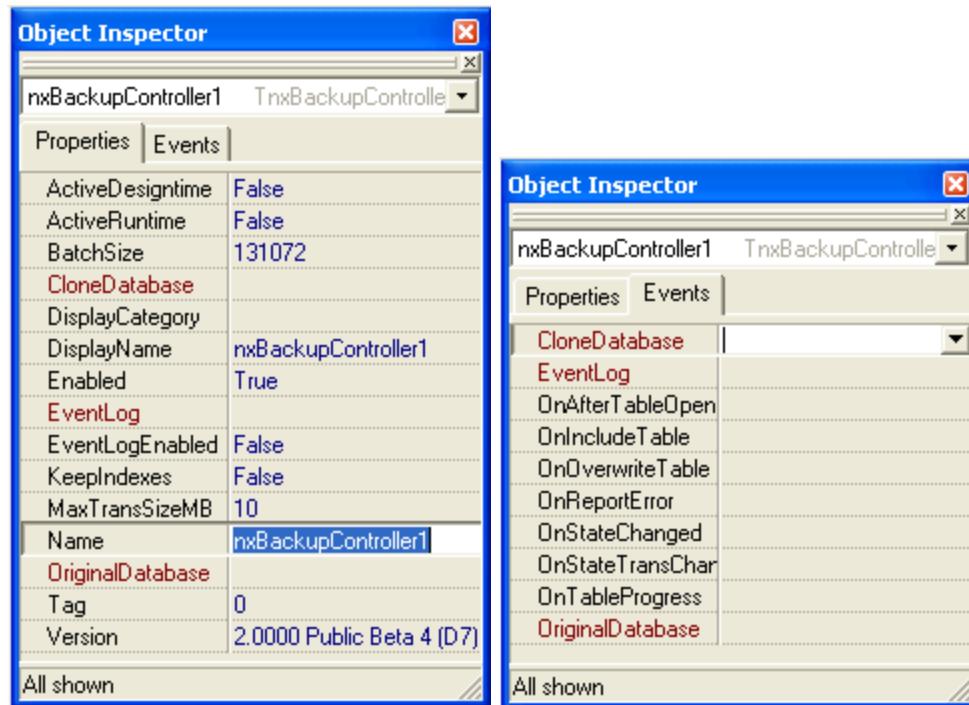
Supported in products: All.

NexusDB real time backup and restore component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The TnxBackupController is a real-time backup component which enables you to make a backup of a database while it is in use. Simply assign source and target database components, which may or may not reside on the same server. Events are available to filter which tables are backed up. To save space and time, the backed up version of the tables do not contain indices. If required, the TnxBackupController can be used to restore tables from the backup. The indices will be recreated during the restore procedure.

If the two databases (original and backup) are on separate servers, the data is sent from the source server to the client and then to the target server using batch read and writes. If both databases reside on a single server, a direct transfer takes place with no data being sent across the network to the client.

The entire process occurs within a snapshot transaction. Therefore, the database is guaranteed to be in a consistent state. Restoring tables requires exclusive access to the destination alias; this is to enable index recreation.

While running the backup process, the server needs enough RAM and temporary storage to cache all updates from other sources. The restore process requires exclusive access to the target (restored) directory.

Key Properties

BatchSize	Size in KiloBytes in which data is backed up. Adjust accordingly if there is a low MaxRAM setting on the server.
CloneDatabase	Target database the data is backed up to.
MaxTransSizeMB	Maximum size in MegaBytes of nested transactions (used to speed up the overall process).
OriginalDatabase	Source database the data is copied from using a snapshot transaction.

Key Events

These events allow you to do custom processing during the backup process.

OnIncludeTable	Raised for each table included in the backup.
OnOverwriteTable	Raised if a table is about to be overwritten.
OnReportError	Raised if an error is detected.
OnTableProgress	Allows you to update a custom dialog informing the user of the progress of the backup.

Please also see Common Properties and Events for more details.



11.6.5 TnxSimpleSession

TnxSimpleSession



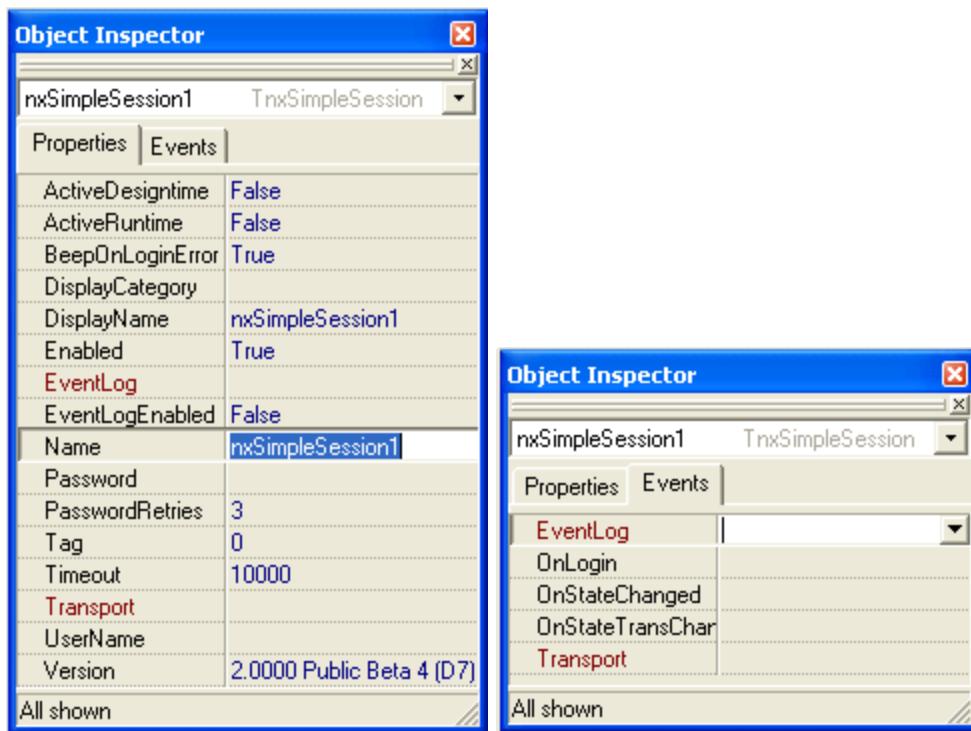
Supported in products: All.

NexusDB simple session component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The TrxSimpleSession is used to open a connection and does not include any database aware functionality. As opposed to the TrxSession component, the TrxSimpleSession connects directly to a transport component instead of a server or remote server. All messages are then sent to a command handler. Using this component enables an efficient messaging system with custom command handlers.

Key Properties

BeepOnLoginError	Makes noise on a login failure.
Password	Set to a password that will be recognized by the server for the set username.
PasswordRetries	How often it brings up the password dialog and retries login with the server before it gives up and raises an error.
Transport	Connect to a TrxTransport component.
UserName	Set to a username recognized by the server.

Key Events

OnLogin	Allows you to show your own custom login dialog.
---------	--

Please also see Common Properties and Events for more details.



11.6.6 TnxTransContext

TnxTransContext



Supported in products: All.

??? todo

Icon



Properties and Events view

Two side-by-side "Object Inspector" windows, both titled "nxTransContext1 TnxTransContext".

Left window (Properties tab):

ActiveDesignime	False
ActiveRuntime	False
Default	False
DisplayCategory	
DisplayName	nxTransContext1
Enabled	True
EventLog	
EventLogEnabled	False
FailSafe	False
Name	nxTransContext1
Session	
Tag	0
Timeout	-1
Version	2.0000 Public Beta 4 (D7)

Right window (Events tab):

EventLog	
OnStateChanged	
OnStateTransChar	
Session	

Purpose

Please also see Common Properties and Events for more details.



11.6.7 TnxDatabaseMapper

TnxDatabaseMapper



Supported in products: Developer Edition, Embedded Edition.

NexusDB Database Mapper component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Two side-by-side screenshots of the Object Inspector window. Both windows have 'nxDatabaseMapper1 TnxDatabaseMapper' selected in the title bar.

Left Window (Properties View):

Properties	
ActiveDesignTime	False
ActiveRuntime	False
DefaultSession	
DisplayCategory	
DisplayName	nxDatabaseMapper1
Enabled	True
EventLog	
EventLogEnabled	False
Mappings	(TnxDatabaseMapping)
Name	nxDatabaseMapper1
Options	[dmoAddAliasNames]
Tag	0
Version	2.0000 Public Beta 4 (D7)

Right Window (Events View):

Events	
DefaultSession	
EventLog	
OnStateChanged	
OnStateTransChar	

Purpose

This component is required to integrate Report Builder or Shazam into an application. Instead of assigning a session to these reports, a TnxDatabaseMapper is assigned instead. This provides a unique namespace of databases and allows you to include data from different databases on separate servers that have the same name. Each database can be mapped to a different name and uniquely recognised.

Key Properties

DefaultSession	If you set a default session, then the component presents all the alias names from the default session to the outside world.
Mappings	This is a collection which you can set to map database names as required and described above.
Options	???todo

Please also see Common Properties and Events for more details.



11.7 Plugins

11.7.1 TnxServerInfoPlugin

TnxServerInfoPlugin



Supported in products: All.

NexusDB Server Info Plugin component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Object Inspector	
nxServerInfoPlugin1 TnxServerInfoPlugin	
Properties Events	
ActiveDesignTime	False
ActiveRuntime	False
DisplayCategory	Plugins
DisplayName	nxServerInfoPlugin1
Enabled	True
EventLog	
EventLogEnabled	False
Name	nxServerInfoPlugin1
Tag	0
Version	2.0000 Public Beta 4 (D7)
All shown	

Object Inspector	
nxServerInfoPlugin1 TnxServerInfoPlugin	
Properties Events	
EventLog	
OnStateChanged	
OnStateTransChar	
All shown	

Purpose

When associated with a server engine component, this can be used to retrieve information from the server such as current time, component status etc.

Please also see Common Properties and Events for more details.



11.7.2 TnxRemoteServerInfoPlugin

TnxRemoteServerInfoPlugin



Supported in products: All.

NexusDB Remote Server Info Plugin component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Two side-by-side 'Object Inspector' windows for the 'nxRemoteServerInfoPlugin1' component. The left window shows the 'Properties' tab with the following settings:

ActiveDesignTime	False
ActiveRuntime	False
DisplayCategory	Plugins
DisplayName	nxRemoteServerInfoPlugin1
Enabled	True
EventLog	
EventLogEnabled	False
Name	oteServerInfoPlugin1
Session	
Tag	0
Timeout	0
Version	2.0000 Public Beta 4 (D7)

The right window shows the 'Events' tab with the following events listed:

EventLog	
OnStateChanged	
OnStateTransChar	
Session	

Purpose

Containing the same interface as the TnxServerInfoPlugin, this component connects to either a session or simple session and passes messages via transports to the plugin command handler. There must exist a TnxServerInfoPlugin component attached to the command handler or simple command handler respectively.

Key Properties

Session	Connect to a either a simple session or a full TnxSession component.
---------	--

Please also see Common Properties and Events for more details.



11.7.3 TnxServerInfoPluginCommandHandler

TnxServerInfoPluginCommandHandler



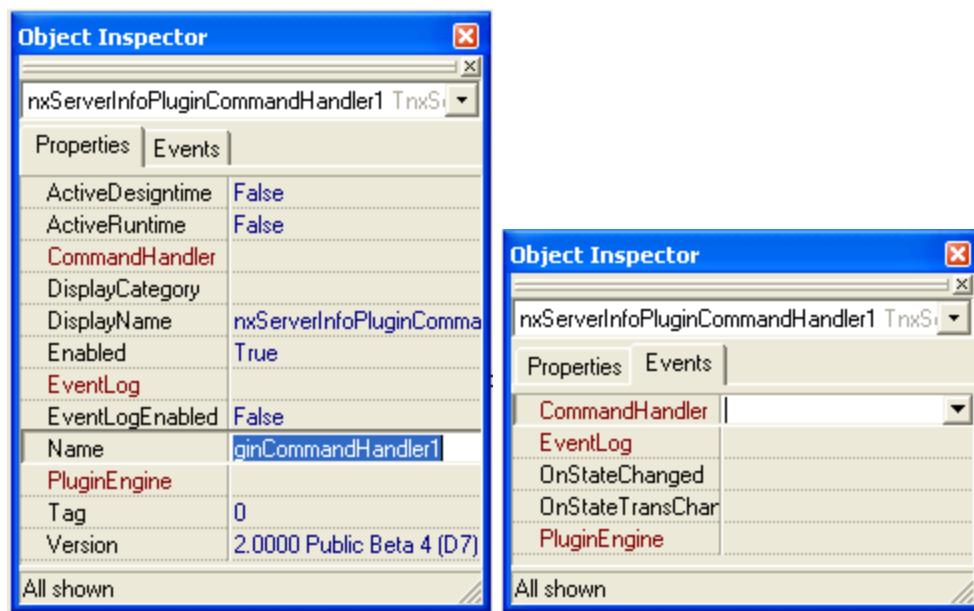
Supported in products: All.

NexusDB Server Info Plugin Command Handler component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The TnxServerInfoPluginCommandHandler connects directly to the server command handler. Unrecognised messages passed to the server command handler will be sent on to all plugin command handlers. This plugin command handler can be used with the TnxServerInfoPlugin component to retrieve server information. Similar to the server command handler, the plugin command handler unpacks recognised messages and forwards the method calls to the plugin engine.

Key Properties

PluginEngine	Assign to a server info plugin engine.
--------------	--

Please also see Common Properties and Events for more details.



11.7.4 TnxRemotingServer

TnxRemotingServer



Supported in products: All.

NexusDB Remoting Server component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
DisplayCategory	Plugins
DisplayName	nxRemotingServer1
Enabled	True
EventLog	
EventLogEnabled	False
Name	nxRemotingServer1
Tag	0
Version	3.0002 Release (D11)

Event	Handler
» EventLog	
OnStateChanged	
OnStateTransChanged	

Purpose

Please also see Common Properties and Events for more details.



11.7.5 TnxRemotingClient

TnxRemotingClient



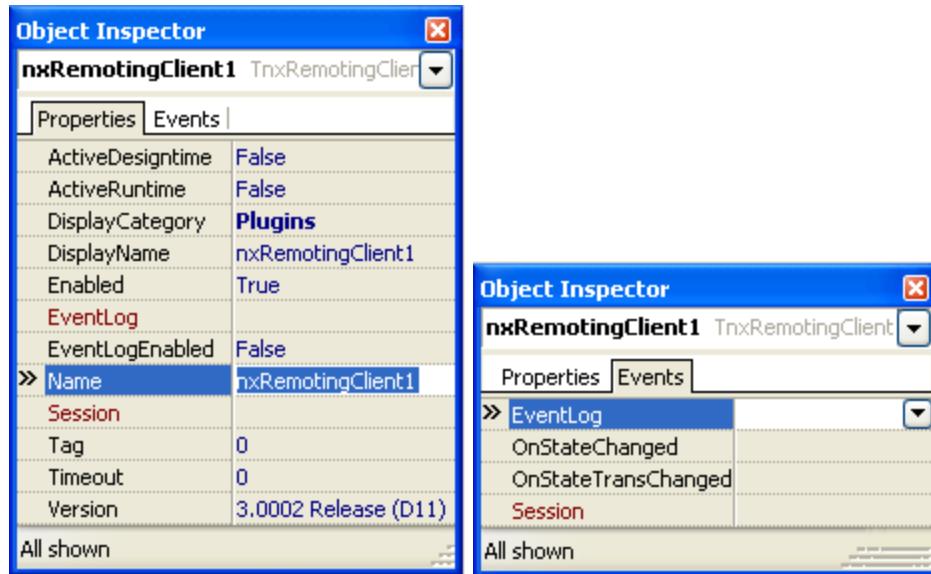
Supported in products: All.

NexusDB Remoting Client component. Default values are shown in the object inspector snapshot.

Icon

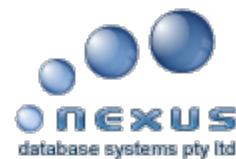


Properties and Events view



Purpose

Please also see Common Properties and Events for more details.



11.7.6 TnxRemotingCommandHandler

TnxRemotingCommandHandler



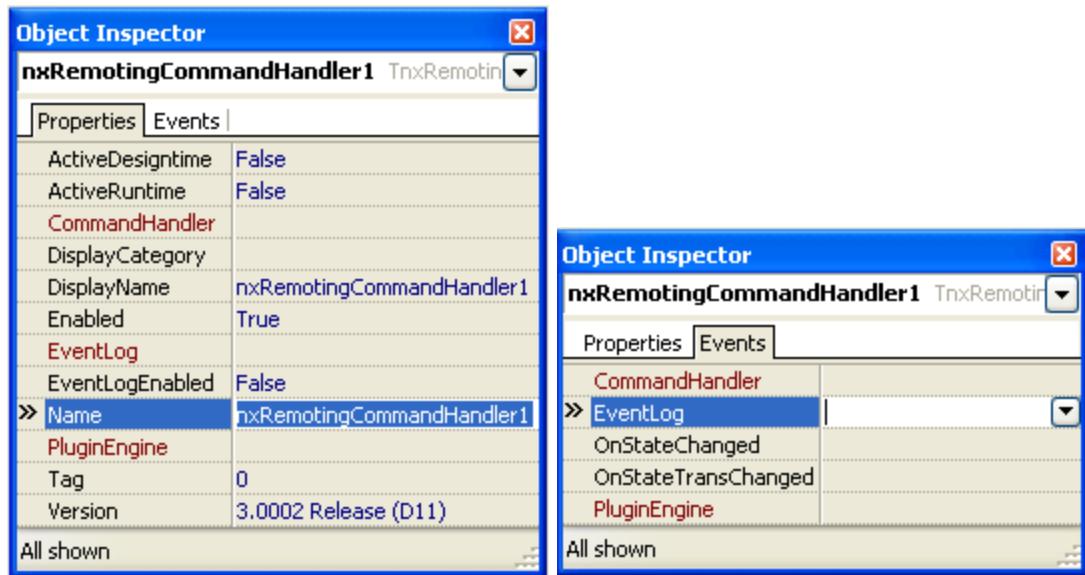
Supported in products: All.

NexusDB Remoting Command Handler component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

Key Properties

PluginEngine	Assign to a server info plugin engine.
--------------	--

Please also see Common Properties and Events for more details.



11.7.7 TnxRemoteRemoteCommandsPlugin

TnxRemoteRemoteCommandsPlugin



Supported in products: All.

NexusDB Remote Remote Commands Plugin component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

The image displays two side-by-side screenshots of the Delphi IDE's Object Inspector window, both titled "Object Inspector" and showing the component "nxRemoteRemoteCommandsPlugin1".

Top Screenshot (Properties View):

Property	Value
ActiveDesignTime	False
ActiveRuntime	False
DisplayCategory	Plugins
DisplayName	nxRemoteRemoteCommandsPlugin1
Enabled	True
EventLog	
EventLogEnabled	False
» Name	nxRemoteRemoteCommandsPlugin1
Session	
Tag	0
Timeout	0
Version	3.0002 Release (D11)

Bottom Screenshot (Events View):

Event
» EventLog
OnStateChanged
OnStateTransChanged
Session

Purpose

Key Properties

Please also see Common Properties and Events for more details.



11.7.8 TnxServerRemoteCommandsPlugin

TnxServerRemoteCommandsPlugin



Supported in products: All.

NexusDB Server Remote Commands Plugin component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view

 Two screenshots of the Object Inspector window for the 'nxServerRemoteCommandsPlugin1' component. The left screenshot shows the 'Properties' tab with the following data:

ActiveDesigntime	False
ActiveRuntime	False
AllowIP	
AllowTransport	
DisplayCategory	Plugins
DisplayName	nxServerRemoteCommandsPlugin1
Enabled	True
EventLog	
EventLogEnabled	False
Name	nxServerRemoteCommandsPlugin1
ScriptDir	
Tag	0
Version	3.0002 Release (D11)

 The right screenshot shows the 'Events' tab with the following data:

EventLog	
OnStateChanged	
OnStateTransChange	

Purpose

Key Properties

Please also see Common Properties and Events for more details.



11.7.9 TnxRemoteCommandsPluginCommandHandler

TnxRemoteCommandsPluginCommandHandler



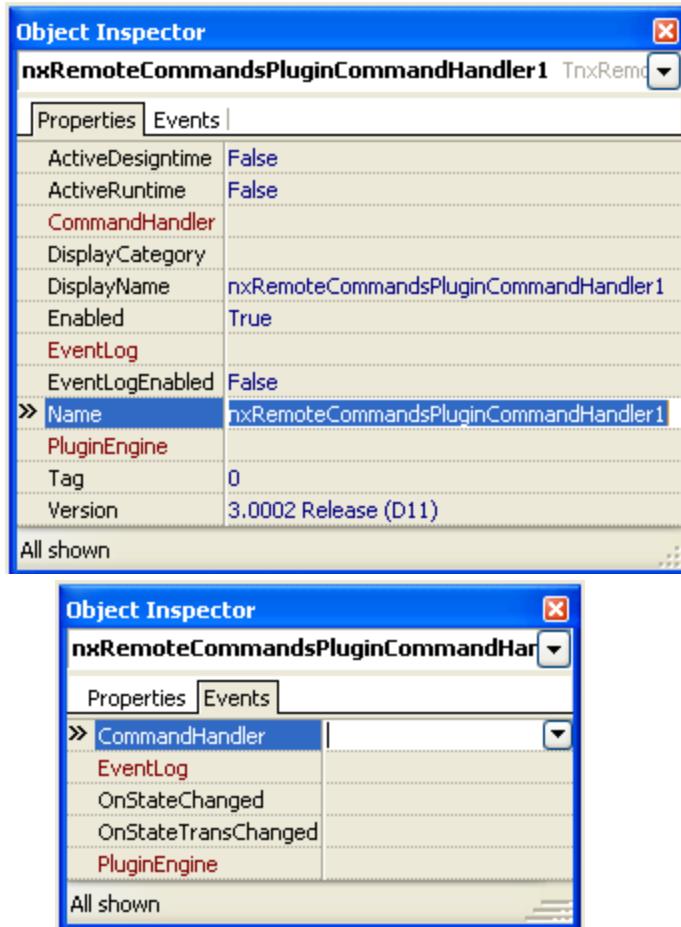
Supported in products: All.

NexusDB Remote Commands Plugin Command Handler component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

Key Properties

PluginEngine	Assign to a server info plugin engine.
--------------	--

Please also see Common Properties and Events for more details.



11.8 TnxDataset Group

11.8.1 Common Properties and Events

Common Properties and Events



These components are inherited from Borland's TDataSet and implements most of the contained functionality. The following properties and events are relevant to all TnxDataSet components.

Key Properties

AliasName	Select from the list of alias names associated with the connected session. (Either set Database or the AliasName and Session properties.)
BlockReadOptions	???todo
Database	Connect to a TnxDatabase component. (Either set Database or the AliasName and Session properties.)
Exclusive	Open dataset in exclusive mode.
FilterTimeout	Maximum time in milliseconds before the onServerFilterTimeout event is raised. If you have a very large number of records in the dataset to be filtered then you may have to specify a suitably large timeout value.
Options	The options that can be set for TnxDataSet components are dsoCacheBlobs, dsoNoNestedTransForCachedBlobs and dsoOptimisticLocks. DsoCachBlobs should be set to true so that write access to blobs will be cached on the client and written to the server in the context of a transaction started on post. This must be set to false if blobs are too large to be cached client-side. Setting dsoOptimisticLocks to true will lock an edited record only during post, ensring the record has not been changed in the meantime. If set to false, a record will be locked on edit, restricting other users from editing the record.
Session	Connect to a TnxSession component. (Either set Database or the AliasName and Session properties.)

Key Events

OnServerFilterTimeout Raised if the filter is not completed in the time specified by the FilterTimeout property.

Advice

A query is good for access to a small number of records out of a large table. It does pre-processing and returns a result set matching your conditions. It will try to use all available optimizations to reduce the amount of data that has to be read from disk.

A table is good for unrestricted (or ranged on active index) access to large number of records. There's no time consuming pre-processing involved and no result set is build.

For your example, returning 10 out of 10,000,000 rows a query is perfect. It will use indices to build your result set in virtually no time and will just return the 10 records you are looking for. If you want to do the same thing with a table on the other hand, it evaluates a filter during Next/Prior calls on a

record by record basis. Following the current index. It will take nearly forever because the Next call has to skip possible millions of records, evaluating each one against the filter.

Now look at the other extreme. You want 9,999,990 out of 10,000,000 rows. (e.g. simply add a tiny little NOT at the begin of your condition...). A Table will return that data instantly. A Query will pre-process your condition, building a huge result set and take forever doing it.



11.8.2 TnxTable

TnxTable



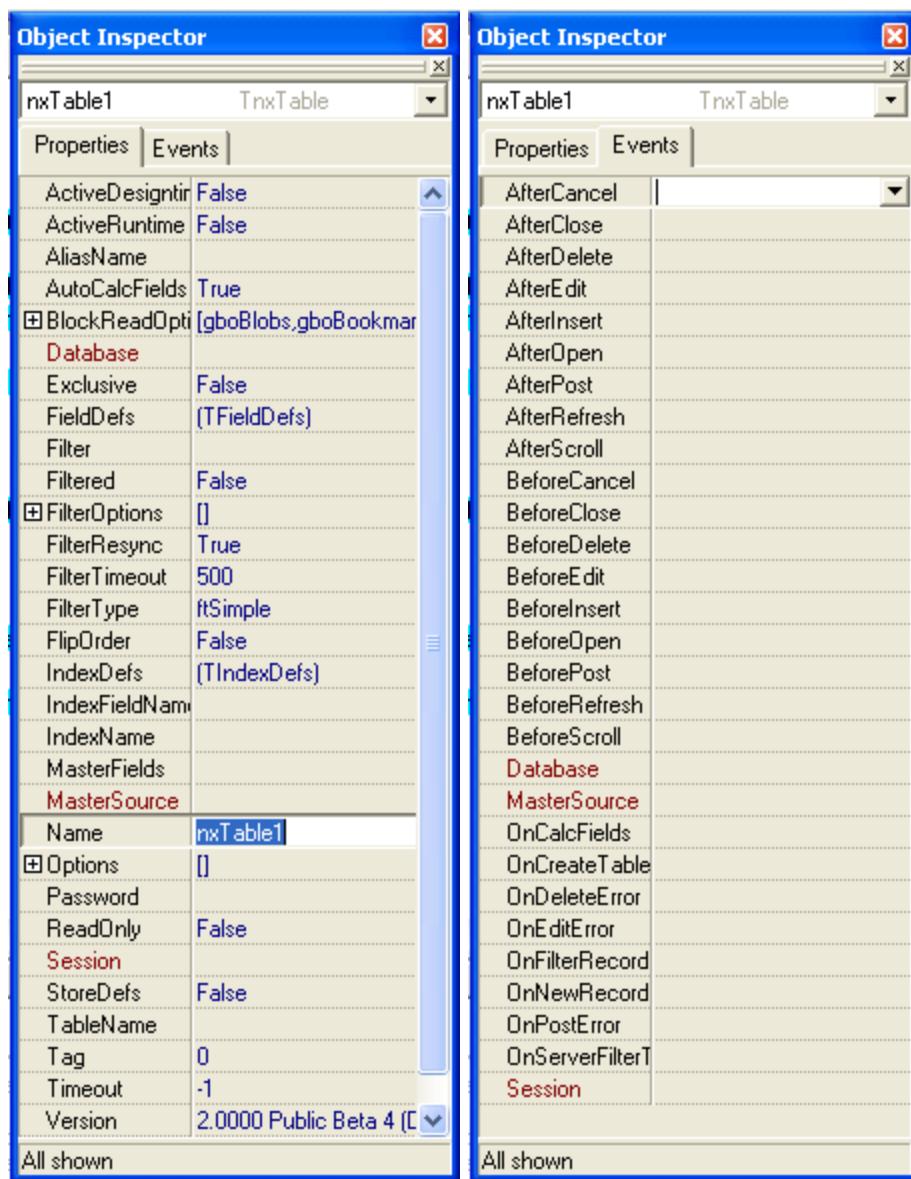
Supported in products: All.

NexusDB table component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

The TnxTable component provides navigational access to a single table in a database. Largely compatible with TTable, it supports the use of indices. A TnxTable component can either be directly linked to TnxDatabase and TnxSession components with a selected table, or alternatively linking to a session and setting the alias name property will create an implicit database component.

By default, TnxTables use pessimistic locking. Editing a table will lock a particular record on the server, such that all other attempts to access this record will be denied.



11.8.3 TnxQuery

TnxQuery



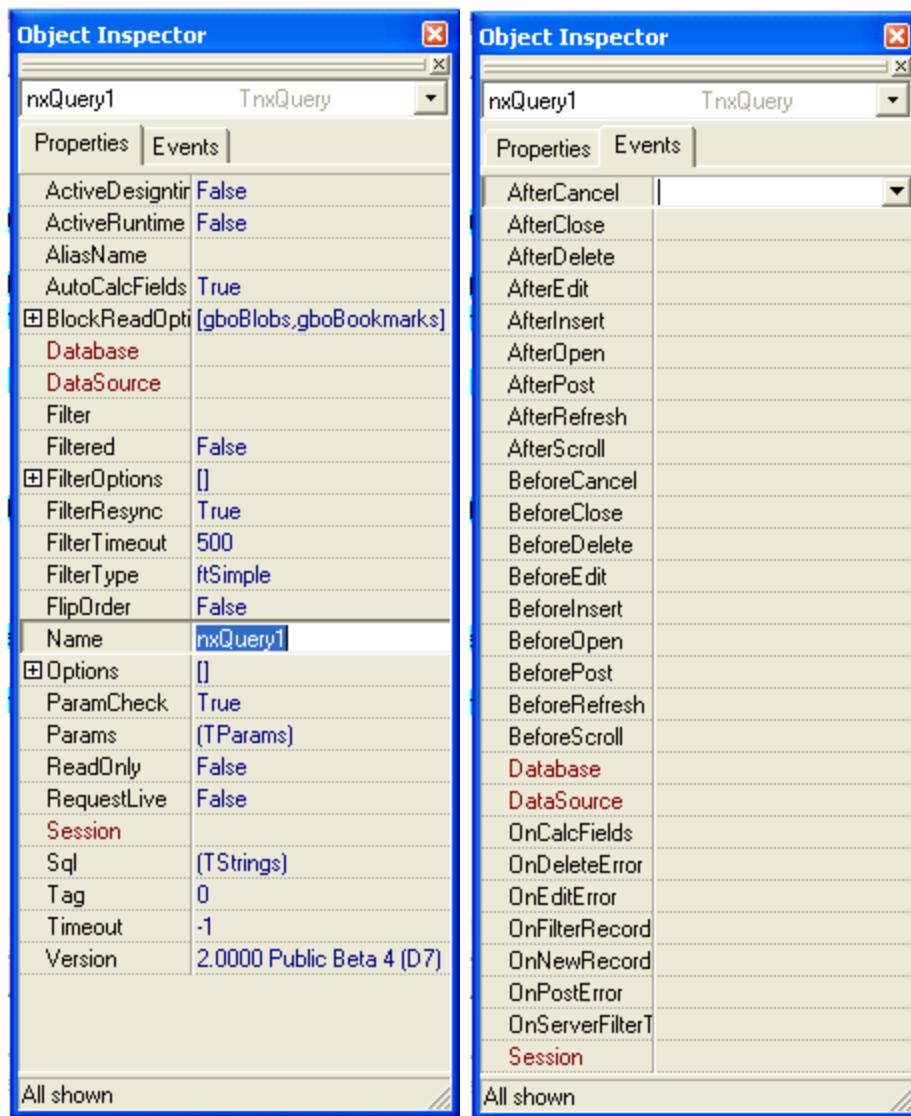
Supported in products: All.

NexusDB query component. Default values are shown in the object inspector snapshot.

Icon



Properties and Events view



Purpose

Similar to a TnxTable component, TnxQuery can be linked to a TnxDatabase and TnxSession and specify a table name for explicit connections or simply link to a session and by using an alias name, create an implicit database component. This component is mostly compatible with TQuery and requires a link to TnxSQLEngine from the server engine. The functionality is limited by the SQL standard and is most commonly used for complex joins, reporting and bulk updates.

Key Properties

RequestLive	Allows live datasets to be selected. If the engine can't return a live dataset, the data returned is read-only, irrelevant of this property.
-------------	--



11.8.4 TnxCachedDataset

TnxCachedDataset



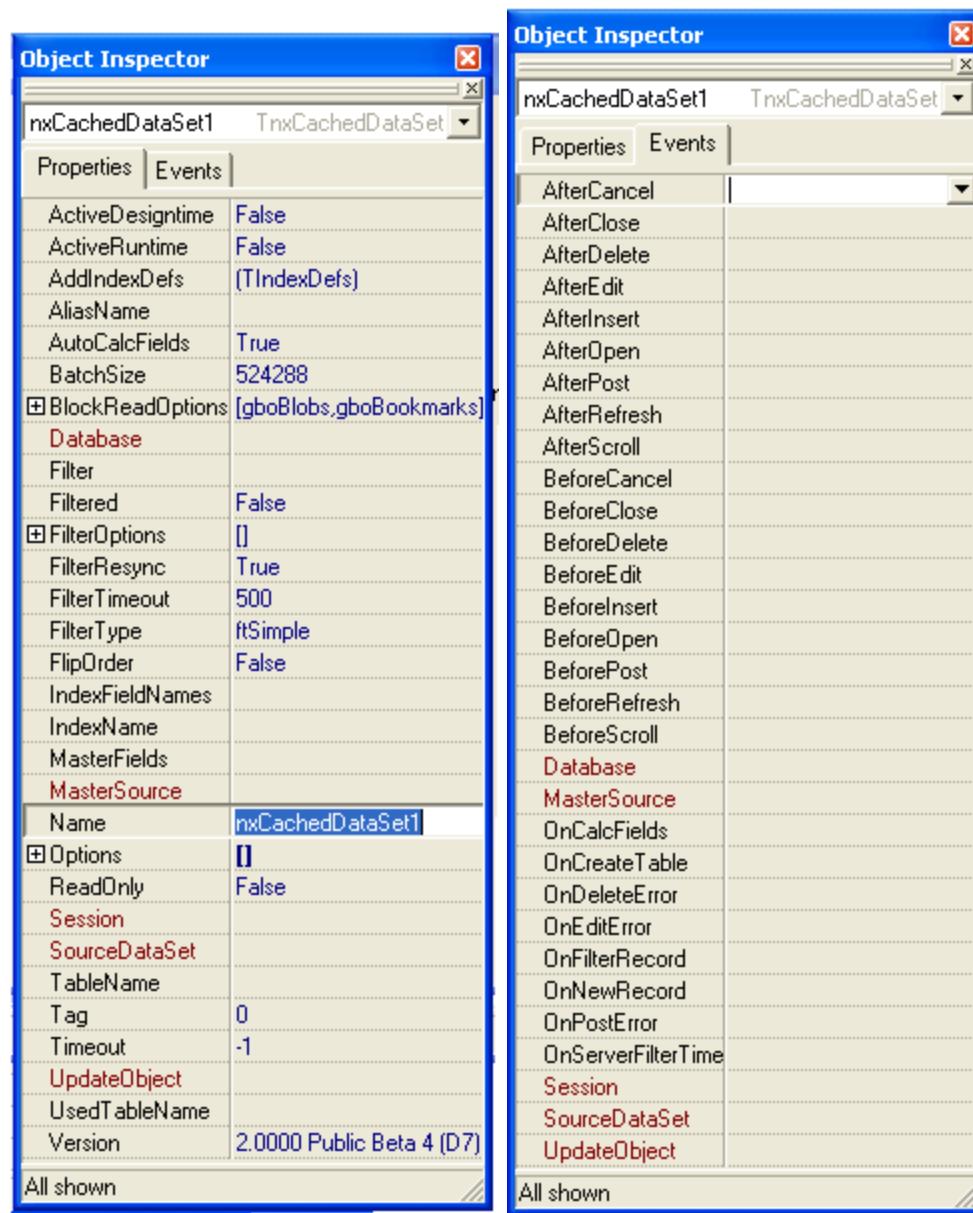
Supported in products: All.

??? todo

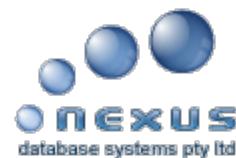
Icon



Properties and Events view



Purpose



11.8.5 TnxStoredProc

TnxStoredProc



Supported in products: All.

??? todo

Icon



Properties and Events view

Properties	
ActiveDesignTime	False
ActiveRuntime	False
AliasName	
AutoCalcFields	True
BlockReadOptions	[gboBlobs,gboBookmarks]
Database	
DataSource	
Filter	
Filtered	False
FilterOptions	[]
FilterResync	True
FilterTimeout	500
FilterType	ftSimple
FlipOrder	False
Name	nxStoredProc1
Options	[]
Params	(TPParams)
ReadOnly	False
RequestLive	False
Session	
StoredProcName	
Tag	0
Timeout	-1
Version	2.0000 Public Beta 4 (D7)

Events	
AfterCancel	
AfterClose	
AfterDelete	
AfterEdit	
AfterInsert	
AfterOpen	
AfterPost	
AfterRefresh	
AfterScroll	
BeforeCancel	
BeforeClose	
BeforeDelete	
BeforeEdit	
BeforeInsert	
BeforeOpen	
BeforePost	
BeforeRefresh	
BeforeScroll	
Database	
DataSource	
OnCalcFields	
OnDeleteError	
OnEditError	
OnFilterRecord	
OnNewRecord	
OnPostError	
OnServerFilterTime	
Session	

Purpose



11.8.6 TnxSQLUpdateObject

TnxSQLUpdateObject



Supported in products: All.

??? todo

Icon



Properties and Events view

Two side-by-side "Object Inspector" windows. Both windows have a title bar "Object Inspector" and a dropdown menu "nxSqlUpdateObject1 TnxSQLUpdateObject". Each window has two tabs: "Properties" and "Events". The "Properties" tab is selected in both. The properties listed are: DeleteSql (TStrings), DisplayCategory, DisplayName (nxSqlUpdateObject1), InsertSql (TStrings), ModifySql (TStrings), Name (nxSqlUpdateObject1), Params (TPParams), Tag (0), and Version (2.0000 Public Beta 4 (D7)).

Property	Type
DeleteSql	(TStrings)
DisplayCategory	
DisplayName	nxSqlUpdateObject1
InsertSql	(TStrings)
ModifySql	(TStrings)
Name	nxSqlUpdateObject1
Params	(TPParams)
Tag	0
Version	2.0000 Public Beta 4 (D7)

Purpose



11.8.7 TnxMemTable

TnxMemTable



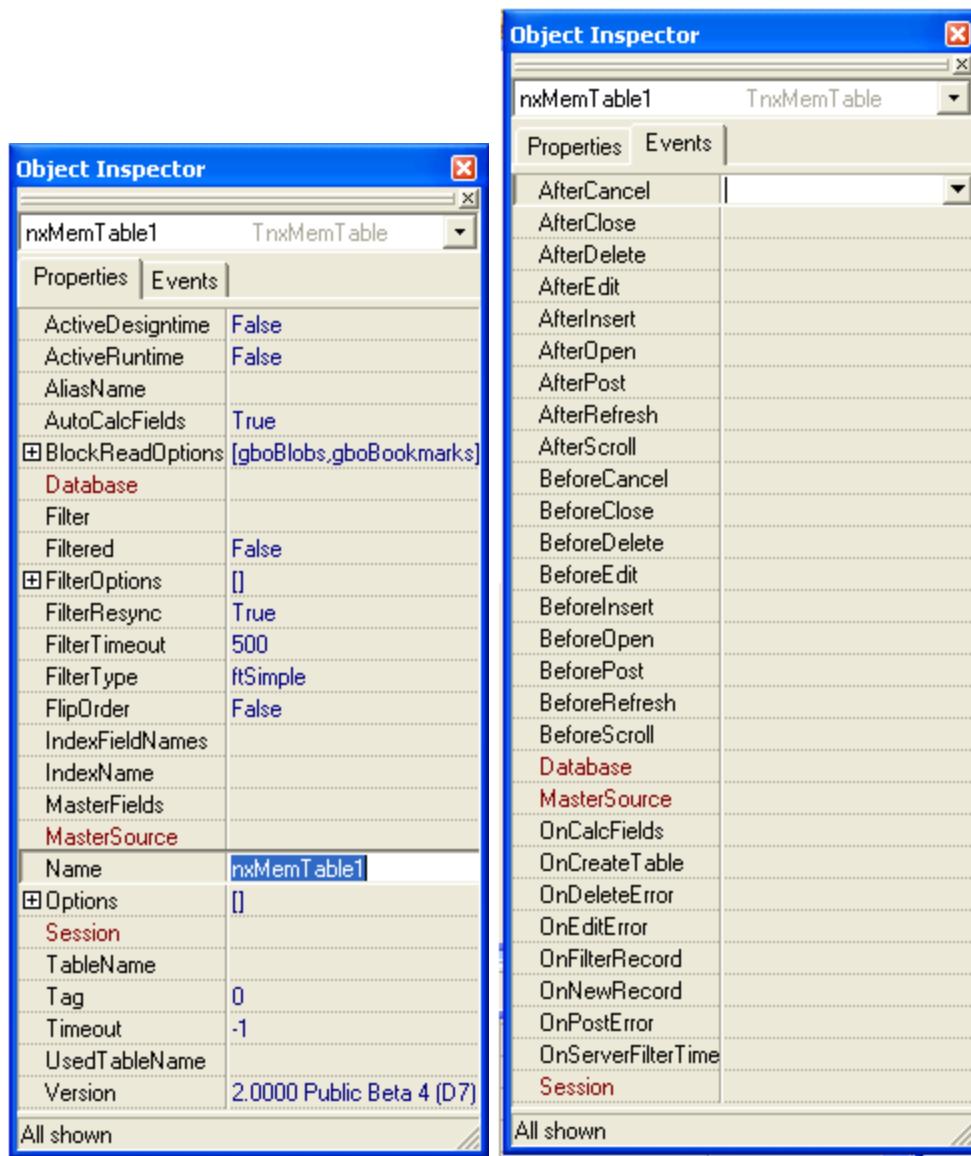
Supported in products: All.

??? todo

Icon



Properties and Events view



Purpose



11.9 Logging Components

11.9.1 TnxEventBasedLog

TnxEventBasedLog



Supported in products: All.

???todo

Icon



Properties and Events view

Object Inspector	
nxEventBasedLog1 TrxEventBas	
Properties Events	
CacheEnabled	True
DisplayCategory	
DisplayName	nxEventBasedLog1
Enabled	True
EventDispatch	
Name	xEventBasedLog1
PrioritiesToLog	[IpDebug,IpInfo,IpW]
SyncWithMain	True
Tag	0
Version	2.0000 Public Beta
All shown	

Object Inspector	
nxEventBasedLog1 TrxEventBas	
Properties Events	
EventDispatch	
OnLogDataFile	
OnPersistLogD	
All shown	

Purpose



11.9.2 TnxWinEventLog

TnxWinEventLog



Supported in products: All.

??? todo

Icon



Properties and Events view

The image shows two separate instances of the 'Object Inspector' dialog box. Both instances have the title bar 'nxWinEventLog1 TnxWinEventL...' and the tab 'Properties' selected. The left window displays detailed property settings:

CacheEnabled	True
DefaultCategory	Database
DefaultMsgPriority	IpError
DisplayCategory	
DisplayName	nxWinEventLog1
Enabled	True
EventDispatcher	
Name	WinEventLog1
PrioritiesToLog	[IpWarn, IpError]
ResourceDllFile	
SourceName	NexusDB
Tag	0
Version	2.0000 Public Beta

The right window shows a simplified list of properties under the 'EventDispatcher' category:

EventDispatcher	
-----------------	--

Purpose



11.9.3 TnxEventLogDispatcher

TnxEventLogDispatcher



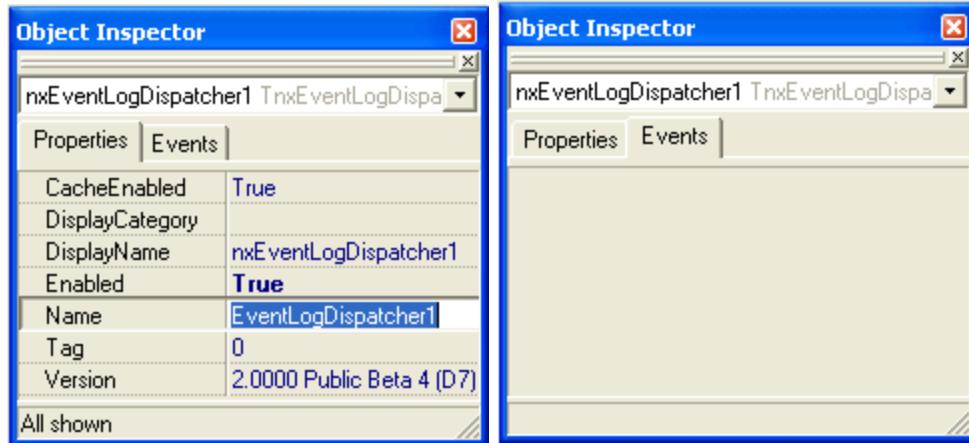
Supported in products: All.

??? todo

Icon



Properties and Events view



Purpose



11.9.4 TnxEventLog

TnxEventLog



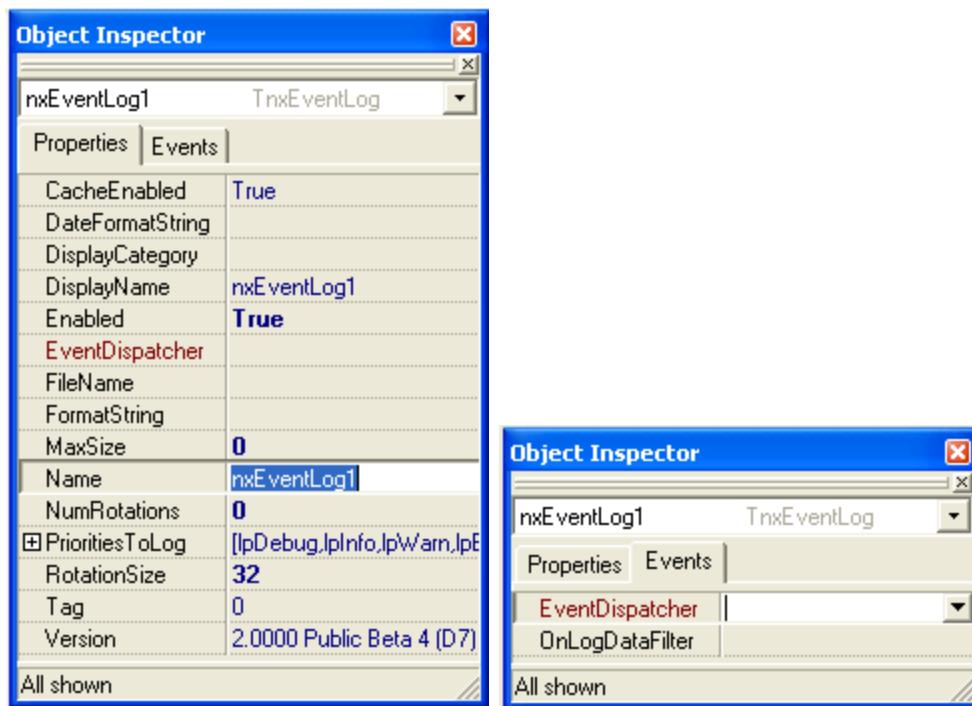
Supported in products: Developer Edition, Embedded Edition.

NexusDB Event Log component. Default values are shown in the object inspector snapshot.

Palette Icon



Properties and Events view



Purpose

This representation of a log file on disk can be used by most components by setting their EventLog property to this TnxEventLog component. Be aware of the different units used by CacheLimit (bytes), TruncateSize (kilobytes) and MaxSize (Megabytes).

Key Properties

CacheEnabled	Boolean value which turns the Cache on/off. If enabled this will significantly improve performance. However, if enabled and the server crashes, all entries in the cache will be lost.
CacheLimit	Size, in bytes, of the cache.
FileName	Output filename for the Event Log. If blank, no logging is performed.
MaxSize	Maximum size, in Mbytes, of the log file. When the file reaches MaxSize, it is truncated to TruncateSize. Truncation keeps the most recent entries in the log file.
TruncateSize	Size of the truncated log file, in Kbytes, when MaxSize is reached.
WriteBlockData	The logging process, depending on the component being logged, writes messages and binary data. WriteBlockData controls whether or not the binary data is written to the log file. For most purposes the binary data is not needed. Recommended for advanced users only.

Please also see Common Properties and Events for more details.



12 Converting from other Versions or Database Engines

12.1 Upgrading V1 Applications

Upgrading V1 Applications

???todo



12.2 Delphi Interfaces

Delphi Interfaces

??? update



They're one of the great secrets of the Delphi language. It allows you to separate DESIGN from IMPLEMENTATION.

I use interfaces mostly for the logical boundaries between program sections, though I also use them to wrap useful classes. If you look through my code, if it looks like I'm using a class, it's probably an interface instead.

Some big reasons off the top of my head:

- 1) Automatic lifetime management
- 2) Abstraction independent of class hierarchy (implementation)
- 3) Multiple interfaces PER CLASS -- you get the benefits of C++'s multiple inheritance without the nightmare.

Let's walk through these quickly:

(1) Automatic lifetime management

This means that the implementing class is freed automatically when all references to the interface are gone.

If you don't think this is any big deal, when's the last time you worked with strings? Could you imagine the huge PITA to have to allocate and free each and EVERY string? To not be able to simply use "+" to concatenate strings? I've lived that. No fun!

I feel the same way about having to allocate and free classes manually.

Compare:

```
//  
// Example without  
interfaces
```

```
//  
var  
    something :TSomething;  
begin  
    something :=  
TSomething.Create;  
try  
    something.Whatever;  
finally  
    something.Free;  
end;  
end;
```

vs.

```
//  
// Example with interfaces  
//  
var  
    something :ISomething;  
begin  
    something :=  
TSomething.Create;  
    something.Whatever;  
end;
```

(2) Abstraction independent of class hierarchy (implementation).

Classes simplify the messiness inherent in procedural programming. Interfaces simplify the messiness inherent in classes. You only deal with an abstraction, not an implementation.

There's more than one way to skin a cat. Imagine being able to skin cats, dogs, giraffes, and remote asteroids with the same tool. Welcome to the ability to HIDE IMPLEMENTATION. Yes, one can hide SOME implementation inside of classes, but classes themselves are still implementation. For the most part this is no big deal, but by the same token, programming without classes is no big deal. What? You're too young to remember procedural programming?

grin

Programmers who think classes are no big deal, because they were raised on classes, do not understand the huge chasm between procedural and object-oriented programming. They claim to "understand" it, but they do not *know* it. When I talked with fellow programmers when we were making the transition from procedural to OO programming, we universally complained about the added complexity of classes, but saw that they could be useful in certain circumstances. (This is the same attitude Delphi programmers express about interfaces. "Useful for COM, but that's about it.") We didn't understand that the benefit from using classes far exceeded the massive amount of extra work involved in using classes. Do you really understand how much extra crap is involved with writing classes? No? Why do I manipulate classes with ModelMaker's Code Explorer (<http://www.modelmakertools.com/ce.htm>)?

My observation is that the power of this does not sink in until one is comfortable with interfaces, and one can freely design with each

complimenting the other.

This allows one to design very stable code, and removed the headache of massive changes because you had to alter a class hierarchy.

As an example, I referred to my database abstraction interfaces. One interface, with ANY NUMBER of database implementations underneath. Need to use NexusDB? Interbase? MySQL? ADO? Oracle? Each set of classes has its own class hierarchy. They cannot be substituted one for another.

Any database code that I write uses the same two interfaces, `IItcDatabase` and `IItcQuery`. When I need to port to a different database, such as performance testing, preparing comparative analyses, or whatever, I generally do not need alter my code much, if at all. All changes are due to SQL variances.

Now try ripping out entire class hierarchies and replace them with a completely different set of classes.

3) Multiple interfaces PER CLASS

Let's face it. If you've programmed for any length of time you get into those nasty class decisions where multiple inheritance would "be nice." Frequently it's not "just nice," but a correct design.

C++ has multiple inheritance, but if you're used it at any length, you've been burned. It comes with the territory. Because we can decouple DESIGN from IMPLEMENTATION with interfaces, we can get the bonuses without the pain.

I've already burned up too much time already; I'll let this one fall short.

.....

I have to admit that I didn't understand interfaces until I prepared a session on COM programming for our local user group. I came back and said, "we need two sessions." The first was on interfaces only. The second on COM. I don't know that I will be able to explain any better than the other guys, but I'll give it a shot:

Interfaces is the Delphi alternative to multiple inheritance. A simple example is almost impossible, because interfaces are really a tool for dealing with design complexity. See if this makes sense:

Let's say, as an example, that you would like to customize a bunch of standard Delphi visual controls, to include field validation. You could create a descendent of each kind of form you want to enhance, and then when you finish editing, do something like:

```
FOR i := 0 TO
Form1.ControlCount - 1 DO
  IF Form1.Controls[i]
IS TMyEditControl THEN
    Ok :=
TMyEditControl(Form1.Controls[i]).ValidateMe
  ELSE IF
Form1.Controls[i] IS
TMyCheckboxControl THEN
    Ok :=
TMyCheckboxControl(Form1.Con
```

```

trols[i]).ValidateMe
ELSE IF
Form1.Controls[i] IS
TMyComboBox THEN
Ok := 
TMyComboBox(Form1.Controls[i]
]).ValidateMe

```

...and so on...

If, instead, you designed an interface that supported a "ValidateMe" procedure, you could support this interface in all your inherited controls,

a la:

```

IAutoValidate = Interface
FUNCTION
ValidateMe : Boolean;
END;

```

Now, when you enhance a control, declare it thusly:

```

TMyEditControl           =
Class(Tedit,
IAutoValidate)
PROCEDURE
ValidateMe;
END;

```

Then you can have a generic "validator" routine that can handle any control, whether it supports your validator interface or not:

```

FUNCTION ValidatesOk( Ctl :
TControl )
VAR
Ival      :
IAutoValidate;
BEGIN
IF
Supports(IAutoValidate, Ctl,
Ival) THEN
Result := 
Ival.ValidateMe
ELSE Result := 
True;
END;

```

Look what this gains you:

1. You have an absolutely generic way of handling special circumstances.
2. Interface "membership" is not evaluated until runtime. That means that you can have standard handling behavior for objects that may not even exist yet at compile time.
3. You can add the desired "interface" methods to any appropriate object without having to go re-craft case statements, as in the first example.
4. You can cluster related capabilities in a single interface.

5. You can define multiple interfaces for an object, cleaning up the design process considerably.
6. You can delegate interface methods to a separate object using the `IMPLEMENTS` keyword.
7. You can radically reduce the amount of superfluous code and code repetition in your programs.

The real trick to understanding interfaces is to know that an interface is really a contract--when an object implements an interface, it promises to provide specific implementations for every method defined in the interface. As a result, if an object supports a particular interface, you KNOW to a certainty that there are certain things you can do with that object, even if you know nothing else about it.

Anyway, in the right situation, interfaces are a tremendously effective tool. I hope this was helpful to somebody...



12.3 Upgrading FF2 Projects to NexusDB

Upgrading FF2 Projects to NexusDB



This section describes for developers a fairly quick way of upgrading existing FF2 applications to NexusDB. The steps involved, at a high level, are:

1. Copy and convert FF2 tables to NexusDB tables in a new directory (alias) using the converter utility.
2. Replace and update FF components in your application to NexusDB components.
3. If you used TffDataDictionary, then you may have to convert some of your source code.

Let us now go through the conversion steps in detail.

Step 1 - Database Conversion

This is the easiest step. Simply run the `nxmlporter.exe` utility. Don't forget to first create an output alias to pump the new tables and data into.

Step 2 - Component Substitution

The following quickly summarizes the component substitutions that you have to make. Note that the `TffClient` (`TffCommsEngine`) and `TffSession` components are replaced by a single `TnxSession` component.

FlashFiler	NexusDB
<code>TffClient</code> , <code>TffCommsEngine</code>	Remove; merged into <code>TnxSession</code>
<code>TffSession</code>	<code>TnxSession</code>
<code>TffDatabase</code>	<code>TnxDatabase</code>
<code>TffTable</code>	<code>TnxTable</code>
<code>TffQuery</code>	<code>TnxQuery</code>
<code>TffServerEngine</code>	<code>TnxServerEngine</code>
<code>TffRemoteServerEngine</code>	<code>TnxRemoteServerEngine</code>
<code>TffSQLEngine</code>	<code>TnxSQLEngine</code>
<code>TffServerCommandHandler</code>	<code>TnxServerCommandHandler</code>

TffLegacyTransport	TnxWinsockTransport or TnxNamedPipeTransport
TffEventLog	TnxEventLog
TffSecurityMonitor	TnxSecurityMonitor
TffThreadPool	Remove; not applicable

Step 3 - Source Code Conversion

Most of the source code changes that you will need to make are related to the data dictionary object. Another change you need to be aware of is that the SessionName and DatabaseName properties no longer exist. Similarly, there is no longer a Client component at all.

For most of the other changes, we provide a simple automatic conversion utility for renaming all the FF2 related names and properties to the NexusDB equivalent. The utility is called Convert_Src2Nx.dpr and is located in the Convert subdirectory of your NexusDB installation. Please note that you almost certainly still have to some manual changes to make the application compile. At the moment the best advice is to run the code converter tool and then fix up the other issues by the old "compile and go to next syntax error" technique!

Finally, you will have a clean compile and go! You should now have converted your FF2 application to NexusDB. As with all such tasks, the first project is the hardest – you should find subsequent applications easier to convert.



12.3.1 Differences between FF and NexusDB

Differences between FF and NexusDB



In this section is detailed some of the differences, enhancements and extensions, based on a broad comparison between FlashFiler and NexusDB. Even though NexusDB builds on many of the same principles as FlashFiler, and has a similar programming API, NexusDB is a complete rewrite. There is no FlashFiler source in the NexusDB code base.

The Memory Manager

This is completely new. It is better for NexusDB than Delphi's memory manager. The Delphi memory manager suffers from memory fragmentation and uses about 30% more memory. The new memory manager can be used in non-NexusDB projects as well. The memory manager is optimized for many small allocations (<64kb), everything larger goes directly to the OS (VirtualAlloc).

To use the memory manager you simply have to add "nxReplacementMemoryManager," as the very first line of the "uses ...;" clause in your project source (.dpr file).

The Buffer Manager

The buffer manager is responsible for all reads and writes to/from the hard disk. It does this via a new OO (and extensible) file access layer. All transaction management is performed here. As a consequence a new form of transaction has been developed and implemented: snapshot transactions. A snapshot transaction is a read only transaction that provides a consistent view of the entire database at the instant in time the transaction was started, without placing any locks on the system. This allows other clients to freely modify the database without affecting the integrity of the system from the viewpoint of the client who started the read only transaction. Snapshot transactions are deletion safe.

In FF2 all reads and writes were done inside a single lock. In NexusDB all write operations have been moved outside this lock. Thus a transaction can be committed and even while it is still writing to disk, other clients can access the data from the committed transaction and even start another transaction. There can even be a situation where multiple transactions are being committed at the same time provided they don't modify the same tables. At all times the data is safe and the integrity of the database assured to a level way above that of FF2.

The buffer manager requires no more than half the memory used by FF2 on newly inserted blocks. The reuse of pages is more efficient. It obeys the max RAM setting even if you have in-memory tables and large transactions, provided read-write temporary storage is available.

The Journal Engine

Completely rewritten and now has 3 settings compared with just one in FF2:

- Mode 1 – before and after images are stored. If a reboot finds these images then a dialog is displayed and the operator asked whether to commit or rollback the changes found.
- Mode 2 – stores before images only. If a reboot finds these images an automatic rollback is performed. This can be thought of as a conservative, or cautious, mode of operation.
- Mode 3 – stores after images only. If a reboot finds these types of images an automatic commit is performed.

Mode 2 and 3 require no user interaction. This makes them perfect if the server is running as a service.

Data Dictionary for a Table

This has been redesigned and uses a fundamentally different architecture. For example, block sizes are now a set of types and not an integer value. The best way to get to grips with it is to read the help file.

It is now completely OO and extensible. You can create your own descriptor objects. Every item in the data dictionary has a string list attached so that you can store extra information for use later by your program. This allows you to select the different sub-engines for use by the table.

The Stream Engine

FF2 could only save a single stream in a table – the data dictionary. In NexusDB an unlimited number of named streams can be stored in a table. Think of this as table specific Blobs. There are four functions to operate on this stream engine – list all streams; read a stream; write a stream; and delete a stream.

Using named streams, additional custom information can be stored in a table besides the records and indices. For example, when the backup component creates a copy of a table, the original data dictionary is stored as a named stream. No indices are added to the copy; just the raw records are copied. When restoring, the indices are recreated based on the information stored in the 2nd stream.

The Record Engine

This manages the records in the pages. A significant improvement over FF2 is the way NexusDB reuses deleted records: NexusDB is faster to reuse a mass of them – it's highly optimized. FF2 uses a linear linked list. NexusDB uses a linked list of blocks and, internal to each block, a linked list. Thus NexusDB will fill block-wise whereas FF2 fills essentially at random.

Currently the record size is fixed, but we also plan to implement a variable length record engine. Because of the extensible architecture of NexusDB it will be possible to use both record engines at the same time. That is, you can mix tables with different record engines in the same database.

The Key Engine

The complete process of key creation and comparison is in a different (new) engine – FF2 does not have this. There are 2 default key engines:

- RefNr (sequential access), and,
- Default composite engine (much like FF2)

The composite engine has one composite key field engine per field. This is comparable to the index helper objects in FF2. ASC/DESC can be set on a per field basis, as can locale and string sort/word sort. Within limits this can be accessed with the VCL. It is straightforward to write your own key field engine or even your own complete key engine to, e.g., implement a SoundEx Sort.

The Index Engine

FF2 used a b-Tree index. The new default index implementation is a modified b*-Tree (hybrid) – the modification is for speed. It is more compact. The modular architecture allows you to write additional index engines and assign them on a per index basis. e.g., hash tables or even a sequential sort.

The BLOB Engine

It is now possible to use different BLOB engines for each table. The new (default) BLOB engine has 3 layers: an intra-block heap manager, the management layer, and the interface layer. It is highly efficient, can deal with huge objects, avoids fragmentation and has a very high speed. Overall it's a big improvement over FF2.

Command Handler

The NexusDB command handler works much the same as in FF2 but much more efficiently. There are less messages and they are faster in what they do. There is a "look-ahead" capability built in to reduce message counts and thus speed things up significantly over slow data connections.

Transports

The basic transport structure has been kept similar to FF by request from FF developers. There are two new classes; Winsock TCP (over IP4) and Named Pipes. Both of these support broadcasts and compression. There is an extensible compression architecture with 3 different default compression algorithms already implemented. On NT-based systems these transports use, if possible, only as many threads as the number of CPU's in the system. (Blocking sockets, asynchronous I/O and I/O Completion Ports are used to achieve this.)

Winsock can be used on Win 95+ clients and servers. If the server is not running on an NT-based system it will require one thread per connection as I/O Completion Ports are not available on these systems. If you want to use TCP as transport but have no network card installed in your machine you can simply add a dialup adapter to your configuration. This will install the Winsock dlls.

Named Pipes can be used on Win95+ clients and NT-based servers.

Both support call-backs (server -> client messaging) but with Named Pipes this only works if the client is NT-based.

Client Side Components

NexusDB has no TffClient equivalent, the functionality is now merged in with the TxSession component. There are no more [automatic] components. The database, session and table/query components link directly by component instead of by SessionName etc, as in FF2.

From the client side you also have access to snapshot transactions and nested transactions. Just remember that for nested transactions you call STARTTRANSACTION multiple times and thus have to call either COMMIT or ROLLBACK the same number of times.

New Batch Modes

You can set a BlockReadSize to be the minimum number of bytes for each batch operation. This includes optional support for Blobs and Bookmarks to reduce network traffic – just set an option.

Once the BlockReadSize is set to greater than zero bytes, you are automatically positioned on the first record. That is, it does a Table.First for you. Thereafter you can only do Table.Next and Table.First to navigate through the records. Use Table.EOF as per normal. To turn this off, just set BlockReadSize back to zero. Note that BlockReadSize in the BDE is the number of records. In NexusDB it means the number of bytes because we have Blobs as well with variable sizes.
To use Batch Writes/Appends look at the following structure:

```
Table.BeginBatchAppend;    //  
to start  
Repeat  
    Table.Append:  
        // set field values  
        including blobs  
        Table.Post;  
    Until {batch complete  
    condition};  
    Table.EndBatchAppend;    //  
    to post batch off
```

The last statement turns batch appending off.

In-memory Tables

To use an in-memory table, simply place <> around the table name. E.g.,

```
Table.TableName:='<MemTable1  
>';
```

In-memory tables live for as long as the server (or embedded server in the client app if created on the client side) is up.

Other Stuff

To avoid the problems you could run into in FlashFiler if you left an Active property set to True at designtime, NexusDB has ActiveDesignTime and ActiveRunTime properties. They are available in all stateful components including:

- Transports
- Server Engines
- Command Handlers
- Plug-ins
- Extenders
- Sessions
- Databases
- Tables/Queries

Furthermore, Transports also have ServerNameDesignTime and ServerNameRunTime as visible properties, for the same reason. With this enhancement over FlashFiler, you need never worry about resetting your development settings before compiling your application.

Plug-in support is similar to FF2. Extenders have been redesigned with a slightly different architecture. Existing FF2 extenders will have to be modified for NexusDB.



13 Code Examples & Fragments

Code Examples & Fragments



In this section we present some example projects, in Delphi, which illustrate some of the more commonly used features of a DBMS. Each of these projects is intended to illustrate a straightforward way of implementing a common task. There are many ways to do each of these (for instance, with table creation we use the standard TDefs approach rather than the TnxDataDictionary object) and the way chosen is usually for familiarity with the BDE way. Naturally there are also a number of projects that illustrate NexusDB specific features such as Memory Tables and the Block Mode operations.

13.1 Creating the Northwind sample database

Creating the Northwind sample database



All examples use the Northwind sample database.

The SQL script to create the database is installed in the root of Examples subfolder of your NexusDB installation and can be downloaded from our web page.

Create the database

Here's a step by step guide how you can run the SQL script from Enterprise manager.

- Create a new directory which will later hold the tables
- If you've downloaded from the webpage, unzip all files to this directory
- Open NexusDB Enterprise Manager
- Right click the **Internal Server** node in the left hand servers window
- Select **New Database Alias** from the popup menu
- Enter a new alias name (e.g. Northwind) and point it to the above directory
- Right click the newly added alias
- Select **SQL** in the popup menu and the SQL window opens on the right hand side
- Press CTRL-O or select Query/Open from the main menu
- Navigate to the above directory and select the Northwind_V2.sql file
- Press CTRL-E or select Query/Execute to run the script. This can take a few seconds.

That's it. You have setup the full example database and it is ready for use.



13.2 Applications with the database embedded

Applications with the database embedded



NexusDB is a true Client/Server DBMS. Even so, one of its many features is that the engine can be compiled directly into a project so that your application can be distributed as a single exe without the need to separately install, configure, and maintain the NexusDB Server DBMS backend.

In this section we show you how to set up a data module that can be used in any embedded application and then provide a complete example of an embedded application.



13.2.1 Setting up a general NexusDB Embedded Server Data Module

Setting up a general NexusDB Embedded Server Data Module



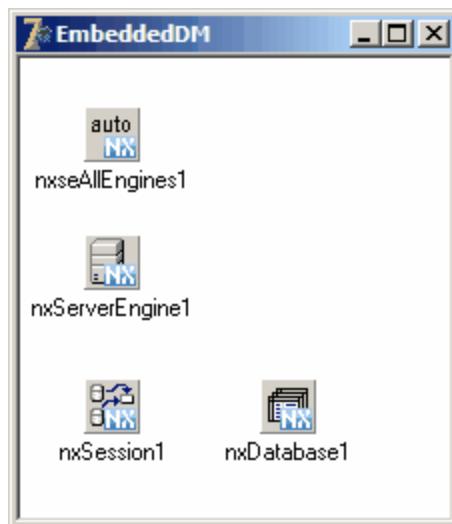
This section explains how to set up a generic simple data module that you can use to create database enabled applications. In the Delphi IDE you will need to create a new project. Now follow these steps to create the data module shown in the figure below (for Delphi 7):

Hint

If you don't want to create this datamodule on your own you can find the complete source in the **Examples\Manual\Repository** subdirectory of your NexusDB installation. If you don't have this directory you download the examples from our webpage.

Step 1: Creating the data module

Create a new data module in Delphi by selecting **File/New/Data Module** from the Delphi main menu. Locate a **TnxseAllEngines**, **TnxServerEngine**, **TnxSession** and **TnxDatabase** component from the NexusDB tab on your component palette and drop them on the form. The data module should now look similar to this:



Step 2: Setting up the session

Connect the nxSession1 component to your NexusDB ServerEngine1 component (pull down list of the Server Engine property). The properties of your session should now look as shown.

Name	nxSession1
Password	
PasswordRetrie	3
ServerEngine	nxServerEngine1
Tag	0
Timeout	-1

Step 3: Setting up the database

Connect the nxDatabase1 component to the Session component (from the pull down list). Finally, you will need to point the AliasPath (not the AliasName) property to where your data tables reside, in this case, I've got mine sitting in **f:\data\Northwind**. The Property Inspector for your database component should now look pretty much the same as the one shown.

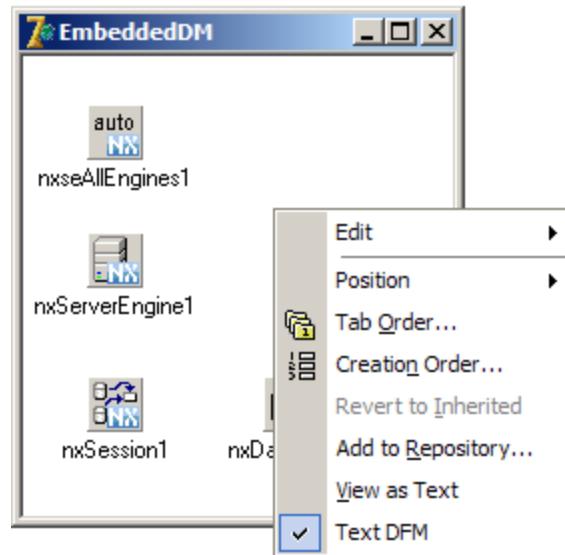
ActivVeruntime	False
AliasName	
AliasPath	f:\data\Northwind
Default	False
DisplayCategor	
DisplayName	nxDatabase1
Enabled	True
EventLog	
EventLogEnab	False
Exclusive	False
FailSafe	False
Name	nxDatabase1
ReadOnly	False
Session	nxSession1
Tag	0

Step 4: Adding the data module to the repository

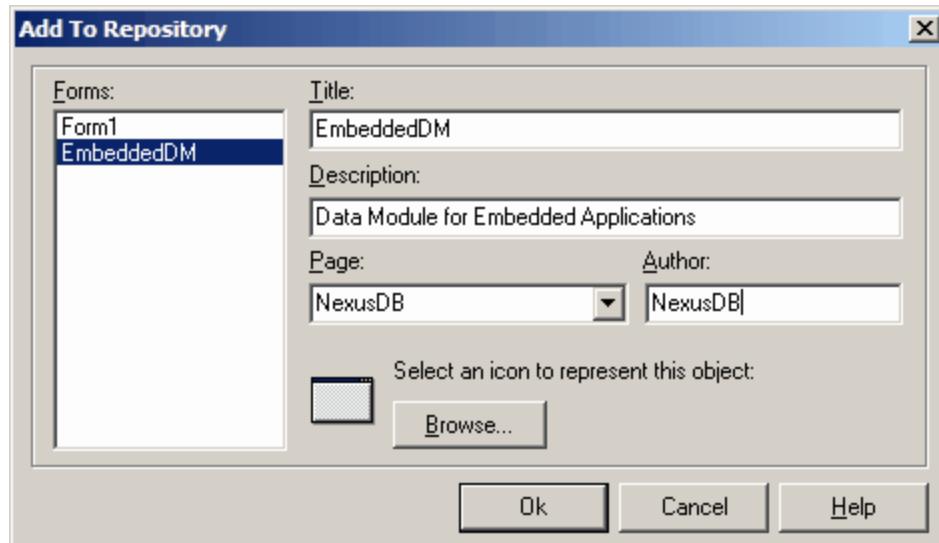
As the last step we're adding the new data module to the Delphi repository so that we don't need to recreate it all the time. For this first save the data module, eg. by pressing CTRL-S. Save the datamodule to a directory which you want to use as a repository in the future.

Caveat

Make sure to not delete or move the files later, otherwise you won't be able to use it anymore. When done, find an empty spot on the data module and right-click. In the popup menu select **Add to Repository**.



This brings up the following dialog. Please fill out the fields like this and press **OK**.



You are now ready to go! It's that straightforward!



13.2.2 Creating a simple databound form

Creating a simple databound form



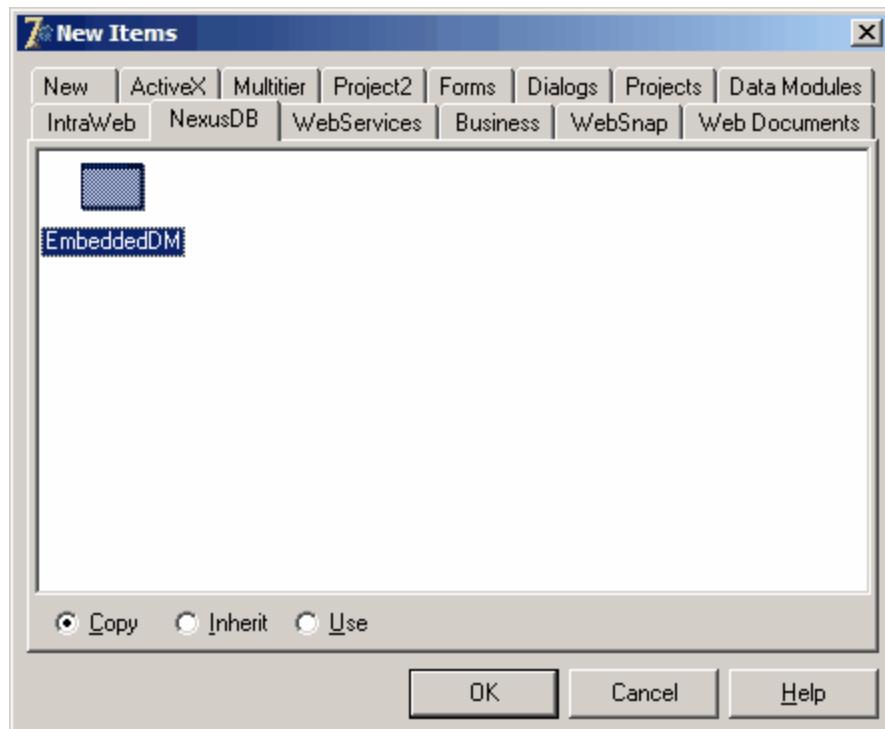
Here we will show how to build a very simple database application using the embedded data module we just created. In the Delphi IDE you will need to create a new project.

Hint

If you don't want to create this example on your own you can find the complete source in the **Examples\Manual\SimpleEmbedded** subdirectory of your NexusDB installation. If you don't have this directory you download the examples from our webpage.

Step 1: Adding the data module

Select **File/New/Other...** from the Delphi main menu. This will popup the **New Items** dialog where you can find the **NexusDB** tab. Select it, focus the **EmbeddedDM** entry, make sure **Copy** is activated.



Pressing **OK** will add a copy of the data module to your application.

Step 2: Adding a table

Locate and drop on a **TnxTable** on the data module. To connect to the database available in the data module, just connect the Database property via the drop down list (the correct Session will be set for you). Finally select the table for this instance in the TableName property. Your **nxTable1** component should now look as follows (after selecting a table).

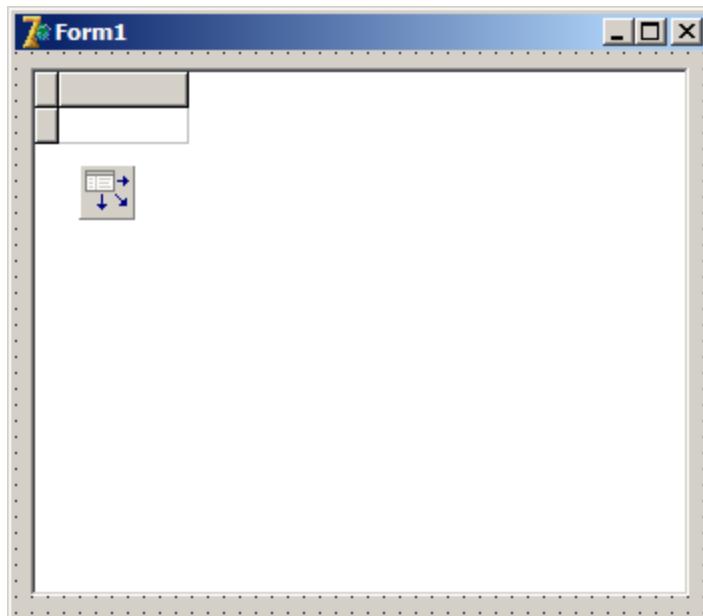


AutoCalcFields	True
BlockReadOpti	gboBlobs,gboBookr
Database	nxDatabase1
Exclusive	False
TableName	Customers
Tag	0

Now save the new data module as embeddeddm.pas to the new project directory.

Step 3: Creating a form

Create a new form and drop a **TDatasource** (Data Access tab in the component Palette) and a **TDBGrid** (Data Controls) on the form. Arrange them similar to this:



Select the DBGrid and set its **Anchors** property to [akLeft,akTop,akRight,akBottom]; this makes sure that we can resize the grid with the form.

Step 4: Hooking up the data

Make sure you've saved the data module above, then add the data module to the uses clause in the interface, which will add the components on it to the drop down lists of the current form. Connect the **Dataset** property of the Datasource to the table

AutoEdit	True
DataSet	EmbeddedDM.nxTable1
Enabled	True
Name	DataSource1
Tag	0

and set the **Datasource** property of the DBGrid to our datasource.

Ctl3D	True
Cursor	crDefault
DataSource	DataSource1
DefaultDrawing	True
DragCursor	crDrag

The only thing left to do is to make sure the tables are opened when starting the app.

Step 5: Opening the table

For this purpose simply set the **ActiveRuntime** property of the table to true. That's it. Compile the app and run.

Form1		
CustomerID	CompanyName	ContactName
ALFKI	Alfreds Futterkiste	Maria Anders
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
ANTON	Antonio Moreno Taquería	Antonio Moreno
AROUT	Around the Horn	Thomas Hardy
BERGS	Berglunds snabbköp	Christina Berglund
BLAUS	Blauer See Delikatessen	Hanna Moos
BLONP	Blondesddsl père et fils	Frédérique Citeaux
BOLID	Bólido Comidas preparadas	Martín Sommer
BONAP	Bon app'	Laurence Lebihan
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln
BSBEV	B's Beverages	Victoria Ashworth
CACTU	Cactus Comidas para llevar	Patricia Simpson
CENTC	Centro comercial Moctezuma	Francisco Chang
CHOPS	Chop-suey Chinese	Yang Wang

Congratulations. You've just created your first NexusDB database application!

Caveat 1

When using embedded mode always keep in mind that the IDE opens tables if you have **ActiveDesignTime** enabled. Trying to run your embedded application from the IDE/Debugger will fail to open the table at runtime, because it is using a different ServerEngine. Nexus only allows ONE server engine to access a file at a time. In general we recommend always using an external server while developing.

Caveat 2

Since we are relying on Delphi creating the datamodule make sure that the datamodule is automatically created **before** the form is created. Check the forms tab of the project options!



13.2.3 Making the embedded application path independent (copy)

Making the embedded application path independent



Let's add a run-time enhancement to our application so that the resulting exe is not restricted to the path coded into the AliasPath property of the database component. That is, let's make the application automatically change the ServerName point to the one given as the first command line parameter (here we are assuming that the data tables will reside in the alias we use on this server).

Please note that for this to work you need to set the ActiveRuntime properties of the transport, database and tables to false.

To have the tables created in the same directory as where the exe resides, plus to activate the NexusDB component chain, add the following code to the OnCreate event handler of the form (or data module if you used one) as follows:

```

procedure
  TMainformDialog.FormCreate(S
ender: TObject);
begin
  if ParamCount>0 then
  begin
    // make sure the
    // transport is inactive
    RemoteDM.nxWinsockTransp
    ort1.Active:=false;
    // set the server name
    RemoteDM.nxWinsockTransp
    ort1.ServerNameRuntime:=para
    mStr(1);
    end;
    RemoteDM.nxTable1.Active:=
    True;
  end;

```

That's it. When the program starts, the engine will connect to the server given as first command line parameter. If none it will connect to the one given at compile time.

Recommendation

We recommend to extend above method a bit to optionally read the AliasName too, or alternatively read the configuration from an ini file or the registration.

```
if ParamCount>1 then

    RemotedM.nxDataBase1.Alias
    Name:=paramStr(2);
```

This will later allow your application to be used on different servers and aliases without the need to change the database.

Caveat

For the application to start successfully the data alias **and** the table must exist on the connected server!

Hint

If you don't want to create this example on your own you can find the complete source in the **Examples\Manual\RunEmbedded** subdirectory of your NexusDB installation. If you don't have this directory you download the examples from our webpage.



13.3 Client/Server Applications - Client App Separate from the Server

Client/Server Applications - Client App Separate from the Server



In this section we set up a data module that you can use in any application that you are going to deploy as a true C/S system. That is, you will need to install and configure the NexusDB server engine (NexusDB Server.exe) on an appropriately designated file server. Your client applications must then connect to this server with an appropriate transport. We end the section with a small example C/S application.



13.3.1 Starting and setting up the server

Starting and setting up the server



The first thing to do for a Client/Server application is to correctly configure and start the server. Here's a quick step by step guide to get it setup and running for the examples.

Step 1: Start the server

This is very simple. Just double click the nxServer.exe and you're done

Step 2: Setting up the server and activating it

- Select the **Server Engine** item from the Database settings on the left hand tree view. The Server Engine configuration displays.
- Make sure that a **Server Name** is specified and **MaxRAM** is set to a sensible size. You might also want to change other settings.
- Select the **Aliases** item from the NexusDB Server main menu. The Alias Configuration window displays.
- **Add an alias called** Northwind and point it to the directory you've created the db in.
- Now activate the Server modules by selecting **Start all modules** from the Server menu. You can also (de)activate single modules by selecting the item in the tree view and checking the **Active** check box on the top of each configuration.

Step 3: Verifying connections

The next step is to verify you can connect to the NexusDB Server using Enterprise Manager on the same machine. Start NexusDB Enterprise Manager and look out for the servers found. If your server name appears your server is set up correctly.



13.3.2 Setting up a generic NexusDB Data Module for C/S

Setting up a generic NexusDB Data Module for C/S



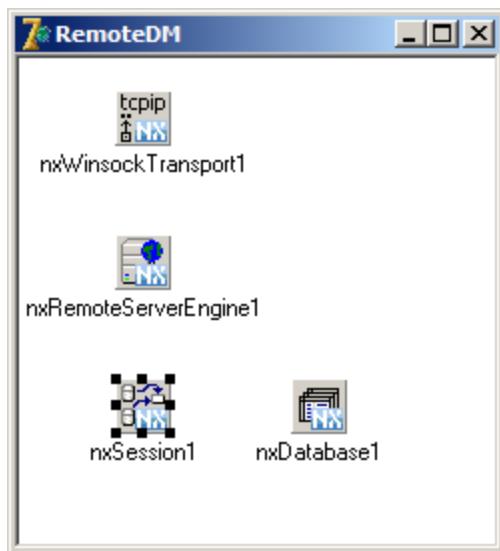
This is almost identical to setting up the data module for embedded applications. Note that there is no need to include the TnxseAllEngines component as this is a client application - the engine registration needs to be performed on the server side only. For the examples we will set up a data module that uses TCP, the most common transport option, to connect to the server.

Hint

If you don't want to create this datamodule on your own you can find the complete source in the **Examples\Manual\Repository** subdirectory of your NexusDB installation. If you don't have this directory you download the examples from our webpage.

Step 1: Creating the data module

Create a new data module in Delphi by selecting **File/New/Data Module** from the Delphi main menu. Locate a **TnxWinsockTransport**, **TnxRemoteServerEngine**, **TnxSession** and **TnxDatabase** component from the NexusDB tab on your component palette and drop them on the form. The data module should now look similar to this:



Step 2: Setting up the Transport

Select **nxWinsockTransport1** and make sure that **Mode** is set to **nxtmSend**. Then set the **ServerNameRuntime** property to **nexusdb@localhost** if the server was started on the same machine and called NexusDB (or **ServerName@ipaddressofserver** (e.g. **NexusDB@192.168.1.1**) otherwise). If you want to see data at designtime set the **ServerNameDesignTime** to the same value.

RespondToBroadcasts	False
ServerNameDesigntime	nexusdb@localhost
ServerNameRuntime	nexusdb@localhost
ServerThreadPriority	tpNormal
Tag	0

Step 3: Connecting the Remote Server Engine

Connect the remote server engine to the Transport component by simply double-clicking the **Transport** property or select from the drop-down list.

Name	nxRemoteServerEngine1
RemoteThreadPri	NORMAL
Tag	0
Transport	nxWinsockTransport1
Version	2.0000 Release Candidate 2 [D]

Step 4: Setting up the session

Connect the nxSession1 component to your NexusDB RemoteServerEngine1 component (pull down list of the Server Engine property). The properties of your session should now look as shown.

Password	
PasswordRetries	3
ServerEngine	nxRemoteServerEngine1
Tag	0
Timeout	-1

Step 5: Setting up the database

Connect the nxDatabase1 component to the Session component (from the pull down list). Finally, you will need to point the AliasName (not the AliasPath) property to where your data tables reside, if you followed the server setup steps earlier then this is the alias **Northwind**. The property Inspector for your database component should now look pretty much the same as the one shown.

ActiveRuntime	False
AliasName	Northwind
AliasPath	
Default	False
DisplayCategory	
DisplayName	nxDatabase1
Enabled	True
EventLog	
EventLogEnabled	False
Exclusive	False
FailSafe	False
Name	nxDatabase1
ReadOnly	False
Session	nxSession1
Tag	0

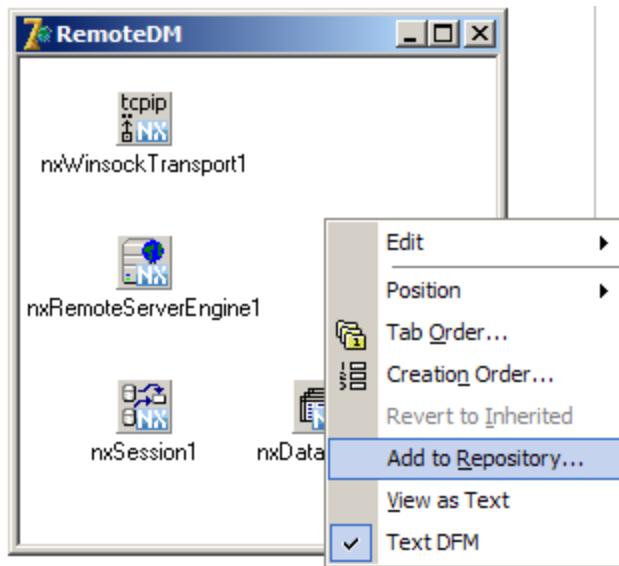
Step 6: Adding the data module to the repository

As the last step we're adding the new data module to the Delphi repository so that we don't need to recreate it all the time. For this first save the data module, eg. by pressing CTRL-S. Save the datamodule to a directory which you want to use as a repository in the future.

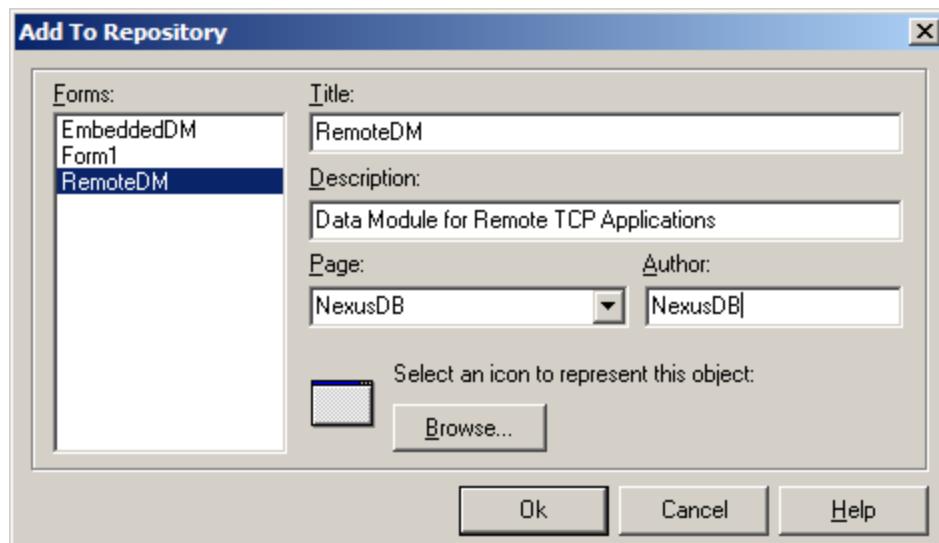
Caveat

Make sure to not delete or move the files later, otherwise you won't be able to use it anymore. When done, find an empty spot on the data module

and right-click. In the popup menu select **Add to Repository**.



This brings up the following dialog. Please fill out the fields like this and press **OK**.



You are now ready to go! It's that straightforward!



13.3.3 Creating a simple databound form

Creating a simple databound form



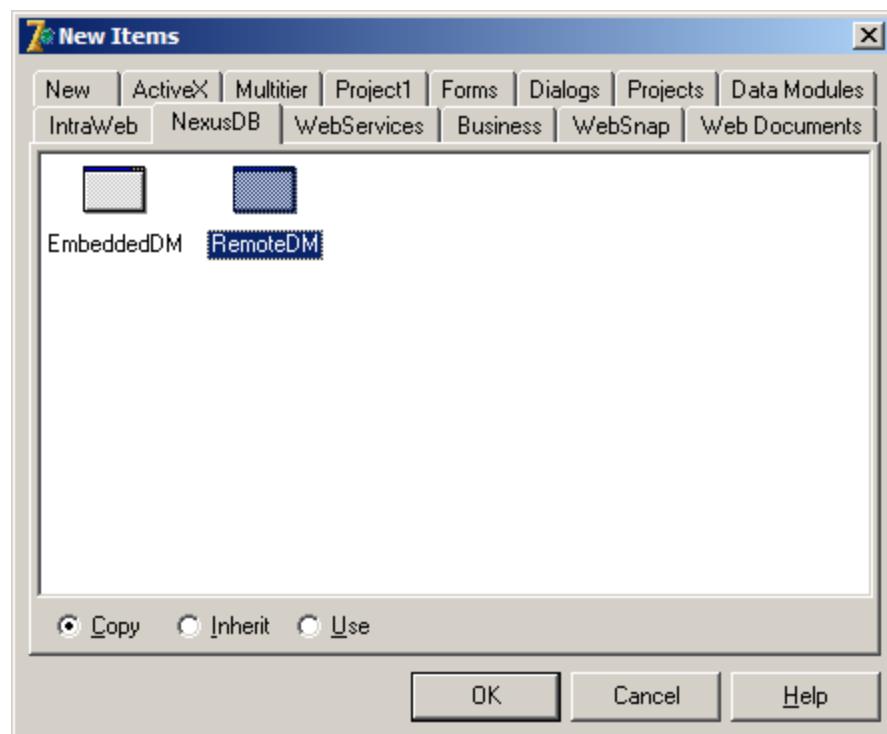
Here we will show how to build a very simple database application using the embedded data module we just created. In the Delphi IDE you will need to create a new project.

Hint

If you don't want to create this example on your own you can find the complete source in the **Examples\Manual\SimpleRemote** subdirectory of your NexusDB installation. If you don't have this directory you download the examples from our webpage.

Step 1: Adding the data module

Select **File/New/Other...** from the Delphi main menu. This will popup the **New Items** dialog where you can find the **NexusDB** tab. Select it, focus the **RemoteDM** entry, make sure **Copy** is activated.



Pressing **OK** will add a copy of the data module to your application.

Step 2: Adding a table

Locate and drop on a **TnxTable** on the data module. To connect to the database available in the data module, just connect the Database property via the drop down list (the correct Session will be set for you). Finally select the table for this instance in the TableName property. Your **nxTable1** component should now look as follows (after selecting a table).

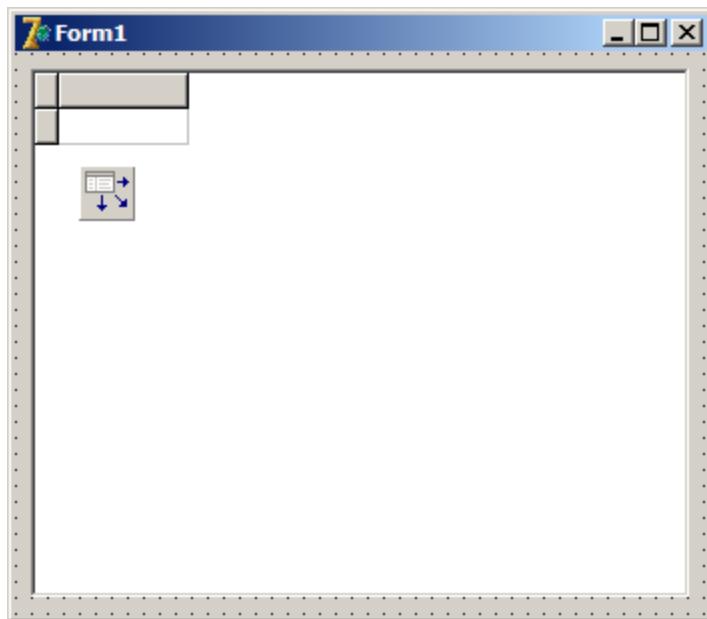
AutoCalcFields	True
BlockReadOpti	[gboBlobs,gboBookr
Database	nxDatabase1
Exclusive	False
FileID	[FILE_ID]

ReadOnly	False
Session	nxSession1
StoreDefs	False
TableName	Customers ▾
Tag	0

Now save the new data module as embeddeddm.pas to the new project directory.

Step 3: Creating a form

Create a new form and drop a **TDatasource** (Data Access tab in the component Palette) and a **TDBGrid** (Data Controls) on the form. Arrange them similar to this:



Select the DBGrid and set its **Anchors** property to [akLeft,akTop,akRight,akBottom]; this makes sure that we can resize the grid with the form.

Step 4: Hooking up the data

Make sure you've saved the data module above, then add the data module to the uses clause in the interface, which will add the components on it to the drop down lists of the current form. Connect the **Dataset** property of the Datasource to the table

AutoEdit	True
DataSet	RemoteDM.nxTable1 ▾
Enabled	True
Name	DataSource1
Tag	0

and set the **Datasource** property of the DBGrid to our datasource.

Ctl3D	True
Cursor	crDefault
DataSource	DataSource1
DefaultDrawing	True
DragCursor	crDrag

The only thing left to do is to make sure the tables are opened when starting the app.

Step 5: Opening the table

For this purpose simply set the **ActiveRuntime** property of the table to true. That's it. Compile the app and run.



Congratulations. You've just created your first NexusDB C/S database application!

Hint

When using C/S mode of Nexus you can have **ActiveDesignTime** enabled. This will allow you to see your data at design time, which in most cases makes development easier.

Caveat

Since we are relying on Delphi creating the datamodule make sure that the datamodule is automatically created **before** the form is created. Check the forms tab of the project options!



13.3.4 Making the application server independent

Making the application server independent



Let's add a run-time enhancement to our data module so that the resulting exe is not restricted to the path coded into the AliasPath property of the database component. That is, let's make the application automatically change this to point to the same path as the exe itself (here we are assuming that the data tables will reside in the same directory as the exe).

Please note that for this to work you need to set the ActiveRuntime properties of the database and tables to false.

To have the tables created in the same directory as where the exe resides, plus to activate the NexusDB component chain, add the following code to the OnCreate event handler of the form (or data module if you used one) as follows:

```
procedure
  TMainformDialog.FormCreate(S
  ender: TObject);
begin
  EmbeddedDM.nxDataBase1.Ali
  asPath:=ExtractFilePath(para
  mStr(0));
  EmbeddedDM.nxTable1.Active
  :=True;
end;
```

That's it. When the program starts, the engine will look for the tables in the same directory as the executable.

Recommendation

We recommend to keep your data in a separate subdirectory of the application which can easily be achieved by modifying the above to, for example,

```
EmbeddedDM.nxDataBase1.Alias
Path:=extractFilePath(params
tr(0))+'\data';
```

In the same way you can set all properties of the database components at start-up. Just make sure that the components have the ActiveRuntime property set to false, if you want to use code changes.

Caveat

For the application to start successfully the data directory **and** the table must exist!

Hint

If you don't want to create this example on your own you can find the complete source in the **Examples\Manual\RunRemote** subdirectory of your NexusDB installation. If you don't have this directory you download the examples from our webpage.



13.4 Mixing Embedded and C/S Applications

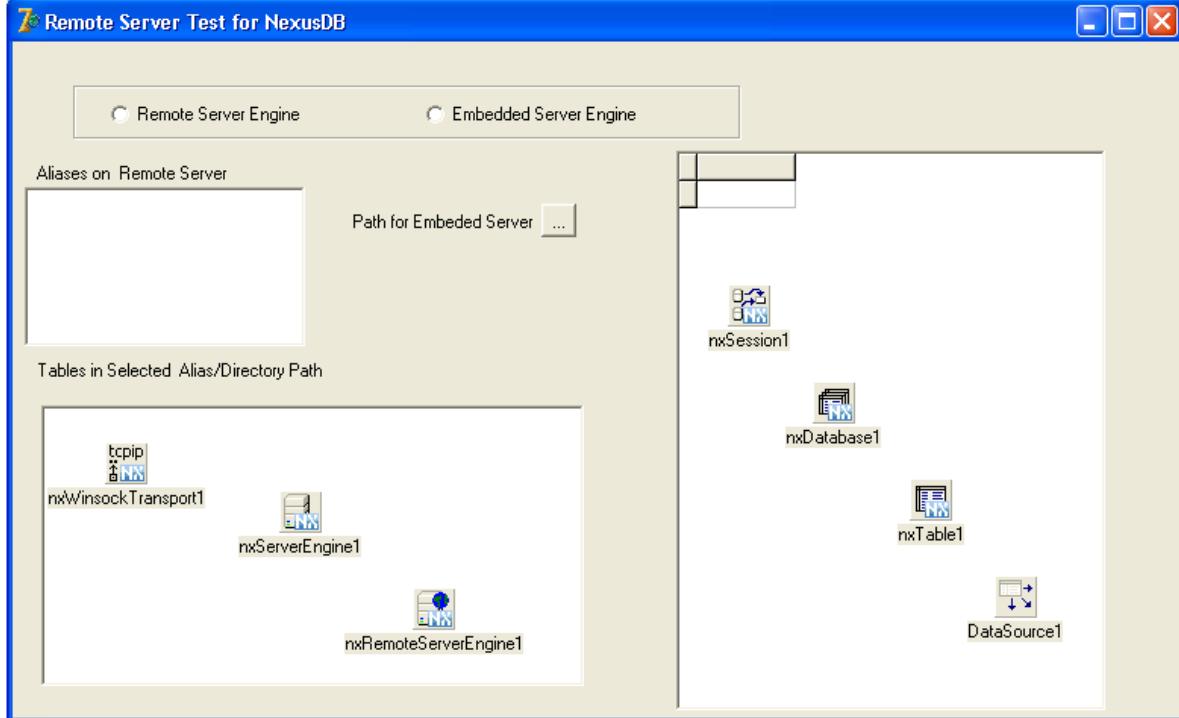
Mixing Embedded and C/S Applications



In this section we create an application that holds both embedded server and remote server engines. A TRadioGroup is used to switch between tables associated with the 2 different engines.

Start the NexusDB Server on your workstation and ensure that the Winsock transport is started and an appropriate alias has been created. Find more information about the NexusDB Server in section 5.

By now you are an expert at whipping up NexusDB applications. So create the application form as shown below:



Please enter the following code. Note that the top ListBox is ListBox1 and the lower (bigger) one is ListBox2. The other components follow easily from their usage in the code.

```
procedure
TMainformDialog.InitCommsEng
ine;
begin
  ListBox1.Clear;
  try
    nxSession1.Open;
    LoadAliases;
  except
    on E:Exception do
    begin
      MessageDlg(E.Message,m
tError,[mbOk],0);
      end;
    end;
  end;

procedure
TMainformDialog.LoadAliases;

var
  Aliases:TStringList;
begin
  Aliases:=TStringList.Creat
e;
  try
    nxSession1.GetAliasNames
(Aliases);
    ListBox1.items.assign(al
iases);
  finally
    Aliases.Free;
  end;
end;

procedure
TMainformDialog.LoadTables;

var
  Tables:TStringList;
begin
  Tables:=TStringList.Create
;
  try
    nxDatabase1.GetTableName
s(Tables);
    ListBox2.items.assign(Ta
bles);
  finally
    Tables.Free;
  end;
end;

procedure
TMainformDialog.ListBox1Db1C
```

```
lick(Sender: TObject);
begin
    nxDatabase1.Connected:=False;
    nxDatabase1.AliasName:=List
    Box1.Items[ListBox1.ItemInd
    ex];
    nxDatabase1.Connected:=True;
    LoadTables;
end;

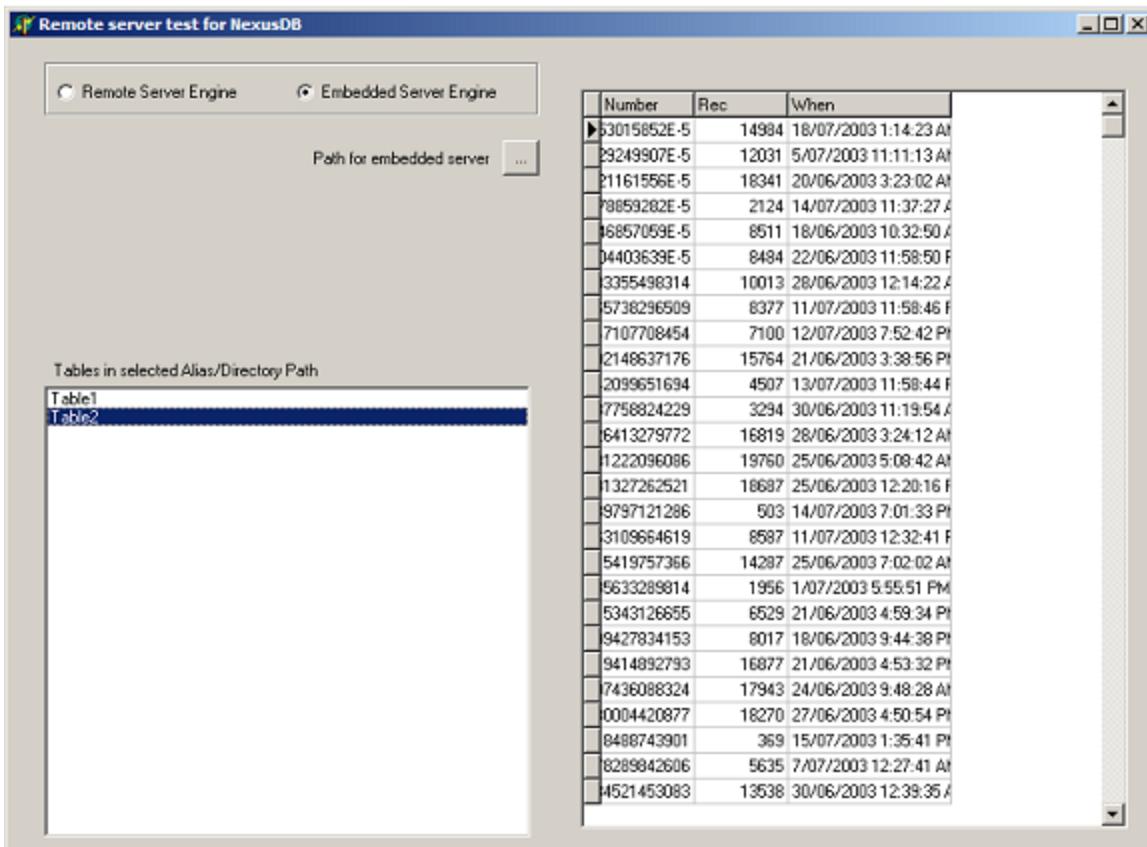
procedure
TMainformDialog.RadioGroup1C
lick(Sender: TObject);
begin
    nxSession1.Connected:=false;
    if RadioGroup1.ItemIndex=0
    then
        begin
            Button1.Hide;
            Label3.Hide;
            Label4.Hide;
            nxSession1.ServerEngine:=
nxRemoteServerEngine1;
            InitCommsEngine;
            ListBox1.Show;
            Label2.Show;
        end else
        begin
            ListBox1.Hide;
            Label2.Hide;
            nxSession1.ServerEngine:=
nxServerEngine1;
            Button1.Show;
            Label3.Show;
            Label4.Hide;
            Label4.Caption:='';
        end;
    nxSession1.Connected:=True
;
end;

procedure
TMainformDialog.Button1Click
(Sender: TObject);
var
    aPath:String;
begin
    aPath:=ExtractFilePath(par
    amStr(0));
    if SelectDirectory(aPath,
    [sdAllowCreate,sdPerformCrea
    te,sdPrompt],SELDIRHELP)
    then
```

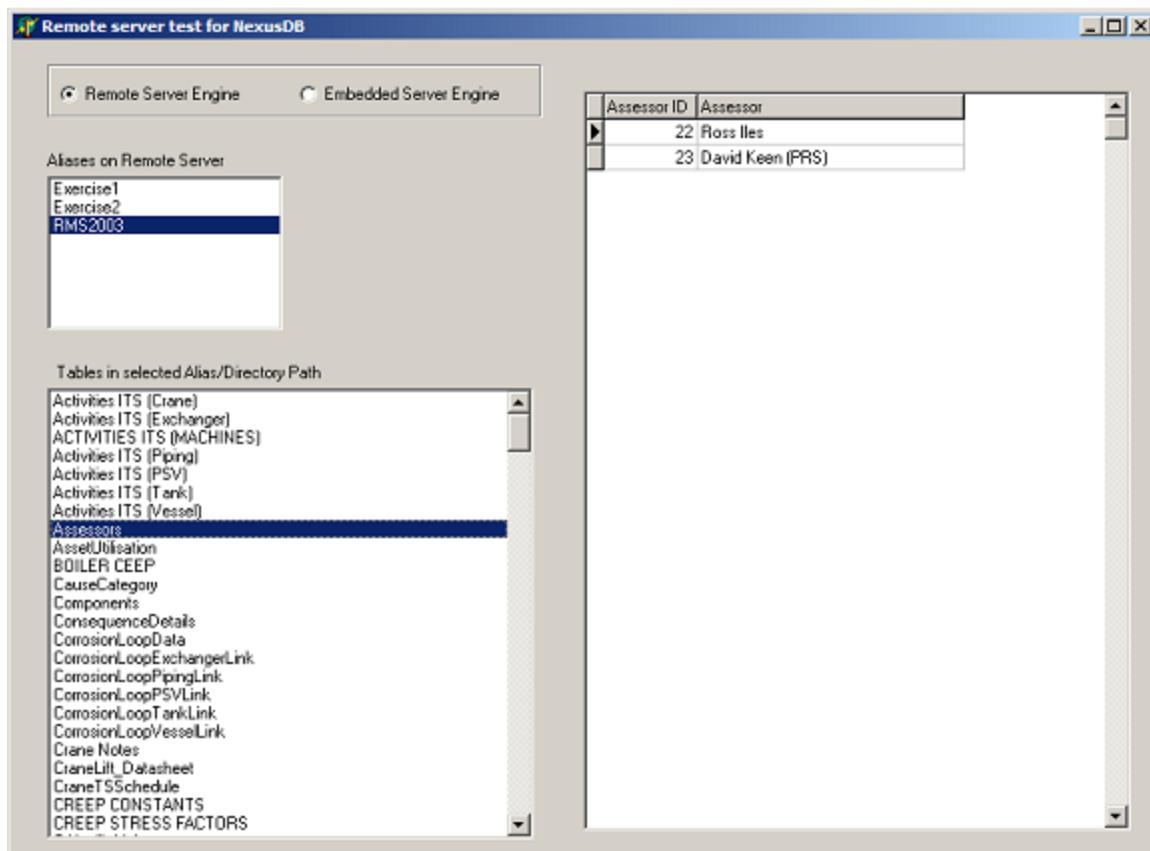
```
begin
  with nxDataBase1 do
    begin
      Connected:=false;
      AliasPath:=aPath;
      Connected:=true;
    end;
    LoadTables;
  end;
end;

procedure
TMainformDialog.ListBox2Db1Click(Sender: TObject);
begin
  With nxTable1 do
    begin
      Close;
      TableName:=ListBox2.Items
      [ListBox2.ItemIndex];
      Open;
    end;
end;
```

Once the application starts up, select the mode you want by pressing the correct radio button. Depending on your selection you will end up with either



or



The technique shown above gives you full control over where to store data. You can for example cache data on client side to avoid slow lookups over the network or you can divide the data into private and public parts. A good use of this is to copy lookup tables from a remote server into a local server engine (and usually as in-memory tables for extra speed) on the assumption that the data in the lookup tables change rarely if at all.



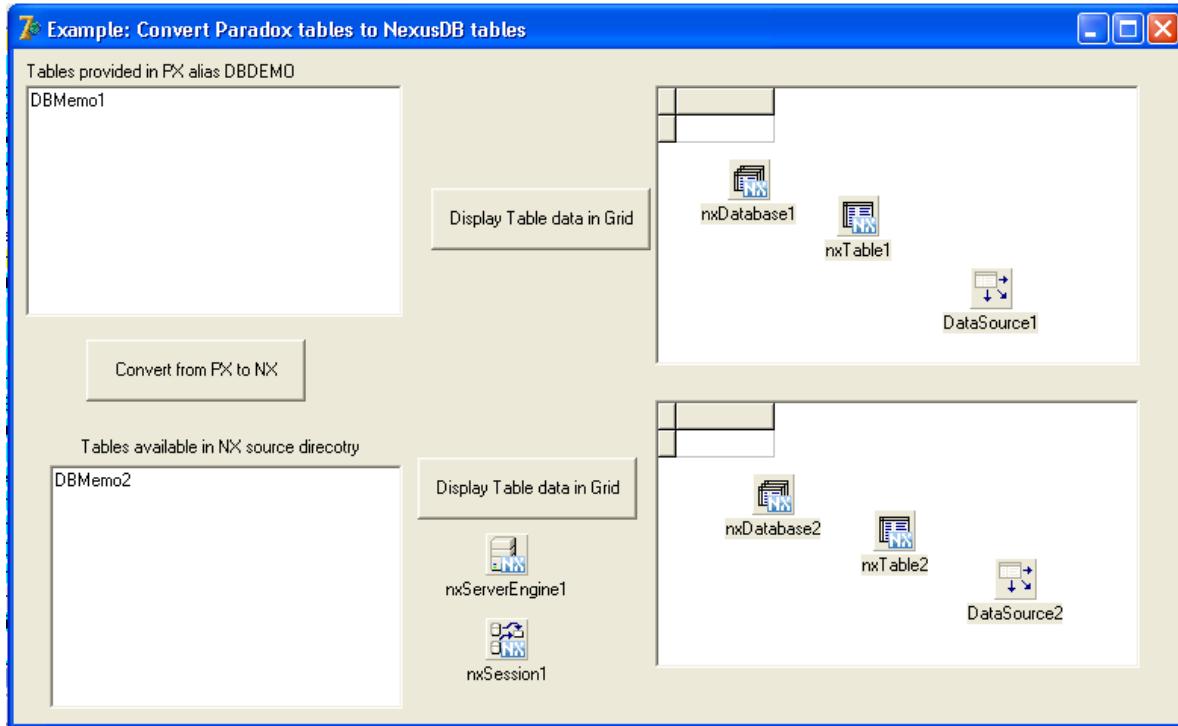
13.5 Converting Paradox Tables to NexusDB Tables

Converting Paradox Tables to NexusDB Tables



This small project is designed to give you a quick method for converting Paradox tables to NexusDB tables. This will be useful if you need to do some special processing during conversion (e.g., delete or process records on during the conversion process) which you wish to keep separate from the Converter Utility that comes with the full NexusDB install system.

Create a new project with a single form as shown below. Note that the DB components at the top are all from the BDE tab of the VCL. Connect up the components: the NexusDB components as per Section 6 above and the BDE components as per the Delphi help (if you need a reference).



Connect up the following code to the appropriate button click events:
Ancillary Code:

```

procedure
TMainformDialog.pxLoadTables
;
var
Tables:TStringList;
begin
Tables:=TStringList.Create
;
try
Database1.GetTableNames(
Tables);
ListBox2.Items.Assign(Ta
bles);
finally
Tables.Free;
end;
end;

procedure
TMainformDialog.nxLoadTables
;
var
Tables:TStringList;
begin
Tables:=TStringList.Create
;
try

```

```
    nxDatabase1.GetTableName
  s(Tables);
    ListBox1.Items.Assign(Ta
bles);
    finally
      Tables.Free;
    end;
  end;

procedure
TMainformDialog.FormCreate(S
ender: TObject);
begin
  DataBase1.Connected:=True;

  nxDataBase1.AliasPath:=Ext
ractFilePath(ParamStr(0));
  nxDataBase1.Connected:=Tru
e;
  pxLoadTables;
  nxLoadTables;
end;
Top Button (Display PX):
procedure
TMainformDialog.Button4Click
(Sender: TObject);
begin
  If ListBox2.ItemIndex<0
then ShowMessage('No Table
Selected') else
  begin
    Table1.Close;
    Table1.TableName:=ListBo
x2.Items[ListBox2.ItemIndex]
+'.'DB';
    if Table1.Exists then
Table1.Open else
ShowMessage('Sorry - that is
not a Paradox table!');
  end;
end;
```

Middle Button (Convert):

```
procedure
TMainformDialog.Button3Click
(Sender: TObject);
var
  i:integer;
  ok:Boolean;
begin
  If Not Table1.Active then
ShowMessage('No table is
open') else
  begin
    with nxTable1 do
    begin
```

```

        Close;
        FieldDefs.Clear;
        FieldDefs.Assign(Table
1.FieldDefs);
        IndexDefs.Clear;
        IndexDefs.Assign(Table
1.IndexDefs);
        TableName:=System.Copy
(Table1.TableName,1,System.P
os('.DB',Table1.TableName)-
1);
        if Exists then
OK:=MessageDlg('Table
already exists.
Replace?',mtConfirmation,
[mbYes,mbNo],0)=mrYes else
OK:=True;
        if ok then
begin
        CreateTable;
        Open;
        with Table1 do
begin
        first;
        while not eof do
begin
        nxTable1.Insert;

        for i:=0 to
Pred(FieldCount) do
            nxTable1.Fields[
i].Assign(Fields[i]);
            nxTable1.Post;
            next;
            end;
            end;
            end;
            nxLoadTables;
end;

```

Bottom Button (Display NX):

```

procedure
TMainformDialog.Button2Click
(Sender: TObject);

begin
  If ListBox1.ItemIndex<0
then ShowMessage('No Table
Selected') else
begin
  nxTable1.Close;

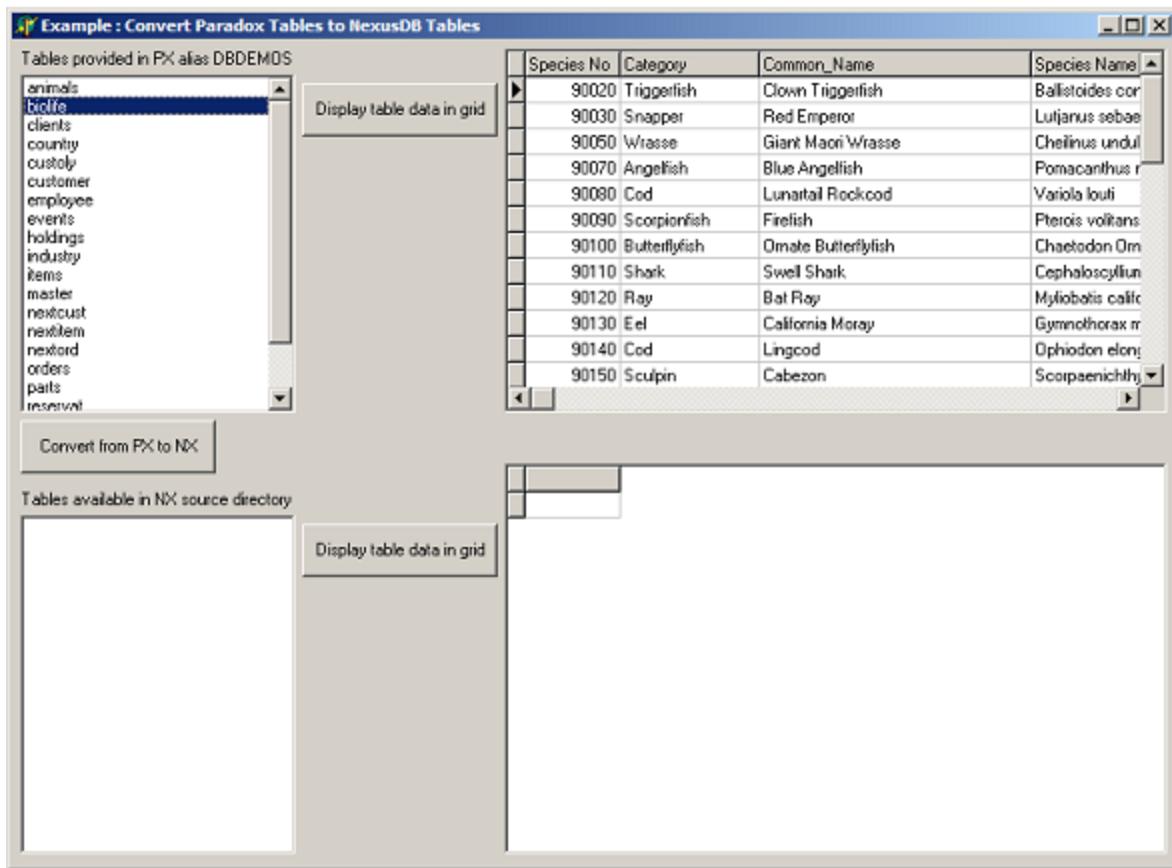
```

```

nxTable1.TableName:=List
Box1.Items[ListBox1.ItemIndex];
nxTable1.Open;
end;

```

When you compile and run this button and then double-click, for example, the biolife table, you should see the following on your screen:



If you now click the Convert button the following screen will be displayed.

Example : Convert Paradox Tables to NexusDB Tables

Tables provided in PX alias DBDEMOS			
Species No	Category	Common_Name	Species Name
90200	Parrotfish	Redband Parrotfish	Sparisoma aurata
90210	Barracuda	Great Barracuda	Sphyraena barracuda
90220	Grunt	French Grunt	Haemulon flavolineatum
90230	Snapper	Dog Snapper	Lutjanus joculator
90240	Grouper	Nassau Grouper	Epinephelus striatus
90250	Wrasse	Bluehead Wrasse	Thalassoma bifasciatum
90260	Jack	Yellow Jack	Caranx ignobilis
90270	Surferch	Redtail Surferch	Amphistichus retrocurvus
90280	Croaker	White Sea Bass	Atractoscion nobilis
90290	Greenling	Rock Greenling	Hexagrammos otakii
90300	Wrasse	Senorita	Oxyjulis californica
90310	Smelt	Surf Smelt	Hypomesus pacificus

Convert from PX to NX

Tables available in NX source directory			
Species No	Category	Common_Name	Species Name
90020	Triggerfish	Clown Triggerfish	Balistoides conspicillum
90030	Snapper	Red Emperor	Lutjanus sebae
90050	Wrasse	Giant Moray Wrasse	Chelirinus undulatus
90070	Angelfish	Blue Angelfish	Pomacanthus annularis
90080	Cod	Lunatal Rockcod	Variola louti
90090	Scorpionfish	Firefish	Pterois volitans
90100	Butterflyfish	Oriente Butterflyfish	Chaetodon orientalis
90110	Shark	Swell Shark	Cephaloscyllium laticeps
90120	Ray	Bat Ray	Myliobatis californica
90130	Eel	California Moray	Gymnothorax griseus
90140	Cod	Lingcod	Ophiodon elongatus
90150	Sculpin	Cabezon	Scorpaenichthys marmoratus
90160	Spadefish	Atlantic Spadefish	Chaetodipus faber

This example has shown you, in a very quick way, how to read all the tables from a particular directory (or alias) into a list box as well as how to create a NexusDB copy of an existing Paradox table, including transferring the data across.



13.6 How to create Tables?

How to create Tables?



There are many ways to create tables in NexusDB. Here are the ways you can create tables:

- Direct Data Dictionary code
- TFieldDefs
- SQL DDL
- Enterprise Manager



13.6.1 Creating Tables Using the Data Dictionary

Creating Tables Using the Data Dictionary



As discussed in the Data Dictionary Section, the Data Dictionary is NexusDB's native object structure for storing structural information. This is the most direct, powerful and efficient way to create tables in NexusDB. Here is an example of creating a "Products" table:

```
procedure
TDataModuleMain.CreateProductTable(aNexusDB:
TnxDatabase);
var
  ProductsDict:
TnxDataDictionary;
  NewIndex:
TnxIndexDescriptor;
  FieldDesc:
TnxFieldDescriptor;
  KeyFieldDesc:
TnxKeyFieldDescriptor;
begin
  ProductsDict := 
TnxDataDictionary.Create;
try
  ProductsDict.FieldsDescriptor.AddField('PRODUCTID',
'', nxtAutoInc, 0, 0, True);

  ProductsDict.FieldsDescriptor.AddField('NAME', '',
nxtNullString, 64, 0, True);

  ProductsDict.FieldsDescriptor.AddField('RETAILPRICE'
, '', nxtCurrency, 0, 0,
True);
  ProductsDict.FieldsDescriptor.AddField('WHOLESALEPRI
CE', '', nxtCurrency, 0, 0,
True);
  ProductsDict.FieldsDescriptor.AddField('MANUFACTURER
ID', '', nxtWord32, 0, 0,
True);
  ProductsDict.FieldsDescriptor.AddField('PRODUCTNUMBE
R', '', nxtNullString, 64,
0, False);
  ProductsDict.FieldsDescriptor.AddField('DESCRIPTION'
, '', nxtBlobMemo, 0, 0,
False);
```

```

        ProductsDict.FieldsDescriptor.AddField('THUMBNAIL',
        '', nxtBlobGraphic, 0, 0,
        False);
        ProductsDict.FieldsDescriptor.AddField('PRODUCTIMAGE
        ', '', nxtBlobGraphic, 0, 0,
        False);
        FieldDesc :=
        ProductsDict.FieldsDescriptor.AddField('CREATED', '',
        nxtDateTime, 0, 0, True);
        FieldDesc.AddDefaultValue(TnxCurrentDateTimeDefaultValueDescriptor);
        ProductsDict.FieldsDescriptor.AddField('MODIFIED',
        '', nxtDateTime, 0, 0,
        True);

        NewIndex :=
        ProductsDict.IndicesDescriptor.AddIndex('PK_PRODUCTID',
        0, False,
        'Primary Key of
        Products Table',
        TnxCompKeyDescriptor);
        TnxCompKeyDescriptor(NewIndex.KeyDescriptor).Add(
        ProductsDict.GetFieldFromName('PRODUCTID'));

        NewIndex :=
        ProductsDict.IndicesDescriptor.AddIndex('I_NAME', 0,
        True,
        'Name Index',
        TnxCompKeyDescriptor);
        KeyFieldDesc :=
        TnxCompKeyDescriptor(NewIndex.KeyDescriptor).
        Add(ProductsDict.GetFieldFromName('NAME'),
        TnxTextKeyFieldDescriptor);

        TnxTextKeyFieldDescriptor(KeyFieldDesc).IgnoreCase
        := True;

        NewIndex :=
        ProductsDict.IndicesDescriptor.AddIndex('I_MANUFACTURERID',
        0, True,
        'Manufacturer Index',
        TnxCompKeyDescriptor);
        TnxCompKeyDescriptor(NewIndex.KeyDescriptor).Add(ProductsDict.GetFieldFromName('MANUFACTURERID'));

```

```

        NewIndex :=
ProductsDict.IndicesDescriptor
or.AddIndex('I_PRODUCTNUMBER
', 0, True,
            'Product Number
Index',
TnxCompKeyDescriptor);
TnxCompKeyDescriptor(New
Index.KeyDescriptor).Add(
ProductsDict.GetFieldF
romName('PRODUCTNUMBER'));

aNexusDB.CreateTable(true
e, 'PRODUCTS', '',
ProductsDict);
finally
    FreeAndNil(ProductsDict)
;
end;
end;

```



13.6.2 Creating Tables Using FieldDefs

Creating Tables Using FieldDefs



This is the standard Delphi way of creating tables. NexusDB supports this method of table creation completely. Create TFieldDefs and TIndexDefs and simply invoke the TnxTable's CreateTable method. NexusDB also supports a CreateTableEx method which requires the block size.

Remember to set the StoreDefs property to True if storage of TFieldDefs and TIndexDefs is required.



13.6.3 Creating Tables Using SQL DDL

Creating Tables Using SQL DDL



NexusDB supports SQL DDL, which means that you may Create Tables and Indices through SQL. Here are SQL DDL statements to create the Products Table:

```
CREATE TABLE PRODUCTS
```

```

(
    PRODUCTID AUTOINC NOT
    NULL PRIMARY KEY,
    NAME VARCHAR(64) NOT
    NULL,
    RETAILPRICE MONEY NOT
    NULL,
    WHOLESALEPRICE MONEY NOT
    NULL,
    MANUFACTURERID INTEGER
    NOT NULL,
    PRODUCTNUMBER VARCHAR(64)
    NULL,
    DESCRIPTION TEXT,
    THUMBNAIL IMAGE,
    PRODUCTIMAGE IMAGE,
    CREATED DATETIME DEFAULT
    CURRENT_TIMESTAMP NOT NULL,

    MODIFIED DATETIME NOT
    NULL
);

CREATE INDEX I_NAME ON
PRODUCTS (NAME IGNORE CASE);

CREATE INDEX
I_MANUFACTURERID ON PRODUCTS
(MANUFACTURERID);

CREATE INDEX I_PRODUCTNUMBER
ON PRODUCTS (PRODUCTNUMBER);

```

By using standard SQL data types, the Products table that is created from the SQL DDL, is not exactly the same as the one created by the Data Dictionary code. There are four differences:

- The NAME and PRODUCTNUMBER columns are of type nxtShortString because they are of type VARCHAR with a length less than 256. The Data Dictionary code sets the NAME and PRODUCTNUMBER columns as type nxtNullString. In SQL the nxtNullString type may be forced by declaring the type as NULLSTRING.
- The MANUFACTURERID is an nxtInt32, which is a signed 32-bit integer rather than the nxtWord32 type (unsigned 32-bit integer) that the Data Dictionary code produces. In SQL, the nxtWord32 type can be used by declaring the type as DWORD.
- The Primary Key index that is created by the PRIMARY KEY specifier on the PRODUCTID column creates an auto named primary key (e.g. key0). The Data Dictionary code names this index "PK_PRODUCTID".
- Descriptions for the Indices are not set in the SQL DDL.

The full SQL DDL is covered in detail in the SQL reference chapter. **Using SQL DDL is the most database neutral Table creation method.**



13.6.4 Creating Tables Using the Enterprise Manager

Creating Tables Using the Enterprise Manager



Tables can be created using the Enterprise Manager. The Enterprise Manager is covered in depth in the Management Tools books of this manual.



13.6.5 Thoughts about In-Memory Tables

Thoughts about In-Memory Tables



Nexus memtables are a bit different from other memtable implementations. That often leads to a bit of misunderstanding in the beginning as it is unlike e.g. kbmMemTable.

It is easier to think of NX memtables as usual tables that are lost whenever you stop the server process. Like usual tables Nexus memtables are created by the server when you tell it to do so.

This means:

- The table belongs to a database and can therefore be part of transactions with other (mem)tables in that same database.
- The table data is kept at the server and transferred to/from the clients via transports. The transports only come into play with external servers, not with an embedded serverengine. But the principle is the same even in this situation. The server(engine) handles the physical data, be it a file or just some memory blocks.
- Memory tables can have different scopes.
- When using an exposed server the same table can be used by different clients.

To read more about Memory tables please refer to ???todo.



13.7 How to create Indices?

How to create Indices?



NexusDB allows very fine control over Indices by using the data dictionary descriptor instead of the VLC functions.

The default dictionary structure uses these types of objects, linked similarly to a master-detail relationship, for the index support:

```
IndexDescriptors
KeyDescriptors
KeyFieldDescriptors
```



13.7.1 Index Descriptors

Index Descriptors



At the top level, the dictionary object has an `IndexDescriptor` array, which is the list of Indices. Each item in this array is an instance of the `TnxIndexDescriptor` class. An instance of this class contains global index info, typically whether the index allows duplicates, whether it is the default index, and a description.



13.7.2 Key Descriptors

Key Descriptors



Each `IndexDescriptor` object owns a `KeyDescriptor` object. The `KeyDescriptor` classes are responsible for creating the index keys which are used to determine the collation sequence. There are two built-in variants: `TnxRefKeyDescriptor` and `TnxCompKeyDescriptor`. The `TnxRefKeyDescriptor` class is used to generate simple unique keys for the Sequential Access Index, and owns no further child objects. The `TnxCompKeyDescriptor` class is used for Indices composed from one or more fields.



13.7.3 Key Field Descriptors

Key Field Descriptors



A TnxCompKeyDescriptor object owns a list of KeyFieldDescriptors, one for each field in the index. There are three types of KeyFieldDesctiptors: Comp, Text, and ExtText. The TnxCompKeyFieldDescriptor is used for all non-text comparisons, the TnxTextKeyFieldDescriptor is used for non-localized string comparisons, and TnxExtTextKeyFieldDescriptor is used for localized comparisons. See the windows SDK help file's CompareString topic, accessible from your Delphi or C++Builder's Help menu, for more info about the many options available on the localized sort. The KeyFieldDescriptor objects contain ascending or descending sort info for its field, this means that an index can have one field sort ascending, and another one descending.



13.7.4 How to create Indices in Code

How to create Indices in Code



In code, creating and linking the index object structure together works like this: First, add a new IndexDescriptor object to the dictionary's array of IndexDescriptors. The AddIndex call does this for you. Notice that you also specify in this call wether you want a regular field-based index or a SAI index, by passing in the type of the KeyDescriptor object you want. Note that the KeyDescriptor object is also created for you inside this call.

```
Index :=  
Dict.AddIndex('New_Index',  
0, False,  
'Text Description',  
TnxCompKeyDescriptor);
```

Then (assuming it's a field-based index), using the newly constructed object, we add one or more fields to the index:

```
IndexFieldDescriptor :=  
TnxCompKeyDescriptor(Index.K  
eyDescriptor).Add(  
Dict.GetFieldFromName(  
'IndexedField'));
```

Member and thus Index Field Options can now be changed from the defaults, if needed:
IndexFieldDescriptor.IgnoreCase := True;

As mentioned above, using the data dictionary gives you much more flexibility over index creation. If you don't need the fine control that the data dictionary gives you, you can just use TIndexDefs, TFieldDefs and the CreateTable method of TDataset instead. See the Borland VCL help for description and examples.



13.8 How to create a monitor/extender?

How to create a monitor/extender?



NexusDB has an internal event infrastructure that you can hook into by means of Monitors and Extenders. Monitors/Extenders allow to change the default behavior of database events that are fired on particular changes or conditions that occur within NexusDB's core engine. These function very much like triggers but due to its integration into the core of the database engine are actually much more powerful. Monitors and extenders always work together.

Some background information

Monitors get notified on creation of server internal objects such as sessions, databases, cursors, ... At this time the monitor can instantiate extenders and attach them to these objects.

Extenders can be attached to a variety of server internal objects. Once created they get notifications of events occurring within these objects (e.g. the posting of a record in a Cursor object) and can stop, change or extend the default behavior (e.g. disallowing post for certain users).

Monitors and Extenders can be used for many purposes: logging, custom security systems, referential integrity, ...

The event mechanism within NexusDB is extensive and complete; Monitors and Extenders empower you to exploit this mechanism.

An example - Keeping track of last update time of a record

An often needed requirement is to keep track when a record was created or last updated. This sounds easy to implement on the client side on first sight. Just use the OnBeforePost event of the TDataset and change the date. Well, bad luck if you're using a C/S solution that is running clients in different time zones for example. You will end up in a big mess.

Luckily this can be implemented very easily in NexusDB. Here's how.

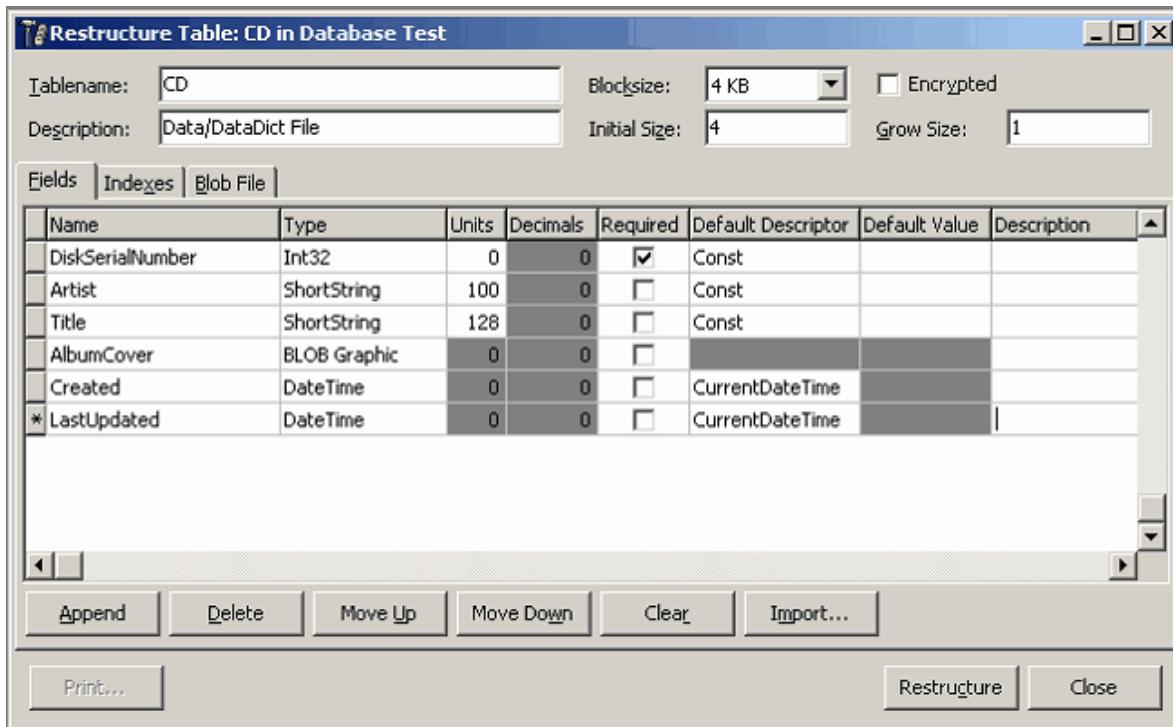
Steps needed:

- Add a Created and LastUpdated field with type TDateTime to the table
- Create a Monitor/Extender combo
- Implement the methods for the Monitor class
- Implement the methods for the Extender class
- Add an instance of the Monitor to the Server, set the properties and recompile the server

Add a Created and LastUpdated field with type TDateTime to the table

Now that's pretty simple and straightforward, isn't it? If you haven't already, please download and unzip the Demo Database into a directory of your choice. Then start the NexusDB Enterprise Manager (nxEnterpriseManager.exe). In the Servers window double click the Internal Server entry. The databases you've added in earlier sessions will appear now. Select the database pointing to the Demo Database directory or if that doesn't exist, add a new Database by clicking the right mouse button and

selecting New Database Alias. Open the database by double clicking and press the right mouse button on the CD Table. In the menu select the Redefine item. In the dialog window that appears please add the two fields as shown here:



Please note the selection of the CurrentDateTime descriptors in the Default Descriptor column. This will for all new records set both fields to the Server time of the record creation. In other words that means that we don't have to bother with setting the Created date manually for newly added records.

Hint: Default Value Descriptors

In NexusDB every field is set to a certain default value when a new record is created and is in the default setting the constant NULL. When defining the structure of a table the developer can assign a different value as Default Value to each field or can assign a different Default Value descriptor. Readily available descriptors are Const and CurrentDateTime. The developer can implement his own descriptors and register them into the NexusDB server if required.

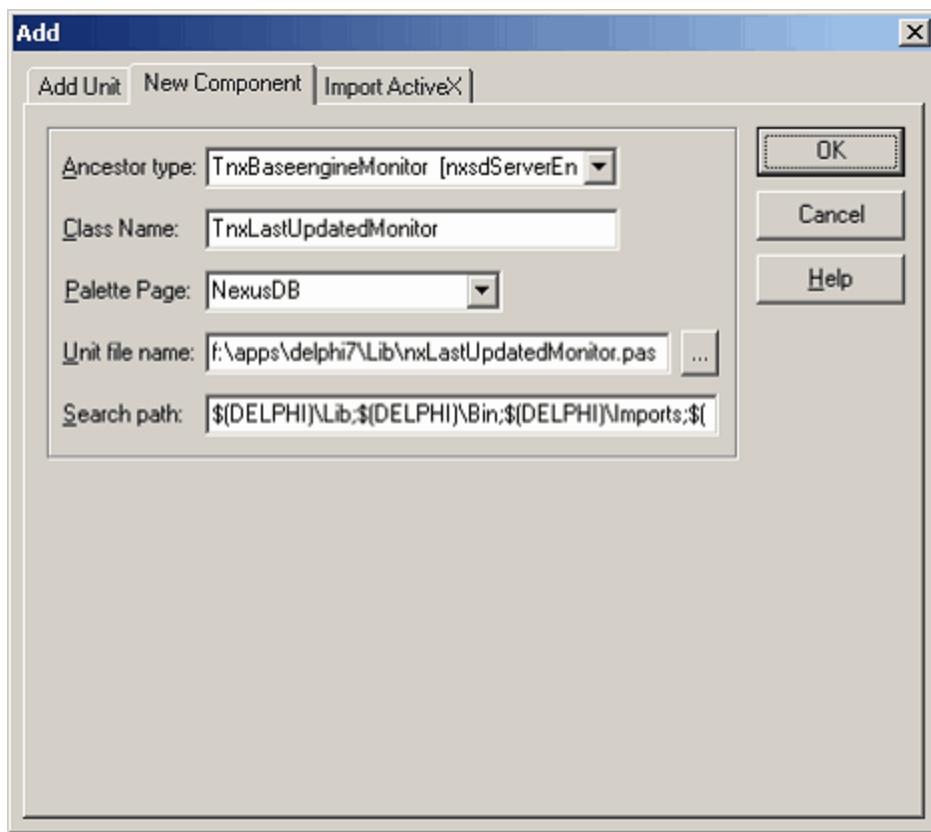
Please note that the default values will be applied to newly added fields on restructure of the table. They are not applied to already existing fields, even if they are NULL.

When you have added the new fields press the Restructure button and repeat the same for the Tracks table. The two tables now have these new fields which are initialized to the restructure time. We have now prepared the data structure and can commence to step 2.

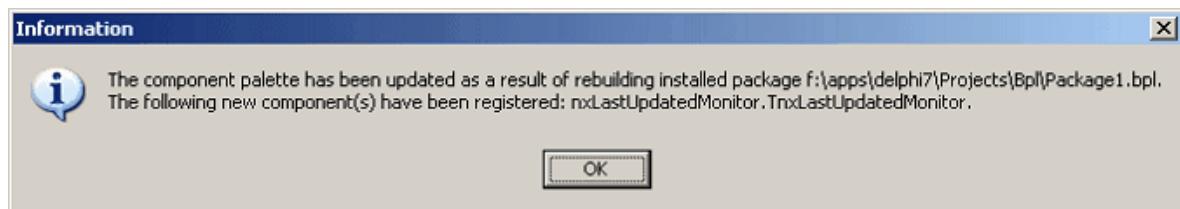
Create a Monitor/Extender combo

As mentioned above a monitors and extenders always work together. So we will now create a monitor that will get notified every time a new NexusDB server object is created. For this we will create a new package so we can register and install it into the Delphi component palette.

Start Delphi, and select New->Other from the File menu. In the New Items dialog select Package from the New tab and press ok. This will create new Delphi package project. Press CTRL-S and save the new project to your liking. Now press the Add button to create a new component. Fill out the New Component tab of the popping up dialog like this:



Press the OK button and Delphi will create new unit including the component registration code for you. Press CTRL-S to save the unit to nxLastUpdatedMonitor.pas. If you compile and install the package now Delphi will add a new component to the NexusDB palette.



We've successfully created and installed our own monitor class, so it's time now to add the code for our own extender.

Implement the methods for the Monitor class

Since every extenders has to be descended from nxBaseEngineExtender, we define our own class descendet from it. For this we add the following code to the interface of nxLastUpdatedMonitor:

```

type
  TnxLastUpdatedExtender =
    class(TnxBaseEngineExtender)

end;

```

As mentioned above, the monitor will later be attached to a ServerEngine and get notified by this engine about every creation of a server object (which are all descendants of TnxExtendableServerObject). This notification is done by calling the ExtendableObjectCreated method of the monitor, which is defined as

```

procedure
ExtendableObjectCreated(aExt
enableObject :
TnxExtendableServerObject);
override;

```

The purpose of a monitor is to attach a certain extender instance to the object created. To do this we simply override above method and create an instance of our TnxLastUpdatedExtender class.

```

procedure
TnxLastUpdatedMonitor.Extend
ableObjectCreated(
  aExtendableObject:
TnxExtendableServerObject);

begin
  inherited;
  // check if the Object
  created was a cursor
  if aExtendableObject is
  TnxAbstractCursor then
    TnxLastUpdatedExtender.C
  reate(self,
  aExtendableObject);
end;

```

As you can see we first check, if the object created at the server is a TnxAbstractCursor descendant. As we want to extend inserting and updating of records, we only want to attach our extender to cursors, and not to e.g. sessions.

Hint: Make sure to attach your server to the needed server objects only!

An extender is typically only interested in a small subset of all available notification events. Attaching extenders to objects that never will trigger the needed notifications is unnecessary and will influence the performance of the whole server (if used excessively). As a consequence always think about what you want to achieve and which objects you want to attach to. The most important ones are

- TnxAbstractCursor - for all record based notification

- TnxAbstractDatabase - for all transaction and table notifications
- TnxAbstractSession - for all settings and Alias notifications

We now have a system in place, that will monitor the creation of server objects and if a cursor is created we attach an instance of our extender to it. So all that is left is the actual implementation of the notification handler.

Implement the methods for the Extender class

Let's look at the methods of TnxBaseEngineExtender and we find

```
function Notify(aAction :  
TnxEngineAction; aBefore :  
Boolean;  
const aArgs : array of  
const) : TnxResult;  
override;
```

This function is called for every functional call inside the server objects. Since our extender is attached to a cursor this can be eaRecordGet, eaRecordInsert, eaRecordModify or eaRecordDelete. For a complete list of notifications and the arguments, please look here. Back to our notification handler - what do we have to do:

- first we check for the correct notification id
- then we will check if we are dealing with a table
- we look for the LastUpdate field and if the field is of the correct type (DateTime)
- then we need to convert the current date & time to the internal NexusDB storage format
- and at last replace the field value in new record buffer.

The source code for above steps looks like this:

```
function  
TnxLastUpdatedExtender.Notif  
y(aAction : TnxEngineAction;  
  
aBefore : Boolean; const  
aArgs : array of const) :  
TnxResult;  
var  
  lFieldIndex : Integer;  
  lLen : Integer;  
  lBuffer : PnxByteArray;  
  lRecordBuffer :  
PnxByteArray;  
  lCursor :  
TnxAbstractCursor;  
  lDateTime : Variant;  
begin  
  Result := DBIERR_NONE;
```

```
// if the action is a
eaRecordModify we will now
set the field LastUpdate
// (if available!) to be
the current datatime. this
field will then
// always hold the
LastChange date.
if ((aAction in
[eaRecordModify]) and
aBefore) then begin
    // since we already know
that the object we are
attached to is a cursor
    // we can safely cast
the object to a
TnxAbstractCursor
    lCursor :=

TnxAbstractCursor(beeExtenda
bleObject);

    // first we check if
this is a table. after all
it could be a specialized
    // cursor that has
nothing to do with our
purpose
    // if yes we also need
to check if this is the
right table
    // if no we've nothing
to do
    if not (lCursor is
TnxServerCursor) then
        exit;

    // next we check if
there IS a field LastUpdated

    // if not we've nothing
to do
    lFieldIndex :=

lCursor.Dictionary.GetFieldF
romName('LastUpdated');
    if lFieldIndex < 0
then
    exit;

    // check for the correct
fieldtype!
    // if wrong we've
nothing to do
    if
lCursor.Dictionary.FieldDesc
riptor[lFieldIndex].fdType
<> nxtDateTime then
        exit;
```

```
        // now we need to
        convert the value we want to
        set to the internal
        // storage value, as we
        are working on low level
        here.
        // for convenience we
        use the VariantToNative from
        the nxSQLProxies unit
        lDateTime := now;
        // get the field
        length
        lLen :=
        lCursor.Dictionary.FieldDescriptor[1FieldIndex].fdLength
        ;
        // get a buffer for the
        internal data
        nxGetMem(lBuffer, lLen);

        try
            VariantToNative(nxtDataTime, lDateTime, lBuffer,
            lLen);

            // get the record
            buffer from the passed
            parameters
            // for eaRecordModify
            the aArgs[0] is a pointer to
            a byte array to the
            // new record buffer
            lRecordBuffer :=
            PnxByteArray(aArgs[0].VPChar
            );

            // finally set the
            field
            lCursor.Dictionary.Set
            RecordField(
                // first get the
                field
                1FieldIndex,
                // pass in the
                passed parameter for the
                record buffer
                lRecordBuffer,
                // pass in the value
                of the field
                lBuffer
                );
            finally
                // free the buffer
                nxFreeMem(lBuffer,
                lLen);
                end;
            end;
        end;
```

I admit it looks long and complex at first sight. Actually it is not, it's just a bit different from what we are used to, but essentially it's the same as doing a FindField followed by a Field.AsDateTime:=Now.

The majority of the code is to check the environment of the notification and if we can apply the new value. The main reason for this being a bit more complicated than the way we're used to is, that we need to use the low level routines of the data dictionary to access fields.

As all fields in NexusDB are stored in a native format we also need to convert the TDateTime value accordingly. As mentioned in the comment we use the VariantToNative function defined in nxSQLProxies. For NexusDB V1 this will add a dependency to the SQL package to our package (for V2 these will be moved to a different place). If you don't want this you need to copy them into our package sources.

To compile the above you need to add nxlTypes, nxlibDE, nxsdTypes, nxsqlProxies, nxsrServerEngine and nxlMemoryManager to the uses clause of our unit. That's it, the Monitor/Extender combo is implemented. Let's use it!

Add an instance of the Monitor to the Server

The rest is trivial. Open the nxServer project in Delphi. Load nxdmServer.pas and drag a TnxLastUpdateMonitor on the data module form.

Set the ServerEngine property of the monitor to the server engine, ActiveRuntime to True (if you want it to auto start by default) and make sure that the Enabled property is True.

Recompile the server and run. You should now see nxLastUpdatedMonitor1 entry in the Database Settings tree view. We now have a server that automatically updates LastUpdated fields (of type DateTime) in all tables.

To test it, just open a table with this field in Enterprise Manager and modify it. Congratulations, you've built your first Monitor/Extender for NexusDB.

Now that you know how to do it you can use that knowledge to extend and/or change the functionality of NexusDB core to your liking.



13.9 How to optimize read/write access?

How to optimize read/write access?



As most of you already know, NexusDB has a very fast engine structure at its heart and performs very well out of the box. Its flexibility and tuning options still allow lots of improvement though. This short article shows you the impact of different settings for write access to your data, especially bulk insert speed.

The basic insert loop

Let's start with the typical piece of code to insert records into a table.

```

nxTable1.EmptyTable;
nxTable1.Open;
StartTime;
for I := 0 to 100000 do
begin    // Iterate
    nxTable1.Append;
    nxTable1.FindField('FileName').AsString:='TestFileName';
    nxTable1.FindField('Directory').AsString:='TestDirectory';
    nxTable1.FindField('FileType').AsString:='TestType';
    nxTable1.FindField('FSize').AsInteger:=1234567890;
    nxTable1.FindField('FDat e').AsDateTime:=now;
    nxTable1.FindField('Attributes').AsInteger:=654321;

    nxTable1.FindField('ParentID').AsInteger:=-1;
    nxTable1.Post;
    if (i mod 10)=0 then
        SetCounter(i);
    end;    // for
    SetCounter(i);
    PrintTime('Base');
nxTable1.Close;

```

The methods `StartTime`, `SetCounter` and `PrintTime` are just simple methods to progress the UI and are of no interest here. As you can see all we do is to insert 100.000 records into a table. We first look at how this performs on an embedded server engine, while we look at remote servers later on.

Let's take a look at the structure of the table. As you can readily see the record size is rather large due to the use of a 1024 byte string field.

The table also has indices defined for all fields except the Directory field.

Running this loop on a current machine (Athlon 2800, 1 Gig Ram, fast 7200 rpm disk system) takes about 61 seconds resulting in roughly 1640 records/sec performance. In other words NexusDB is capable of inserting one record per 0.6 milliseconds. This insert operation includes writing and safely flushing the data to disk, updating the different indices and, again, writing and safely flushing the index blocks to disk as well. Not bad, is it! But can we do better? Sure we can!

Let's look at transactions:

Transactions

A transaction essentially wraps a number of calls into one logical block and executes these as ONE unit. For our example we just need to do some small changes to the code to make explicit use of transactions:

```

nxTable1.EmptyTable;
nxTable1.Open;
StartTime;
nxTable1.Database.StartTransaction;
for I := 0 to 100000 do
begin    // Iterate
    nxTable1.Append;
    nxTable1.FindField('FileName').AsString:='TestFileName';
    nxTable1.FindField('Directory').AsString:='TestDirectory';
    nxTable1.FindField('FileType').AsString:='TestType';
    nxTable1.FindField('FSize').AsInteger:=1234567890;
    nxTable1.FindField('FDate').AsDateTime:=now;
    nxTable1.FindField('Attributes').AsInteger:=654321;

    nxTable1.FindField('ParentID').AsInteger:=-1;
    nxTable1.Post;
    if (i mod aSize)=0
    then
        begin
            nxTable1.Database.Commit;
            nxTable1.Database.StartTransaction;
            SetCounter(i);
        end;
    end;    // for
    nxTable1.Database.Commit;

    SetCounter(i);
    PrintTime('Transaction ('+inttostr(aSize)+'):');
    nxTable1.Close;

```

Note the aSize variable. It is used to define certain sizes (here in number of loops) for the transaction. Using for example aSize=1000 means that the server is caching 1000 loop operations in memory and then executes ALL of them at once to disk. This again heavily reduces write access and flushing to the much slower file system. It is essential for this to work efficiently that the server has enough memory to handle the selected transaction size. (Also when you design your system for multiple users, keep in mind that there may be a number of transactions active at any single point in time, thus affecting overall memory requirements.)

Running our example again with different transaction size yields the following results:

Loops per transaction	Time
1000	11.5 seconds
10000	9.4 seconds
20000	9.7 seconds
100000	10.1 seconds

Now that is quite a lot faster isn't it. For the fastest case we are now able to insert around 10640 records/sec. That's an average of 0.09 milliseconds per record.

Please note that a larger transaction size does not always mean faster execution times. As the block organization complexity rises the total speed falls again. So keep in mind that you might have to experiment a bit to find the optimal size for your table/s within an application

So this all begs the question: can we squeeze out any further speed? The answer is "Yes"!

Batch mode

NexusDB has a Batch insert mode that wraps a number of Posts into one single operation by combining a number of bytes to write into one so called BatchPost command.

To use this we just need to make some slight changes to our original loop:

```

nxTable1.EmptyTable;
nxTable1.Open;
StartTime;
nxTable1.BeginBatchAppend(
trunc(aSize*1024*1024));
for I := 0 to 100000 do
begin    // Iterate
    nxTable1.Append;
    nxTable1.FindField('FileName').AsString:='TestFileName';
    nxTable1.FindField('Directory').AsString:='TestDirectory';
    nxTable1.FindField('FileType').AsString:='TestType';
    nxTable1.FindField('FSize').AsInteger:=1234567890;
    nxTable1.FindField('FDat e').AsDateTime:=now;
    nxTable1.FindField('Attributes').AsInteger:=654321;

    nxTable1.FindField('ParentID').AsInteger:=-1;
    nxTable1.Post;
    if (i mod 10000)=0
    then
        SetCounter(i);

```

```

        end;      // for
        SetCounter(i);
        nxTable1.EndBatchAppend;
        PrintTime('Batch
        ('+floattostr(aSize)
        +'M)');
        nxTable1.Close;
    
```

Again note the aSize variable. This time it is used to define the size of the batch operating in Megabytes (note that NexusDB expects its value in bytes, that's why we multiply it with $1024*1024$). Each post is just caching the data and once we reach the given ASize, it posts the changes and writes/flushes them to disk. This reduces write access and flushing to the much slower file system. It is essential that the server has enough memory to handle these blocks of memory. Because if the server needs to swap memory to disk the effect turns around and makes your operation slower instead of faster!

Running our example again with different batch size gives us these results:

Bytes per Batch	Time
512 kByte	11.45 seconds
1 Mbyte	11.25 seconds
10 Mbyte	9.2 seconds
15 Mbyte	8.7 seconds
20 throughput	9.3 seconds

This time we can even reach 11494 records/second which is an average of 0.087 milliseconds per record.

Again note that bigger is not necessarily better. With increasing batch size the internal management complexity rises and thus the timings get worse after an optimal batch size.

Conclusion

Our original loop took 61 seconds. By tweaking the code we were able to reduce the time to 8.7 seconds, a speedup of 7 times faster. Lots of real-world bulk operations involve operations to millions of records and cutting the time taken to a 1/7th, or less, for these operations is very appealing indeed.

Looking at the final table size (including indexes) of 201.105.408 bytes we are talking about 22.6 Mbyte/seconds raw disk write speed. Using Passmark's PerformanceTest, the same partition can perform to 33.2 MByte/second in the Sequential Write test. In other words: NexusDB is capable of performing to about 2/3 of the raw disk speed using batch mode. We think that's a very good speed :)

Please note that the speed of write operations rises and falls with the speed of your I/O subsystem. Running the same tests on a machine with triple striped 15k rpm harddisks resulted in a result of 4.3 seconds for the 15 MByte batch (about 45 Mbyte/sec raw speed). Passmark's PerformanceTest returned 68 MByte/sec for this setup.

Considerations about remote servers

The above tests were all run with an embedded server. But how does the above apply to remote servers?

On networks the main performance killer is network speed and there especially the latency. Let's take a look at this briefly now.

The most common network speed nowadays is 100MBit. On the machine used for the tests a "ping NexusDBServer -l 1276" (a ping to the server machine using 1276 bytes - equal to the record size - packages) takes an average of 8 ms. Now let's look a bit closer at what the VCL does with Append/Post. It first positions at the end, thus we need to send a message to the server and wait for a reply. Then we post the data to the server and again wait for an answer. So in total we have to deal with network latency 4 times per loop instance. This results all together in a total optimal latency of $4 * 100.000 * 8 \text{ ms} = 3.200.000 \text{ ms}$ or 3200 seconds. In plain words that means that our original base loop performance will be completely unacceptable even if it is running under optimally achievable network speeds.

Well, that's under optimal circumstances. In real life that loop took about 15700 seconds or 260 minutes. Not a very good state of affairs.

Let's wrap this into transactions and see what happens.

Loops per transaction	Time
1000	603 seconds
10000	604 seconds
20000	602 seconds
100000	634 seconds

Still not really fast. But then, transactions only slightly reduce the message count and do nothing about the transferred data size.

So now let's look at batch operations. Using batch mode means that we massively reduce the number of messages. For example using a 1 MByte batch size reduces the number of messages sent to the server from 200.000 to about 130 (total of approx 128 MByte of data transferred), for 10 MByte roughly 15 messages (please note the VCL does not position the cursor at the end before each Post when using batchmode!).

Bytes per Batch	Time
512 kByte	73 seconds
1 MByte	65 seconds
5 Mbyte	65 seconds
10 Mbyte	64 seconds

Looks like the limiting factor here is not latency anymore, but the need to transfer about 128 MByte of data over the network. NexusDB has built in message compression, so let's take a look at this. We add nxlzZipCompressor to our uses clause and set the CompressionType property of the transport to 9

(highest zip compression). In this (optimal, because same data) case the amount of data is reduced from 128 MByte to 400 KByte.

Bytes per Batch	Time
512 kByte	71 seconds
1 MByte	40 seconds
5 Mbyte	31 seconds
10 Mbyte	41 seconds

Ok. That's much better isn't it? We've cut the time from 15700 seconds to 31 seconds, which is more than 500 times faster.

The limiting factor is now the CPU for compression and decompression of data. Mind you we are looking at a 100MBit network here. The slower the network the more important it becomes to reduce the message count and transfer volume.

As an example we look at a real world 802.11b network. Same loop, same server machine as above but this time we use the wireless network for the test. I really didn't want to run the base and transaction loops as this would have taken far too long.

Here are the results without compression. Again the limiting factor is not just latency but also the throughput of the network. Transferring 128 MByte simply takes some time.

Bytes per Batch	Time
512 kByte	280 seconds
1 MByte	248 seconds
5 Mbyte	243 seconds
10 Mbyte	256 seconds
20 Mbyte	261 seconds

And here are the (excellent) results with maximum zip compression.

Bytes per Batch	Time
512 kByte	72 seconds
1 MByte	56 seconds
5 Mbyte	41 seconds
10 Mbyte	40 seconds
20 Mbyte	48 seconds

We've now seen a lot of numbers and have been able to get some impressive improvements.

You can imagine that what has been said applies in a similar way to read operations. Keep these things in mind when designing an application that needs to do bulk inserts or lots of operations over a

network (especially slow networks like the internet) and you should be able to get a superbly performing solution running and your clients happy.

Don't just use the standard out of the box methods, but use NexusDB's flexibility and built in features to fine-tune and improve the performance of your applications! Develop like the NexusDB professionals do!



13.10 How to restructure Tables?

How to restructure Tables?



Restructuring Tables is simply a matter of utilizing TnxDatabase's RestructureTable method. The RestructureTable method declaration is as follows:

```
function
TnxDatabase.RestructureTable
(const aTableName : string;

          aDictionary : 
TnxDataDictionary;

          aFieldMap    : 
TStrings;

          out aTaskInfo   : 
TnxAbstractTaskInfo)

          : 
TnxResult;
```

The aDictionary parameter represents the new structure for the table. The aFieldMap provides the field name mapping in the form of: NewFieldName=OldFieldName. The aTaskInfo is an Object that can be used to query the completion status of the restructuring or cancel the restructuring. It is most commonly used to provide User feedback during this process.

IMPORTANT: The aTaskInfo instance that is returned must be freed when completed.
NOTE: Restructuring requires exclusive access of the table.



13.11 How do handle Autoinc manually?

How do handle Autoinc manually?



NexusDB tracks a "autoinc" value in the file header. This value starts out as 0 on a newly created table. GetAutoIncValue directly returns this value. SetAutoIncValue directly sets this value.

When a record is inserted the server checks if it contains an autoinc field and if the field is still NULL. If yes it increments the autoinc value in the table header and assigned the incremented value to the autoinc field.

That means:

- If you assign (client side) a value to the autoinc field the server side processing will not be done and the value in the file header will not be incremented. It is your responsibility to update that value using SetAutoIncValue.
- You have to make sure that between your call to GetAutoIncValue and later SetAutoIncValue no one else modifies the autoinc value in the header which can happen as a result of either calling SetAutoIncValue or inserting a record into the table that contains NULL in the autoinc field.

To achieve this the code should look like this:

```

var
    AutoIncValue: TnxWord32;

with nxTable, Database do
begin
    StartTransactionWith([nxTable]);
    try
        Append;
        try
            GetAutoIncValue(AutoIncValue);
            Inc(AutoIncValue);
            AutoIncField.AsInteger
            := Integer(AutoIncValue);
            SetAutoIncValue(AutoIncValue);
            <SetOtherFieldValues>

            Post;
        except
            Cancel;
            raise;
        end;
        Commit;
    except
        Rollback;
        raise;
    end;
end;

```



13.12 How to handle locking errors?

How to handle locking errors?



Pessimistic locking:

```
Table.Edit; // locks the
record, no other client will
be able to make
conflicting changes
try
    // set field values
    Table.Post;
except
    Table.Cancel;
    raise;
end;
```

Optimistic locking:

```
repeat
    Table.Edit; // just gets
the current record version
from the server
without placing a lock
try
    // set field values
    Table.Post;
    break; // exit the
repeat loop, record posted
successfully
except
    on E:
EnxDatabaseException do
begin
    Table.Cancel;
    case E.ErrorCode of
        DBIERR_OPTRECKFAI
LED:
        { conflicting
change... retry };
        DBIERR_OPTRECKREC
DEL: begin
        { record has been
deleted... }
        raise;
```

```
        end;
    else
        raise;
    end;
end;
else
    Table.Cancel;
    raise;
end;
until False;
```



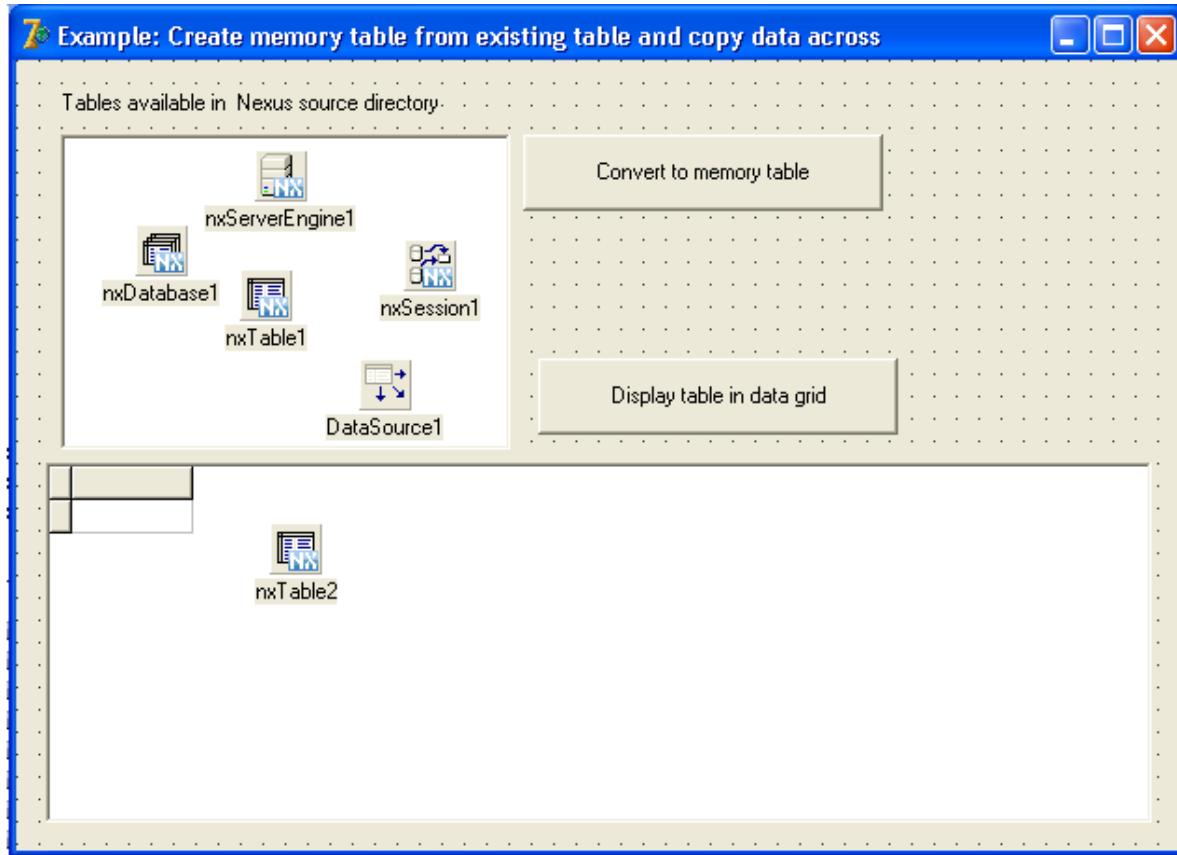
13.13 How to use Memory Tables?

How to use Memory Tables?



Memory tables in NexusDB are, to be blunt, nothing more than normal tables that are RAM resident. There are no limitations with a Nexus memory table over a disk-based Nexus table. Consequently, you can treat your memory tables with exactly the same code and in exactly the same way as you would your normal tables. This example shows you how to easily and quickly create and populate memory tables.

Create a new project with a single form as shown in the screen snapshot below. Connect up the NexusDB components as per Section 6 above. Be aware that we have hardcoded the "uses nx1xAllEngines" registration rather than use the Tnx1xAllEngines component.



Connect up the following code to the appropriate button click events:

Ancillary Code:

```

procedure
  TMainformMemTablesDialog.nxLoadTables;
var
  Tables:TStringList;
begin
  Tables:=TStringList.Create
  ;
  try
    nxDatabase1.GetTableName
    s(Tables);
    ListBox1.Items.Assign(Ta
    bles);
    finally
      Tables.Free;
    end;
  end;

procedure
  TMainformMemTablesDialog.For
  mCreate(Sender: TObject);
begin
  nxDataB
  ase1.AliasPath:=Ext
  ractFilePath(ParamStr(0));

```

```
    nxDataBase1.Connected:=True;
  end;
  nxLoadTables;
end;
Top button - converts a
normal table into a memory
table:
procedure
TMainformMemTablesDialog.Button1Click(Sender:
TObject);
begin
  If Not nxTable1.Active
then ShowMessage('No table
is open') else
  if nxTable1.TableName[1]
='<' then ShowMessage('This
is already a memory table')
else
  begin
    with nxTable2 do
    begin
      Close;
      TableName:=nxTable1.Ta
bleName;
      Open;
      Close;
      TableName:='<' +TableNa
me + '>';
      CreateTable;
      Open;
      CopyRecords(nxTable1,T
rue);
    end;
    nxLoadTables;
  end;
end;

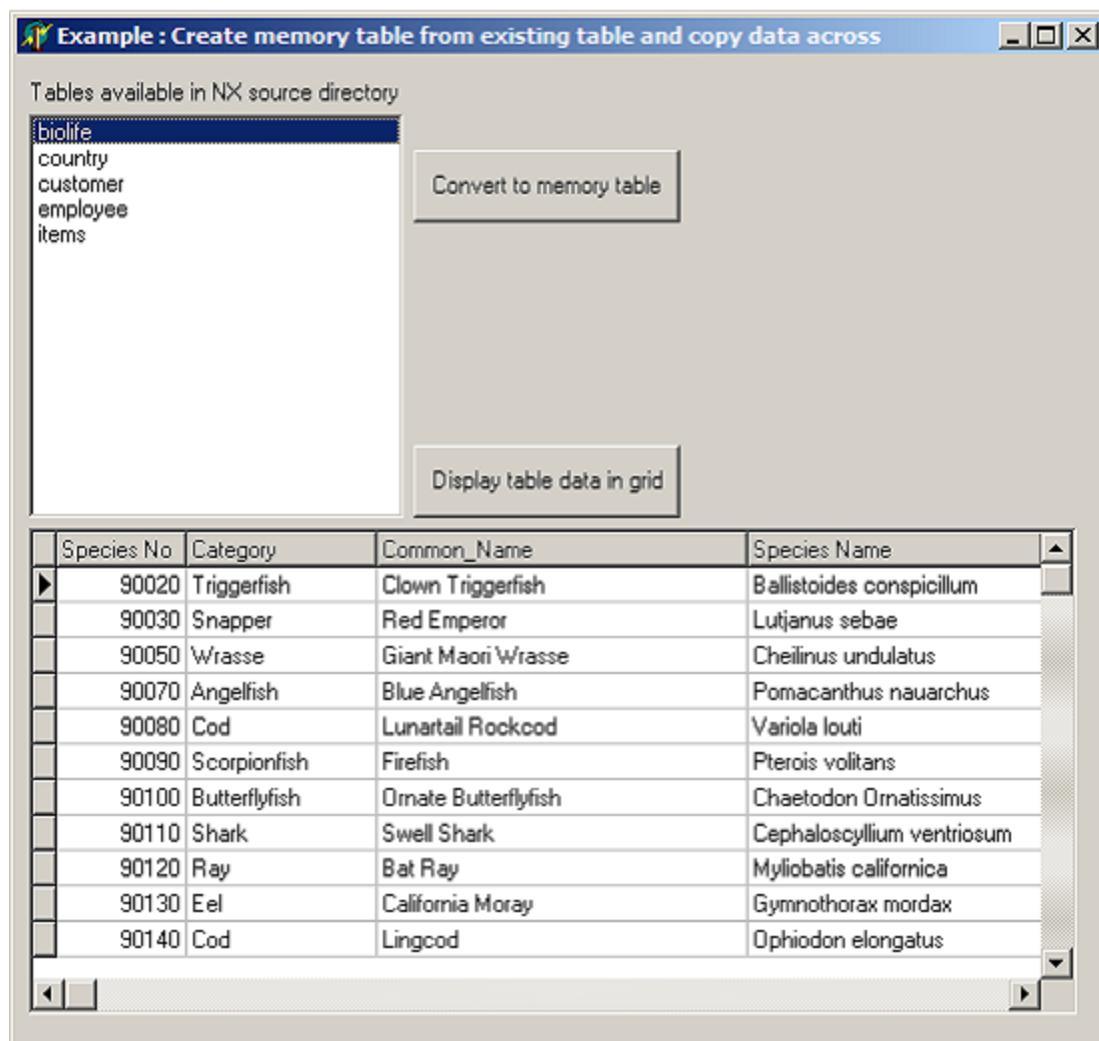
```

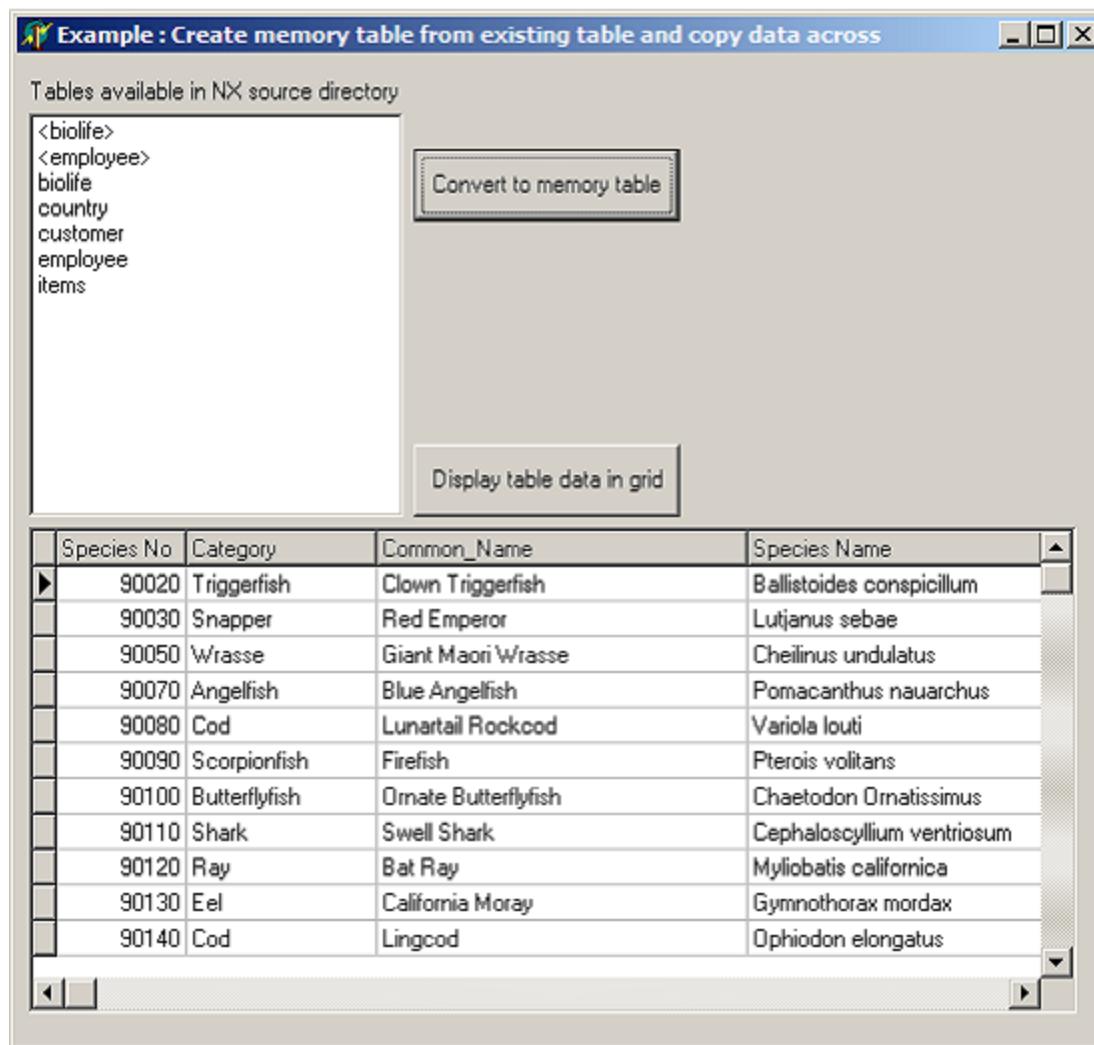
Bottom button – displays data from selected table in the TdbGrid:

```
procedure
TMainformMemTablesDialog.Button2Click(Sender:
TObject);

begin
  If ListBox1.ItemIndex<0
then ShowMessage('No Table
Selected') else
  begin
    nxTable1.Close;
    nxTable1.TableName:=List
Box1.Items[ListBox1.ItemInde
x];
    nxTable1.Open;
```

```
        end;  
    end;
```





This example has shown you, in a very quick way, how to create and populate a memory table from an existing Nexus table. This is the simplest way to do it : associate a TnxTable component with an existing Nexus table, rename the table with angle braces (< and >), and then create the table. The CopyRecords method of the table then copies all the records across in one operation.

For those who use excellent kbmMemTable want an in-memory structure with database like facilities are to note that memory table in Nexus is basically the same thing but with the added advantage that a NexusDB in-memory can be used by multiple threads in the same way as a normal nexusdb table and that can use all the capabilities of the SQL engine. There also is the added advantage that, as required, NexusDB can swap out blocks of the tables into temporary storage if you happen to put more information in then fit within the MaxRam setting, kbmMT has to depend on the normal windows page file which is not always an optimal solution.

Is Memory Table simply a way of caching a normal Nexus table?

More the other way around, a "normal" table is technically the same as an in-memory table with the following additional functionality: blocks are read as needed from disk, when a transaction commits the modified blocks are written to disk, if MaxRam is reached blocks can be simply discarded (and

later read in again) instead of having them written out to temporary storage. These are the only real differences between a in-memory table and a persisted table.

Does NexusDB aggressively cache?

So long as your data fits into the available memory...

That's the huge advantage a proper pure C/S implementation (server opens files exclusively, locks are handled with custom structures inside the server instead of OS locks on shared files) has over file-sharing databases, both direct file-sharing or file-sharing with remoting layer on top which some people then call C/S.

The server can always be sure it's cached data is consistent. All changes to the files go through the same layer (BufferManager in NexusDB) which is responsible for caching. As long as the server has more memory available as the size of the table files every block will only be read once and stays in the buffer manager from then on. The BufferManager is far below the point in the design where different users/sessions are distinguished, in a multi-user environment all sessions share the same buffer manager and a block needs to be read only the first time a user accesses it. In a file-sharing design if one session writes to a table all other sessions need to read that changed block in again (actually, they have to read the blocks in all the time because it's unknown if the block in question has changed or not).

Difference between a NexusDB in-memory table and a normal table

The only difference between a NexusDB in-memory table and a normal table is that in-memory tables don't have a file assigned and the buffer manager never writes changes for them out to disk. For all intents and purposes, once a normal table is completely present in the buffer manager, as long as only read access takes place, it will perform exactly like an in-memory table would perform.

How is a table hosted by an embedded server very similar to an in-memory server scenario?

The engine never accesses files directly in any way. Instead, they use a function that looks like this:

```

type
  TnxBaseTransaction =
class;
  TnxBaseFile = class;
  TnxBlockNumber = Cardinal;

  TnxReleaseMethod =
procedure(var aBlock :
PnxBlock) of object;
  TnxBlock = array
[TnxWord16] of TnxByte8;
  PnxBlock = ^TnxBlock;

function GetBlock(aTrans
:
TnxBaseTransaction;
  aFile
  : TnxBaseFile;
  aBlockNum
ber : TnxBlockNumber;
```

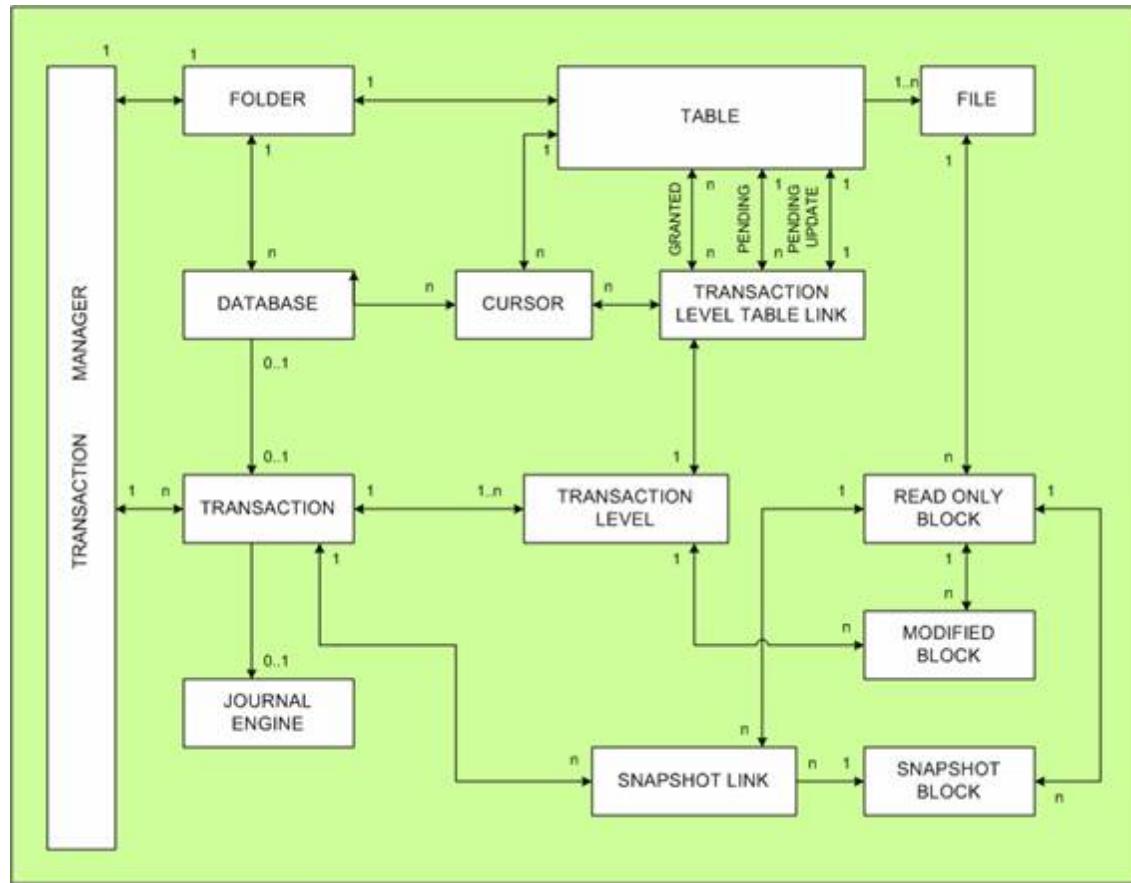
```
        aMarkDirt
y      : Boolean;
var
aReleaseMethod :
TnxReleaseMethod)

: PnxBlock;
```

and the way how that function is used looks roughly like this:

```
var
  Block      :
PnxBlock;
  ReleaseMethod :
TnxReleaseMethod;
begin
  Block := GetBlock(aTrans,
aFile, 1234, True,
ReleaseMethod);
  try
    { us Block to read/write
that specific block }
  finally
    ReleaseMethod(Block);
  end;
end;
```

now take a quick look here:



Pay special attention to "read only block", "modified block" and "snapshot block".

if GetBlock is called with:

- aTrans = nil then the returned block is the "read only block" (newest committed version of the block)
- aTrans being a snapshot transaction then the returned block is either the "read only block" or a "snapshot block" if the block has been changed by committing another transaction which modified that block since the snapshot transaction was started.
- aTrans is a normal transaction then the returned block will be the newest modified block (or the read only block if the block hasn't been modified in this transaction yet), if aMarkDirty is true and there either is no modified block yet or the newest modified block has a lower level then the current transaction then a new modified block is created, the contents of the prior block (newest modified or read only) is copied into it and the new modified block is returned.
- When a transaction is rolled back all modified blocks for the current transaction level are simply discarded.
- When a nested transaction is committed all the modified blocks for the current transaction level are added to the next lower trans level, replacing exiting modified blocks if required.
- When a non-nested transaction is committed all the read only blocks are replaced by the modified blocks, if there are any snapshot transactions active the readonly blocks are moved into snapshot blocks as required instead of being discarded. If the "file" is actually backed by a physical file on disk all the modified blocks are written to disk now, if requested (failsafe transactions) using a 2 phase commit system.

- When a snapshot transaction ends all snapshot blocks that are no longer required are discarded. The buffer manager keeps track of all these blocks, the time they were last accessed, if they are currently "in use" (meaning GetBlock has been called but the ReleaseMethod hasn't yet) and how much memory they require. Whenever the buffer manager needs to allocate an additional buffer (w.g. because a block that isn't yet present in memory has been requested, a new modified block needs to be created, a new snapshot block needs to be created...) it checks if all current blocks together exceed the MaxRAM setting. If yes, it starts to check the oldest blocks if their memory can be freed up for other use till enough memory has been freed to stay below MaxRAM.
- the block can't currently be "in use"
- read only blocks can just be discarded because they can be read from disk again. *If* the "file" for this block isn't backed by a real physical file (= in-mem table) it has to be written to temporary storage instead.
- all other blocks can be written into temporary storage.

When a block is requested the buffer manager reads the block from disk or temporary storage as required if it isn't already present in memory.

The *only* difference between a persisted table and an in-mem table is that an in-mem table is not backed by a physical file on disk, that means:

When committing a non-nested transaction:

- persisted tables have to write the modified blocks into their physical file before returning
- in-mem tables just do nothing. When MaxRAM is reached and the oldest blocks are used to free up memory:
- persisted tables can just discard the read only blocks as they are identical to the version on disk
- in-mem tables have to write the block into temporary storage

Given your description of what you want to do I don't see much point at all in using in-mem tables. As long as your MaxRAM setting is larger than all tables together sooner or later they will be cached in memory completely...if they are larger than MaxRAM the most often used blocks will be cached in memory, with blocks being read from disk as required and the oldest blocks just discarded. With in-mem tables everything that doesn't fit into memory would need to be written into temporary storage. You can force the data into memory by just opening a TrxTable and doing a "while not Eof do Next"; After that all the data blocks and the blocks for the current index should be in memory (again, assuming MaxRAM is larger than the table).



14 Frequently asked Questions

Frequently asked Questions



Q: I get an error with just an error number. What does the error number mean? What should I do?

Most likely you hit an OS problem. For a complete list of public error messages in windows take a look at this MSDN page.

Q: How to I query a date in SQL? What's the date format?

Nexus (and FF) SQL want the date in yyyy-mm-dd format with a (optional) Date keyword.

Example:

```
Select * from mytable where  
mydate = Date '2002-11-24'
```

Q: No file header access factories have been registered" problem in BCB. What now?

In BCB, developing an Embedded Server App, I got a "NexusDb: No file header access factories have been registered. You may need to include nxseAllEngines in your USES Clause.[\\$3C36/15414]."

Add to the project source, or one of the cpp files of your project, the following line:

```
#pragma link  
"nxseAllEngines"
```

Alternatively, you can add this OBJ directly to the Project, using the Project Manager.

Q: Do you have some more Info on SQL 92, 99 and 2003?

SQL92:

<http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt> (last full standard before approved; careful: ~1.7megs)
<http://www.contrib.andrew.cmu.edu/~shadow/sql/sql2bnf.aug92.txt> (bnf)
<http://developer.mimer.se/validator/parser92/index.tml> (conformity checker)
<http://csf.colorado.edu/local/sql/>

SQL99:

<http://www.ncb.ernet.in/education/modules/dbms/sql99index.html> (last full standard before approved)
<http://developer.mimer.se/validator/parser99/index.tml> (conformity checker)

SQL2003:

http://www.wiscorp.com/sql/sql_2003_standard.zip (last full standard before approved; careful: ~10megs!)
<http://developer.mimer.se/validator/parser200x/index.tml> (conformity checker)

General:

<http://www.wiscorp.com/SQLStandard.html>

<http://members.tripod.com/er4ebus/sql/ch01.htm> (Teach Yourself SQL in 21 Days)

Q: I'm getting a strange error (AV, or it is simply not working) when using a function declared virtual in the Delphi source. What to do?

A: In BCB5 and BCB6 there are some compiler bugs that end up screwing up the class virtual table, but, fortunately, the solution is pretty simple. In the classes that you have discovered this problem, move the public part before the private/protected part of the class. Be aware that the properties, in most cases, should stay after the protected and private parts (as they usually depend on private functions). Another solution is to change the "virtual" keyword to "dynamic" for those functions. Doing this will impact the performance of the functions, but, hopefully, in most cases this won't be noticed, and less if you are working with a unmodified compiled server (in a client/server setup, of course). For both ways, the packages must be recompiled with the changes made.

Q: When using a function that has as parameter a PnxByteArray I'm getting a linker error.

When compiling the Delphi unit, the compiler doesn't do the same as the header creation engine when dealing with the unit. The compiler creates the binary taking a PnxByteArray as an unsigned char[2147483647] *, as the following dump from the compiled OBJ shows:

```
Name: 97: '__fastcall
Nxsqproxies::VariantToNative
e(Nxsdtypes::TnxFieldType,
const System::Variant&,
unsigned char[2147483647] *,
int)'
```

The problem here is that a PnxByteArray is not an unsigned unsigned char[2147483647] *, as the translator correctly does, in nxltypes.hpp:

```
typedef Byte *PnxByteArray;
```

but only a unsigned char *. So, the solution here is to create an alias for the correct function. The way to create this is, as found in the borland.public.cppbuilder.vcl.components.wrting:

<==quote==>

a) tdump both sides or the unresolved external problem:

```
tdump -m myhintwindow.obj
myhintwindow.dmp
tdump -m C:
\Windows\System32\VCL60.bpl
VCL60.dmp
```

b) find the name-mangled names:

```
grep -i hintwindow.*ncpaint
*.dmp > manglednames.txt
```

c) paste the names correction into an alias in the .cpp source:

```
//place in
myhintwindow.cpp
#pragma alias
"@Controls@THintWindow@NCPaint
nt$qqrpv" =
"@Controls@THintWindow@NCPaint
nt$qqrui"
```

====quote====>

In this case, instead of vcl60.bpl, you will dump the obj from the nexus delphi unit which defines the problematic function. If you use this function in several cpp files, then you can add the #pragma alias in the header file for the delphi unit. Thanks to Craig Farrell, from the borland.public.cppbuilder.vcl.components.writing newsgroup, for pointing out the solution.

Q: What is fastest? FieldByName, FindField, or something else?

FieldByName is fine, but if your accessing the field a lot try and keep a pointer to the returned field, this saves a little bit of time.
eg. Instead of the this->

```
procedure TForm1.Test;
var
  tot:integer;
begin
  tot:=0;
  table1.first;
  while not table1.eof do
begin
  tot:=tot+table1.fieldbyname('Value').AsInteger;
  table1.next;
end;
end;
```

Do this instead->

```
procedure TForm1.Test;
var
  tot:integer;
  ValueField:TField;
begin
  tot:=0;
  ValueField:=table1.fieldbyname('Value');
```

```

        table1.first;
        while not table1.eof do
begin
        tot:=tot+ValueField.AsIn
teger;
        table1.next;
end;
end;

```

As you can see from above the FieldByName is only called once and will save Delphi from having to find the field each time.

Another option is to create persistent fields in Delphi by double clicking on the Table and adding them. This then allows you to access the fields like

```

Table1SomeField.Value:=SomeV
alue;

```

What should one use instead?

Persistent fields should be faster than FieldByName. one can also assign the result of FieldByName to a local variable.

If one uses FieldByName a lot since that's how he knows his fields?

Nothing wrong with using FieldByName unless one need to squeeze the last ounce of speed out.

Q: What is the best way to perform Auditing of changes/updates via an Extender? Create and destroy the nx... objects within the event each time or implement a thread-safe cache of nx objects?

a) don't use the simple monitor for this. instead:

- implement your own monitor/extender classes
- don't implement anything in the monitor, instead keep your complete implementation (and all cursors you need to perform your updates/logging) in the extender. There will be one instance of the extender per cursor you are monitoring. as the cursor will never be used in a multi threaded way your extender doesn't need to be threadsafe.

b) never use any of the nxdb components in a monitor/extender. Instead: directly use the core server API defined in nxsdServerEngine.

c) make sure the cursor you are using to log your changes belongs to the same database/session as the cursor you are extending, otherwise transaction processing will not work as you expect.

for b) and c) if your extender is extending a TnxServerCursor, you can us the Database property of that to get hold of the TnxServerDatabase object owning that cursor. The database object has a CursorOpen function that allows you to open other cursors owned by the same database and living in the same transaction context as the cursor you are extending. You can create that cursor when your extender is created and store it in the extender object. You won't need to do anything about transactions because there will always be an implicit or explicit transaction already active when your

extender is notified about any of the eaRecord* events. For a complete example of how to implement monitors/extenders please take a look at the RefIntegrity classes in the bonus directory.

Q: When i change an index segment of a bookmarked record, GotoBookmark returns the error "Record/Key deleted [\$2204/8708]". I suppose the reason for this behaviour is because the index information is used to retrieve the right record. Shouldn't the bookmarked record be found, even if i change it?

Our bookmark implementation saves the index info in the bookmark, so the answer would be 'no'. Use a different index from the one you are changing data in.

Q: Would it be painful to add the record's "refnr" value to the bookmark structure? Then, if the index info causes a problem, the value could be re-located using the refnr. (I think I'm right that the refnr of a record is never changed.) It would be unusual, I'd think, to have more than a few dozen live bookmarks in an app (wouldn't it?).

The RefNr is already in there. (for non-unique indices the RefNr itself becomes part of the key as the B*Tree algo itself only operates on unique keys)

The problem is that RefNr's are reused. There is no way to distinguish between a record that had been deleted and the RefNr reused for a completely different record and a record that has been modified which resulted in changed keys. A bookmark is supposed to bring you back to the record it came from. Using RefNr alone is not guaranteed to do that. In the end it comes down to the definition of "record identity" and that is currently defined as key + refnr.

Q: I know that "many" indexes per table caused a performance drawback (slow append and so on) in the "old" days, but how is it in Nexus? Is there any "max-count" of indexes per table that I should try not to get above?

First some concepts:

- record - a sequence of bytes, logically separated into different fields
- record engine - a piece of code that's responsible for storing records, handing out refnr's, later being able to return that record given the refnr and able to "delete" (mark for reuse) a record given the refnr
- refnr - a 64bit value representing a specific record that was stored using a record engine
- key - a sequence of bytes composite key - a key which is composed of subkeys each derived (usually from one field of the) record.
- key engine - a piece of code that is able to a) create a key given a record b) compare 2 keys, the following conditions must always be true: $(A > C) = (A > B)$ and $(B > C)$; $(A < C) = (A < B)$ and $(B < C)$; $(A = C) = (A = B)$ and $(B = C)$;
- index - an ordered list of keys + refnr, the order of the keys is determined by the key comparison function
- index engine - a piece of code that is able to maintain an index, the following functionality is required: a) given a key and a refnr, insert that information into the index b) given key + refnr delete that key from the index c) given a key find and return the key+refnr that fulfills a specific condition relative to the given key (smaller, smaller-or-equal, first equal, last equal, larger-or-equal, larger) d) given a key+refnr return the next/prior key+refnr in sort order.
- b*tree - an algorithm designed to implement an index using page based approach.
- key path - for a b*tree this is an array with 1 entry per level of the tree containing page number and offset in page

B*Tree Index(which is what the default index implementation of NexusDB does):

There are 2 kinds of pages, internal and leaf nodes. actual keys are only stored in the leaf nodes, the internal nodes store values derived from that actual keys to lead the search for a specific key starting from the root node. The depth of the tree (number of pages from the root node to the leaf node containing the key) is always constant for all keys and increases as the tree grows. each page contains a sorted list of keys (or key derived values for internal nodes). Searching for specific key has to start at the root page, doing a standard binary search on that page (requiring the square root of the number of keys in the page of key comparisons), follow the reference to the next page on level deeper into the tree, searching again.. and so on till the key is found (or not found) in a leaf node.

The maximum number of keys per page depends on the size of the key (plus some constant overhead) and the size of a page (minus some constant overhead). The overheads are different for internal and leaf nodes. But for larger (>16 bytes) key sizes this isn't very relevant.

Each page is filled between 50-100%, the avg. fill factor in a regularly modified b*tree is 66.6%. This means the AvgNumberOfKeysPerPage is $\text{MaxKeysPerPage} * 0.666$.

The avg. number of leaf nodes is $\text{Ceiling}(\text{TotalNumberOfKeys} / \text{AvgNumberOfKeysPerPage})$

The avg. depth of the tree is $\text{Ceiling}(x^{\text{th}} \text{ root of LeafNodeCount})$ where $x = \text{AvgNumberOfKeysPerPage}$.

The total number of key comparisons is about $\text{AvgTreeDepth} * \text{Sqrt}(\text{AvgNumberOfKeysPerPage})$.

Given all this information you can put a diagram together that shows how the number of required key comparisons needed to find a key in the index changes with the number of keys.

Inserting a record requires inserting one key per index (which requires a find operation to find the insertion point).

Deleting a record required deleting one key per index (which requires a find operation to find the key and generate a key path on the way from the root to the leaf).

Updating a record requires one delete and one insert per index which had it's key changed (2 find operations)

In addition to the pure time it takes to compare the keys when walking the tree it's especially important to consider the IO costs:

Inserting/Deleting a record (without any indices):

read/write - 2 pages (table header page + data page)

The header contains the total count of records in the table and needs to be updated.

If this was the last used slot in the data page (for a delete) or if there were no data pages with empty slots available (for an insert) between 1 and 2 additional pages need to be read/written to either add the page back into the main list of reusable pages or get a new page from there or add an additional page to the table

Updating a record (without indices) read/write - 1 page

Inserting/Deleting a key into an index:

read - 1 page per level of the tree + indices header page (containing the root node page number)
write - 1 leaf node page + indices header page (containing the total number of keys for the index)

If there are multiple indices the "indices header page" will be read/written only once as all indices use the same page, all modifications take place in an implicit transaction and the buffer manager will cache the page.

In addition, if the leaf node that the find operation determines as the insertion point/deletion point is either full (insert) or would be less than 50% after the operation (deletion) it's required to either balance the page with its siblings (adding 1 or 2 more page reads and 1 page write) or, if the siblings are also full/half-empty execute a page split (adding a new page, splitting the keys into 2 pages) or page merge (removing one page and merging the keys with the sibling). which costs a few more page reads/writes. In case of a split or merge the parent page needs to be updated as well and the operation will cascade upward through the tree as long as the parent page is full/half-empty and can not be balanced with siblings. If it reaches the root node of the tree a new root node is created (and the depth of the tree increases by one level, the new root node will have 2 keys) or the current root node is removed (and the depth of the tree decreases by 1 level, this happens when the key count of the root node reaches 1).

Last thing we have to look at is page size. For each page read/write there is a (more or less static) time (drive seek time) + a variable time depending on the page size. Larger pages result in a higher fan-out factor (avg. keys per internal node) of the tree, reducing the depth of the tree, reducing the number of pages read/written. At the same time, larger pages take longer to read/write and contain more information that is possibly not relevant. The "negative" impact of larger pages is the highest if you do single record operations (insert/update/delete) and gets smaller if you use larger transactions (because multiple operations that write to the same page will only result in a single read / write).

In summary... the answer to your questions is "that depends".

Q: How does Nexus work with DB Grids?

When you use a non-data-aware form to collect data, you can start a transaction and then do your commit immediately after the data was UPDATED/INSERTED. But when you use a DBGrid there is no StartTransaction or Commit. NexusDB does nothing special with DB grids.

Q: How do I handle Exceptions during transactions?

Here's how an exception handler might look:

```

        except
            if
                tnxDataBase1.InTransaction
            then
                tnxDataBase1.Rollback
            ;
            end;
    
```

Don't forget to call Cancel on any tables that are in Edit mode when rolling back!

Code actually should always look like this:

```
nxTable.Insert; //or
.Edit;
try
//....
nxTable.Post;
except
nxTable.Cancel;
raise;
end;
```

Q: How to load from/to a Blob field?

The code to save a RichEdit content into and out of a blob field

To Load

```
var bs : tStream;
begin
MyTable.Open;
If not
MyTable.FieldByName('text').
IsNull then
RichEdit.Lines.Clear;
bs := myTable.CreateBlobStream(MyTable.FieldByName('text'),bmRead);
try
RichEdit.Lines.LoadFromStream(bs);
finally
bs.Free;
end;
MyTable.Close;
```

To Save

```
var bs : tStream;
begin
MyTable.Open;
MyTable.Edit; // or
append, u know...
bs := MyTable.CreateBlobStream
(MyTable.FieldByName('text')
,bmWrite);
try
RichEdit.Lines.SaveToString
(bs);
finally
```

```

        bs.Free;
end;
MyTable.Post;
```

Q: Is there an error with StartTransactionWith, as it shoud log the Tables immediately or is the behavor of StartTransactionWith correct?

StartTransactionWith only starts the transaction if it can place exclusive transaction locks on all passed in tables. If it can't place exclusive locks on all tables within the timeout period no transaction will be started.

While StartTransactionWith tries to acquire exclusive locks on all tables it will NOT keep tables where it can acquire an exclusive lock locked. Instead the function waits till ALL tables are available for an exclusive lock at the same time within the timeout period. If StartTransactionWith is successful all passed in tables will be exclusively locked by the time the call returns.

Please pay attention to the fact that StartTransactionWith is a function returning an TnxResult, only if that is = DBIERR_NONE has a transaction been started. No exception will be raised on error.

Q: If a filter is applied to a 10,000,000 row table and returns 10 rows, it won't be instantaneous because NexusDb will process all 10,000,000 rows looking for the 10 that satifies the filter. This is sloooowwww. Why?

That is, by definition, the fundamental difference between a table and a query. Query's pre-process and return result sets. Table's are direct cursors into the data for navigational data access.

Q: How do I access server side objects?

You can use the server engine's IterateDependents method for that.

```

with
bsServerEngine.IterateDepend
ents do try
  for i := 0 to
Pred(Count) do
  if TObject(Items[i])
is TnxStateComponent then
  if
TnxStateComponent(Items[i]).
Enabled then
  if not
TnxStateComponent(Items[i]).
Active then
    // do your stuff

  finally
  Free;
end;
```

Q: I get an edit Timeout when a record is locked. Is there any way to control the time-out instead? Is the TxnTable.Timeout property involved in this at all?

Yes. That's the value you want to reduce. It's the maximum time in msec the server will wait for the lock to become available.

Although I must say that to me it seems strange to use the same time-out for how long you want to wait for a record to be unlocked as for how long you wait for the server to respond at all to various client operations.

Just for the record:

- Setting a different timeout on a TxnTable only affects that single cursor. All other cursors (TxnTable/TxnQuery), database or sessions are not affected.
- You can change the timeout value at any time. It's not something that you have to decide about before opening the table.
- really the one and only thing that is affected by this timeout value is waiting for locks, be it content locks (record and table level, e.g. Edit / LockTable) or transaction locks (implicitly acquired for all read / write access to a table in the context of a transaction).
- Independent of how many locks a specific call has to acquire, when a call enters the server engine the current time is stored. All wait operations in the context of this single call will apply the timeout relative to the moment when the call initially entered the server engine.

Q: My old app. under BDE used OnFilterRecord to display a history. Under the BDE and Paradox tables with 25,000 records it was instant. Now under Nexus it takes 10 secs..Any ideas on speeding this up?

The explanation, *why* NexusDB is much much slower: The BDE is working on local file. When setting a filter each record is loaded and the OnFilterRecord method is called directly. In a C/S system (like NexusDB eg.) there's a bit more involved: you tell the server that you want to move to the next record. the server does this and answers back (over the network) the *physical next* record, the client is then doing the onFilterRecord calculation. if it matches fine, if not, it needs to request the next record over the network, the server again returns the next physical record, ...

With 25000 records to filter even with a very fast network (and latency of say half a ms) this needs quite some time (with half ms at least $25000 \times 0.5 = 12\,500$ ms is 12.5 seconds) for filtering the whole dataset.

With setrange you just transfer the criteria to the server and the server deals with it using an index. If you don't have a matching index you can still speed the thing up a **lot** by using the normal filter property. In this case the filter evaluation takes place on the server and you shouldn't see too much of a difference between BDE and NexusDB.

Q: Why does applying a range in Enterprise Manager take so long?

EM has recordcount turned on by default. Turn that off. The only way to get an exact recordcount for a range is to count the records. That's what takes so long. Not the range itself.

Q: How can I avoid "Lost communication with network server. [\$2C0C/11276]." while i'm debugging inside the IDE and the application is stopped?"

Set the client side transport's HeartbeatInterval to 0 before activating the transport. Remember to set back the original value when you are finished debugging.

Q: I'm using TCP/IP as transport, but the firewall prevents the ping answer. Is there a connection possibility on the NexusDB without the Ping functionality?"

Yes there is a possibility.

First, the firewall comes into play only if one wants to access the Nexus server from outside the server's subnet. So it takes this as a fact.

Second, it has nothing to do with Ping being blocked by the firewall. One has to configure his/her firewall to allow connections through it on port 16000. This is the TCP/IP port the Nexus server uses by default. In this case clients outside the server's subnet can connect to the Nexus server if they know the server's address. This is usually this external IP address of the firewall. The firewall must be configured to route IP traffic on port 16000 to the machine the Nexus Server is running on. Still no broadcast but connections can be made to the server.

Third: If one wants the clients to find the server behind the firewall automatically, he/she needs to make sure broadcasts make it through the firewall and the Nexus server must be configured to respond to broadcasts. So one has to configure the firewall to pass the Nexus client's broadcasts to the server. Broadcasts are using UDP protocol but it doesn't know the port.

Q: What is the unit init order when i want to use MadExcept?

If you want to use your product with Nexus MM/DB and MadExcept, please use the following order for the unit initialization in MadExcept:

```
nx11FastCpuDetect,  
nx11FastFillChar,  
nx11FastMove,  
nx11MemoryManagerImpl,  
nx11MemoryManager,  
nxReplacementMemoryManager
```

Q: How to avoid "System has been illegally reentered" errors?

If you look at the NexusDB sources, the only place where this error is triggered is if different threads are accessing the same session concurrently. This is not allowed, so in your scenario there must be at least two threads involved. Make sure that *every* thread is using its own VCL components down to transport level. The transport level is the first 100% threadsafe one as all others are derived from the non-threadsafe VCL classes.

Unfortunately in almost all cases this will be hard to debug, especially if you're working with a remote server. If it's a local server engine then open nxsdServerEngine and put breakpoints on all lines that fire

the DBIERR_REENTERED error. Once you stop at one of these lines, take a look at the threadview and the stacktrace of each to figure out what the thread is and where it's coming from ...

Adding Eureka log to Nexus.

Q: We bought Eureka log for our applications and it's a great improvement over what we were using before, so I was looking into adding it to nxserver.exe. Is there an easy way to do this?

It looks like nxExceptionHook logs errors when they're raised while Erekalog logs errors when they aren't handled by the application. Nexus seems to find a way to handle and ignore most errors after they're logged by nxExceptionHook, so none fall through to Erekalog. Is there any point in adding EurekaLog to Nexus? Is there an easy way to go about it?

nxExceptionHook uses the free JCL library to do its work, and yes, it does trap and log exception stack traces just fine. To switch to EurekaLog, undef the {\$IFDEF NX_EXCEPTION_LOG}define in the nxDefine.inc file, and add EurekaLog as described in their docs.

Q: Temp files not stored where i want them to be. This means the application will "pollute" all the directories where data is accessed. Why is that?

If you don't specify a directory the system default temp path is used to store the temporary storage files.

nxTrans.cfg (which you are seeing in every database directory) are **NOT** temporary files. They will never have a size other then 16 byte. These files store the highest LSN (log sequence number) and the date/time of the most recently updated table.

When an alias is first opened the server will check for that cfg file, if it exist compare the date/time stored there with the date/time of the newest table. If it matches the server assumes the LSN in the cfg file is the highest in the alias and uses that to initialize the transaction manager for that alias.

Otherwise all tables in the alias will be opened to determine the highest LSN. The nxTrans.cfg file is also used to exclusively "lock" the alias. While a server engine has the alias opened it holds an exclusive lock on that file preventing all other server engines from trying to access the alias.

Q: How do turn off debugging NexusDB units?

Just put a {\$D-} as first line into nxDefine.inc and all debugging inside all NexusDB units should be gone.

Q: How do I store/load any memory content to/from a Blob field?

Use TnxTable.CreateBlobStream to create a stream for the field. Use TStream.WriteBuffer to write to the stream and TStream.ReadBuffer to read from it. Be sure you free the stream before posting the record.

Q: NexusDB seems to "loose" records. When I post new records and then restart the application they are simply not there. What should I do to prevent this?

Every time you Post a record that has been edited in a grid, the changes are sent to the server and stored. There is no caching or such anywhere. The only things I can think of is:- you have a StartTransaction call *somewhere* in your app (maybe a 3rd party component does it). If you close an app without calling Commit, then the server automatically does a Rollback.

There might be some mixup with the UI and the code behind it, such as batchmode is activated but never finished, or something like that.

Q: What is the speed and resource utilization differences between the table/range and query use?

TnxTable is a direct cursor into the table. "Setting a range" has no other effect than limiting the navigation on the index to a specific subset. No additional resources are needed.

Using a query would require to parse the query string, build a tree of evaluation objects, acquire cursor handles on all source tables, execute the evaluation tree using the cursor handles to read the source data which builds a result set (a result set is conceptionally an unnamed, indexless, in-memory table) and return a cursor handle for this result set.

From an architectural perspective, TnxDataset is a wrapper around a cursor handle. In case of a TnxTable that cursor handle is a direct cursor into a table. In case of a TnxQuery that cursor handle belongs to an unnamed, indexless in-memory table which was created and populated during query evaluation. The SQL engine uses direct cursors into tables internally to evaluate the query expression.

Having Table access as well as Query access is one the great benefits of NexusDB over the pure SQL database engines.

Always try using "the right tool for the job". There are cases where SQL is that right tool but there are also many cases when more direct navigational / cursor-based access is "the right tool". NexusDB has always been designed to give you the best of both worlds in a fast, stable, scalable and dependable package.

Filtered indices on table/queries

Q: Why doesn't NexusDb do some pre-processing on Table Filters.

NexusDB, just like e.g. BDE with Paradox tables, and a lot of other DBMS, implements TnxTable as a direct cursor into the data.

Filters are evaluated on a record by record basis. This is "as designed". Some other DBMS that don't support the concept of navigational table access simulate it by having a "table" component that internally executes queries.

Q: How does NexusDB decide if it has "Lost Communication"?

The server-side watchdog thread will monitor the keepalive messages from the client. If 3 messages go missing, the server declares the client dead, and closes the connection. On the client-side, if the network breaks the connection for any reason, then a "lost communication" exception is immediately raised **when a message is sent to the server**.

Note that last part. It means that you will not get any event that tells you the connection disappeared, you need to catch the exception that will be raised by your regular database method calls (use

Application's global event for instance), and handle it. There is no automatic reconnection, and since it's likely that the server has closed all resources your app were using, you should treat a lost connection exception similar to a fresh start of your application.

Q: Advantage database can have a Range and an indexed filter in place at the same time using different indexes. Why can't NexusDB do this?

Advantage doesn't implement tables at an engine level at all. You are comparing two completely different things here. The Advantage table component internally maps to a query with live result set, which you can also do in NexusDB by using a query instead of a "table".

Essentially Tables and Queries have fundamental different implementations and both Tables and Queries are extremely competitive as implemented in NexusDB.

Q: I have tried to run the NX server as a service under the System account on several NT4 servers, but the service will not start. It works perfectly under other accounts with Admin privileges. What should I do?

Due to security issues with running a server like the NexusDB one we are not allowing the server to be run as Local System user.

Q: I'm trying to install (without success) NexusDB as a Service in windows 2003 SBS. The service can be installed but doesn't start. The user doesn't have the rights for run a service. What now?

Windows 2003 (SBS) is not automatically assigning the "log on as service" right when installed from the API

Probably the easiest way to solve this is:

- logon as admin
- go to the services panel
- open the properties of the NexusDB Server service
- set the "Logon/Run as" to Local System
- close the properties
- reopen it
- set the "Logon/Run as" to the user and password you want
- if the user doesn't have the right "log on as service" the system should assign it automatically.

Alternatively you can assign it on your own in the Domain Controller Group Policy (make sure to not just do this in the local group policy as this gets overridden again by the domain group policy on refresh).

Also make sure that the user has the appropriate rights for accessing the directory, make sure the disk quota is high enough (it's enabled by default in sbs), make sure you've no memory quota enabled, also make sure it has a dependency on the tcp/ip services if you're using tcp and the first access to the server can occur before all services are loaded.

Q: When doing DUnit test, if I store high(longword) in a TnxWord32 field, it shows as -1 in the EM. If I retrieve the record it comes out OK. Is this a bug in EM?

This is a limitation in Borlands VCL. There is not field type for unsigned 32 bit integers. NexusDB has to map nxWord32 fields to TIIntegerField

Which is a signed 32 bit Integer.

You can work with Word32 fields as long as you remember to do the required typecasting when accessing the field.

e.g.

```
var x: LongWord;
x :=
LongWord(Field.AsInteger);
Field.AsInteger :=
Integer(x);
```

So, is there any difference internally in NexusDB between an Int32 and a Word32? Sorting behaviour only?

Internally, **YES**. Sure. SQL, Filters, Indices... everything handles Word32 correctly as Word32. Only the access to the field in the TDataSet layer on the client side doesn't support Word32, which is a Borland limitation in the TDataSet design. Once you have taken that slight bump by correctly typecasting when accessing the field all is fine.

Q: When I run NexusDB server as service, then I can't use the COM protocol in EM. When I run NexusDB server as application, then I can use the COM protocol. It is ok or a bug?"

That's ok. The COM transport uses the ROT (running object table) to register itself and allow client transports to find the server transport. The ROT is a "per Window Station" resource. Services run in different "Window Station"s then the interactively logged on user, thus you can't connect to it from the logged on user.

Q: What is the difference between the customcom transport and the registeredcom transport, and when should you use which?

Registered com transport uses the ROT (running objects table) which is a system local list of running objects managed by windows to establish the initial communication between client and server.

Custom com transport just calls an event and you have to implement that event. passing the InxTransport that's passed into the event to the server side transport, call AttachRemoteTransport there, passing in the InxTransport and then pass the InxTransport you get back from that function back to the client.

How you do this is completely up to you. If both client and server transport are in the same process it's very easy. But you can also use e.g. DCOM or any other mechanism to pass the InxTransports between client and server.

Q: Is NexusDB really more expensive than buying a set of components to access MySQL, etc. and the time cost of porting all the stuff to another system?

No, it is not. Please take a look at <http://www.mysql.com/products/licensing.html>. In their simplest form, the following are general licensing guidelines:

- If your software is licensed under either the GPL-compatible Free Software License as defined by the Free Software Foundation or approved by OSI, then use our GPL licensed version.
- If you distribute a proprietary application in any way, and you are not licensing and distributing your source code under GPL, you need to purchase a commercial license of MySQL
- If you are unsure, we recommend that you buy our cost effective commercial licenses. That is the safest solution. Licensing questions can be directed to licensing@mysql.com for our advice, and we encourage you to refer to the Free Software Foundation or a lawyer as appropriate.

In other words that means: If you don't release the source for your application under the GPL you need to acquire a commercial license for mySQL. A look at the pricing for commercial mySQL license:

<http://www.mysql.com/products/pricing.htm>

The MySQL Server license is per database server (single installed MySQL binary). There are no restrictions on the number of connections, number of CPUs, memory or disks to that one database server so these are per deployment royalties.

Number of licenses: 1 .. 9 - Price per copy: EUR 440 / USD 495

So from only **TWO** deployed servers on NexusDB at \$900 per Developer is cheaper for a single Developer than mySQL. This doesn't even take into account the costs for whatever component you plan to use to access the mySQL Server, which is normally licensed per Developer as well.

Q: I had a discussion with a colleague. He told me: "Why do you use NexusDB? Why don't you use standards like Oracle, MSSQL, ..?" I kept in silence, so: What would you say in this case?

As honest as possible: there are reasons to use NexusDB and reasons to use other products. NexusDB has a real strength in three areas: vertical applications, embedded within application servers and embedded within desktop apps. NexusDB is competitive in just about any other area as well, but in some cases there are reasons to select the other db's.

For **vertical applications**, it completely removes the need for a DBA and is extremely easy to install. If you are an ISV and are selling licenses/deployments of your system, NexusDB enables you to add those "standard" database licensing fees *directly* to your bottom line. The business reasons are undeniably in NexusDB's favor.

For **embedded applications** NexusDB is far superior to anything else out there. First, you get a powerful full-blown server that supports direct table access in addition to SQL. No other database gives you this power and flexibility. Second, the performance is better than the other embedded DB's (we suggest you run your own numbers if you doubt it). Third, you can expose NexusDB to external report writers or clients. This is an awesome capability for ISV's developing vertical applications.

Some companies use an embedded NexusDB server in their application server to implement **caching in the middle tier**. They put lookup tables and other things that take a long time to calculate in there. It's easy to do and improves the overall responsiveness of the system for users.

Overall, there is a lot about NexusDB and its flexibility and ability to **complementing other databases** makes it extremely useful.

Oracle is a fine database, but its *way* too bloated and too pricey for many of the applications it's used for. Even the Oracle Client is bloated...500MB install or something? If your customers or management like paying (at least) 10 times more (and for several years) for something that will do the same job as NexusDB then no amount of rational argument can dissuade them.

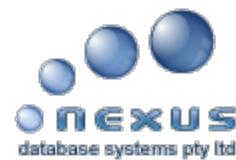
MSSQL is another fine database, but it is also becoming pricey. If your customers or management want a "standard" database, this is the one we'd recommend. The reason why people want "standard" databases is so they can minimize risk and there's not much you can do dissuade them either.

Several reasons in no particular order why NexusDB is very favourable choice:

- support; can't get better than direct access to the developers of the code
- full source code
- superb design and implementation
- royalty free distribution
- easy to deploy/maintain
- flexibility of deployment options (c/s or embedded)
- performance
- full featured

Q: Why NexusDB can be much better choice than Interbase/Firebird?

- Interbase doesn't give you real embedded capability.
- Interbase has deployment fees
- Firebird and Interbase require SQL access which may be all you want/need, but other times I find that TTable coding much better/productive.
- Firebird doesn't scale on SMP machines
- Firebird support is typically more expensive than NexusDB's overall cost
- Firebird is written in C++; NexusDB has code that can be used beyond the database (memory manager, transports, service application wrapper, etc).



15 Introduction

Introduction

Welcome to NexusDB SQL Reference

By choosing NexusDB as your RDBMS, you have acquired access to a modern, state of the art database technology, exposed through standard client data access technologies like Microsoft ODBC and ADO.NET, and Borland VCL and DBExpress.

NexusDB SQL is an interface to the NexusDB core-engine, an advanced and highly efficient relational database management server. This manual provides the information needed to take advantage of the built-in SQL language in NexusDB.

Featuring SQL:2003

NexusDB implements a large subset of the international standard ISO/IEC 9075 - SQL:2003, including most of the Core SQL functionality as well as many of the additional features defined in the standard. In addition, NexusDB augments SQL:2003 with extensions to expose vendor-specific functionality.

While NexusDB does not claim official conformance to Core SQL, the grammar conforms closely to the syntax and processing rules defined in SQL:2003. Any difference between NexusDB SQL and SQL:2003 is documented in the Conformance summary at the bottom of each topic page.

The advantage of using standard SQL

While SQL has been the dominant database language for more than three decades, and also being standardized since 1986, the major RDBMS vendors have implemented proprietary grammar and SQL dialects to meet demands in the market not covered by the standard at the time. The result has been that SQL statements tuned to execute efficiently on a particular RDBMS, will often fail when ported to a different RDBMS.

The recent SQL standards, starting with SQL-92, and particularly the SQL:1999 major revision and the latest SQL:2003 minor revision, implement all commercially accepted functionality found in the proprietary implementations, and probably more than a single vendor will ever offer.

It is expected that major RDBMS vendors are currently working on adopting their implementations to the recent standards. By exposing SQL grammar that conforms to SQL:2003 now, NexusDB users have the advantage of being able to write standard SQL statements that can target a variety of RDBMS in a foreseeable future.

NexusDB SQL main features

- Standard SQL:2003

- Views
 - Referential Integrity
 - Triggers - Active Database
 - Stored Procedures and Functions
 - SQL/PSM Procedure Language
 - Transaction Management
 - Fulltext Index Support
-

About NexusDB SQL Reference

16 About NexusDB SQL Reference

About NexusDB SQL Reference

The intention of this manual is to provide a complete reference of the SQL language used in NexusDB, with detailed documentation of the syntax and proper explanation of the usage.

It is assumed that readers of the manual have at least basic knowledge of SQL. While the many examples throughout the manual are well suited as small tutorials, illustrating how the grammar and syntactic elements are used in SQL statements, we recommend that users who want to learn SQL well, read a good text book about standard SQL.

Since NexusDB has a modern SQL implementation based on the latest SQL:2003 standard, books describing SQL:1999 or SQL:2003 would be an excellent source. Application programmers and database administrators who want a more comprehensive presentation and documentation of standard SQL, might want to acquire a copy of the excellent SQL:1999 book by Jim Melton (the editor of SQL-92, SQL:1999 and SQL:2003), or even the SQL:2003 standard itself:

- SQL:1999 Understanding Relational Language Components
Jim Melton, Alan R. Simon
ISBN 1-55860-456-1

- ISO/IEC 9075:2003
<http://www.iso.org>

Organization and layout of topics

The table of contents (TOC) has been organized in chapters and related sub-chapters following the structure of the main parts of the SQL language.

The topic pages have generally been structured as follows:

- Syntax

- Usage

- Notes

- Examples

- Conformance summary

In some of the major topic pages, like the CREATE TABLE statement and the SELECT statement, the Notes section has been split into several parts to detail the documentation of complex syntax and functionality.

The **Conformance** summary at the bottom of each topic page documents the level of conformance to SQL:2003, differences between the standard and NexusDB SQL, and vendor-specific extensions.

BNF conventions

The syntax of the SQL grammar is documented by using a variation of BNF (Backus Normal Form). A BNF production consists of three parts, a non-terminal symbol enclosed in angle brackets, a definition operator (::=) and a production rule defining the syntactic elements on the left side. The production rule may include one or more non-terminal symbols that are resolved in separate BNF productions. The following example shows a simple BNF production:

```
<current time value function> ::= CURRENT_TIME
| LOCALTIME
```

Since NexusDB SQL is an implementation of SQL:2003, most of the BNF productions are based on standard SQL definitions, but simplified when symbols are self-explanatory. For example, instead of defining the non-terminal symbol <table name> in a separate BNF production, we are using the terminal symbol **table-name**.

Symbols used in BNF

Symbol	Description
< >	A character string enclosed in angle brackets is the name of a non-terminal symbol.
::=	The definition operator is used in a production rule to separate the element defined by the rule from its definition. The element being defined appears to the left of the operator and the formula that defines the element appears to the right.
[]	Square brackets indicate optional elements in a formula. The portion of the formula within the brackets may be explicitly specified or may be omitted.
{ }	Braces group elements in a formula. The portion of the formula within the braces shall be explicitly specified.
	The alternative operator. The vertical bar indicates that the portion of the formula following the bar is an alternative to the portion preceding the bar. If the vertical bar appears at a position where it is not enclosed in braces or square brackets, it specifies a complete alternative for the element defined by the production rule. If the vertical bar appears in a portion of a formula enclosed in braces or square brackets, it specifies alternatives for the contents of the innermost pair of such braces or brackets.
...	The ellipsis indicates that the element to which it applies in a formula may be repeated any number of times. If the ellipsis appears immediately after a closing brace, then it applies to the portion of the formula enclosed between that closing brace and the corresponding opening brace . If an ellipsis appears after any other element, then it applies only to that element.

While the BNF productions may look rather complex at first, they provide a very precise documentation of the SQL grammar. We believe that users who are unfamiliar with BNF, will gain a better understanding of the flexibility and power of NexusDB SQL by spending some time on studying the BNF for frequently used SQL statements and other syntactic elements.

In the online version of this manual, all BNF non-terminal symbols that are part of a production rule, appear as green colored hot-links. Just click the indexed word with the mouse to jump directly to the topic page where the symbol is resolved.

Examples

A large part of the examples provided in the manual are referencing the "Students Database", a small demo database that ships with the NexusDB installation. The "Students Database" is a collection of the following tables:

- students
- courses
- enrolls
- sections
- teachers

Before executing example statements that are making changes to these tables, it is recommended to backup the "Students Database" first. Another way of working with the sample data without persisting any changes, is to make temporary versions of the tables using either a CREATE TABLE statement with the AS subquery clause, or a SELECT INTO statement.

In the examples throughout the manual, we have chosen the notation of writing keywords in upper-case and identifiers in lower-case or mixed case to clearly distinguish between two in the text. However, the SQL language is case-insensitive to keywords and regular identifiers, and in NexusDB SQL also to delimited identifiers, meaning that SQL statements can be written in any case.

17 SQL Language Elements

SQL Language Elements

This section describes the language elements used in SQL statements.

- Keywords
- Identifiers
- Comments
- Operators
- Null Values
- Literals
- Value Expressions
- Parameters

17.1 Keywords

Keywords

Keywords are words recognized by SQL to have a distinct meaning within the context in which they appear. Keywords are used in SQL statements to identify the statement itself, the SQL grammar that specifies processing actions and the SQL elements used by the statement.

The SQL language is case-insensitive. Therefore, keywords, identifiers and other names written in upper case, lower case or mixed case with the same spelling mean the same thing. Throughout this manual, keywords are written in upper case to distinguish keywords from identifiers.

Many SQL keywords are reserved words. Reserved words cannot be used as names for database objects such as tables, columns, views etc. unless the word is enclosed by double quotes. For example, "Date" is a valid column name, while Date would cause an exception.

Overview of keywords in NexusDB SQL

Keyword	Res.	Conformance	Keyword	Res.	Conformance
A					
ABS	✓	SQL:2003	ADD		SQL:2003
AFTER		SQL:2003	ALL	✓	SQL:2003
ALTER	✓	SQL:2003	AND	✓	SQL:2003
ANY	✓	SQL:2003	AS	✓	SQL:2003
ASC		SQL:2003	ASSEMBLY	✓	NexusDB ext
ASSERT	✓	NexusDB ext	ATAN	✓	NexusDB ext

ATAN2	✓	NexusDB ext	ATN2	✓	NexusDB ext
ATOMIC	✓	SQL:2003	AUTHORIZATION	✓	SQL:2003
AUTOINC	✓	NexusDB ext	AVG	✓	SQL:2003
B					
BEFORE		SQL:2003	BEGIN	✓	SQL:2003
BETWEEN	✓	SQL:2003	BIGINT	✓	SQL:2003
BINARY	✓	SQL:2003	BLOB	✓	SQL:2003
BLOCK		NexusDB ext	BLOCKSIZE		NexusDB ext
BOOL	✓	NexusDB ext	BOOLEAN	✓	SQL:2003
BOTH	✓	SQL:2003	BROUND	✓	NexusDB ext
BY	✓	SQL:2003	BYTE	✓	NexusDB ext
BYTARRAY	✓	NexusDB ext			
C					
CALL	✓	SQL:2003	CALLED	✓	SQL:2003
CASCADE		SQL:2003	CASE	✓	SQL:2003
CAST	✓	SQL:2003	CATCH	✓	NexusDB ext
CEIL	✓	SQL:2003	CEILING	✓	SQL:2003
CHAR	✓	SQL:2003	CHAR_LENGTH	✓	SQL:2003
CHARACTER	✓	SQL:2003	CHARACTER_LENGTH	✓	SQL:2003
CHARACTERS		SQL:2003	CHECK	✓	SQL:2003
CHR	✓	NexusDB ext	CLOB	✓	SQL:2003
CLR		NexusDB ext	COALESCE	✓	SQL:2003
CODEPAGE		NexusDB ext	COLLATE	✓	SQL:2003
COLLATION		SQL:2003	COLUMN	✓	SQL:2003
COMMIT	✓	SQL:2003	CONSTRAINT	✓	SQL:2003
CONTAINS		SQL:2003	COS	✓	NexusDB ext
COUNT	✓	SQL:2003	CREATE	✓	SQL:2003
CROSS	✓	SQL:2003	CURRENT_DATE	✓	SQL:2003
CURRENT_TIME	✓	SQL:2003	CURRENT_TIMESTAMP	✓	SQL:2003
CURRENT_USER	✓	SQL:2003			
D					
DATA		SQL:2003	DATE	✓	SQL:2003
DATETIME	✓	NexusDB ext	DAY	✓	SQL:2003
DEC	✓	SQL:2003	DECIMAL	✓	SQL:2003
DECLARE	✓	SQL:2003	DEFAULT	✓	SQL:2003
DELETE	✓	SQL:2003	DELETING	✓	NexusDB ext
DESC		SQL:2003	DESCRIPTION		NexusDB ext

DETERMINISTIC	✓	SQL:2003	DISTINCT	✓	SQL:2003
DO	✓	SQL:2003	DOUBLE	✓	SQL:2003
DROP	✓	SQL:2003	DWORD	✓	NexusDB ext
E					
EACH	✓	SQL:2003	ELSE	✓	SQL:2003
ELSEIF	✓	NexusDB ext	EMPTY	✓	NexusDB ext
ENCRYPT		NexusDB ext	ENCRYPTION		NexusDB ext
END	✓	SQL:2003	ENGINE		NexusDB ext
EQUIVALENT	✓	NexusDB ext	ERROR_MESSAGE	✓	NexusDB ext
ESCAPE	✓	SQL:2003	EXCEPT	✓	SQL:2003
EXISTS	✓	SQL:2003	EXP	✓	SQL:2003
EXTENDED	✓	NexusDB ext	EXTERNAL	✓	SQL:2003
EXTRACT	✓	SQL:2003			
F					
FALSE	✓	SQL:2003	FIRST		SQL:2003
FLOAT	✓	SQL:2003	FLOOR	✓	SQL:2003
FOR	✓	SQL:2003	FOREIGN	✓	SQL:2003
FROM	✓	SQL:2003	FULL	✓	SQL:2003
FUNCTION	✓	SQL:2003			
G					
GLOBAL	✓	SQL:2003	GROUP	✓	SQL:2003
GROW		NexusDB ext	GROWSIZE		NexusDB ext
GUID	✓	NexusDB ext			
H					
HAVING	✓	SQL:2003	HOUR	✓	SQL:2003
I					
IDENTITY	✓	SQL:2003	IF	✓	SQL:2003
IGNORE	✓	NexusDB ext	IMAGE	✓	NexusDB ext
IN	✓	SQL:2003	INDEX	✓	NexusDB ext
INITIAL		NexusDB ext	INITIALSIZE		NexusDB ext
INNER	✓	SQL:2003	INOUT	✓	SQL:2003
INPUT		SQL:2003	INSERT	✓	SQL:2003
INSERTING	✓	NexusDB ext	INT	✓	SQL:2003
INTEGER	✓	SQL:2003	INTERSECT	✓	SQL:2003
INTERVAL	✓	SQL:2003	INTO	✓	SQL:2003
IS	✓	SQL:2003	ITERATE	✓	SQL:2003

J						
JOIN		✓	SQL:2003			
K						
KANA		NexusDB ext	KEY	SQL:2003		
L						
LANGUAGE		✓	SQL:2003	LARGE	✓	SQL:2003
LARGEINT		✓	NexusDB ext	LAST	SQL:2003	
LASTAUTOINC		✓	NexusDB ext	LEADING	✓	SQL:2003
LEAVE		✓	SQL:2003	LEFT	✓	SQL:2003
LIKE		✓	SQL:2003	LIST	✓	NexusDB ext
LN		✓	SQL:2003	LOCAL	✓	SQL:2003
LOCALE		✓	NexusDB ext	LOCALTIME	✓	SQL:2003
LOCALTIMESTAMP		✓	SQL:2003	LOWER	✓	SQL:2003
M						
MATCH		✓	SQL:2003	MAX	✓	SQL:2003
MED		✓	NexusDB ext	MIN	✓	SQL:2003
MINUTE		✓	SQL:2003	MOD	✓	SQL:2003
MODIFIES		✓	SQL:2003	MONEY	✓	NexusDB ext
MONTH		✓	SQL:2003			
N						
NAME		SQL:2003	NATIONAL	✓	SQL:2003	
NATURAL		✓	SQL:2003	NCHAR	✓	SQL:2003
NCLOB		✓	SQL:2003	NEW	✓	SQL:2003
NEWGUID		✓	NexusDB ext	NO	✓	SQL:2003
NONSPACE		NexusDB ext	NOT	✓	SQL:2003	
NSINGLECHAR		✓	NexusDB ext	NULL	✓	SQL:2003
NULLIF		✓	SQL:2003	NULLS	SQL:2003	
NULLSTRING		✓	NexusDB ext	NUMERIC	✓	SQL:2003
NVARCHAR		✓	NexusDB ext			
O						
OBJECT		SQL:2003	OCTET_LENGTH	✓	SQL:2003	
OCTETS		SQL:2003	ODD	✓	NexusDB ext	
OF		✓	SQL:2003	OLD	✓	SQL:2003
ON		✓	SQL:2003	OR	✓	SQL:2003
ORD		✓	NexusDB ext	ORDER	✓	SQL:2003
OUT		✓	SQL:2003	OUTER	✓	SQL:2003

P					
PARTIAL		SQL:2003	PASSWORDS	✓	NexusDB ext
PERCENT		NexusDB ext	PI	✓	NexusDB ext
POSITION	✓	SQL:2003	POWER	✓	SQL:2003
PRECISION	✓	SQL:2003	PRIMARY	✓	SQL:2003
PROCEDURE	✓	SQL:2003			
R					
RAND	✓	NexusDB ext	READS	✓	SQL:2003
REAL	✓	SQL:2003	RECREV	✓	NexusDB ext
REFERENCES	✓	SQL:2003	REFERENCING	✓	SQL:2003
REMOVE		NexusDB ext	REPEAT	✓	SQL:2003
RESTRICT		SQL:2003	RETURN	✓	SQL:2003
RETURNS	✓	SQL:2003	RIGHT	✓	SQL:2003
ROLLBACK	✓	SQL:2003	ROUND	✓	NexusDB ext
ROUTINE		SQL:2003	ROW	✓	SQL:2003
ROWSAFFECTED	✓	NexusDB ext	ROWSREAD	✓	NexusDB ext
S					
SECOND	✓	SQL:2003	SELECT	✓	SQL:2003
SERIALIZABLE		SQL:2003	SESSION_USER	✓	SQL:2003
SET	✓	SQL:2003	SHORTINT	✓	NexusDB ext
SHORTSTRING	✓	NexusDB ext	SIGNAL	✓	SQL:2003
SIMPLE		SQL:2003	SIN	✓	NexusDB ext
SINGLECHAR	✓	NexusDB ext	SMALLINT	✓	SQL:2003
SNAPSHOT		NexusDB ext	SOME	✓	SQL:2003
SORT		NexusDB ext	SQL	✓	SQL:2003
SQRT	✓	SQL:2003	START	✓	SQL:2003
STD	✓	NexusDB ext	STORAGE		NexusDB ext
STRING		NexusDB ext	SUBSTRING	✓	SQL:2003
SUM	✓	SQL:2003	SYMBOLS		NexusDB ext
SYSTEM_ROW#	✓	NexusDB ext			
T					
TABLE	✓	SQL:2003	TEMPORARY		SQL:2003
TEXT	✓	NexusDB ext	THEN	✓	SQL:2003
TIME	✓	SQL:2003	TIMESTAMP	✓	SQL:2003
TINYINT	✓	NexusDB ext	TO	✓	SQL:2003
TOP		NexusDB ext	TOSTRING	✓	NexusDB ext
TOSTRINGLEN	✓	NexusDB ext	TRAILING	✓	SQL:2003

TRANSACTION		SQL:2003	TRIGGER	✓	SQL:2003
TRIM	✓	SQL:2003	TRUE	✓	SQL:2003
TRY	✓	NexusDB ext	TYPE		SQL:2003
U					
UNION	✓	SQL:2003	UNIQUE	✓	SQL:2003
UNKNOWN	✓	SQL:2003	UNTIL	✓	SQL:2003
UPDATE	✓	SQL:2003	UPDATING	✓	NexusDB ext
UPPER	✓	SQL:2003	USE		NexusDB ext
USER	✓	SQL:2003	USING	✓	SQL:2003
V					
VALUES	✓	SQL:2003	VARCHAR	✓	SQL:2003
VARYING	✓	SQL:2003	VIEW		SQL:2003
W					
WHEN	✓	SQL:2003	WHERE	✓	SQL:2003
WHILE	✓	SQL:2003	WIDTH		NexusDB ext
WITH	✓	SQL:2003	WORD	✓	NexusDB ext
WORK		SQL:2003			
Y					
YEAR	✓	SQL:2003			

17.2 Identifiers

Identifiers

SQL provides two types of identifiers that are used to specify the names of database objects, such as tables, columns, constraints and indexes, views, triggers, and user-defined procedures and functions.

NexusDB only allows codepage neutral ANSI characters in identifiers, that is characters having the same ordinal value regardless of codepage. The following table shows the valid characters that can be used in identifiers:

Character	Code	Symbol
Space	#32	
Exclamation mark	#33	!
Pound sign	#35	#
Dollar sign	#36	\$

Percent	#37	%
Ampersand	#38	&
Left parenthesis	#40	(
Right parenthesis	#41)
Plus sign	#43	+
Minus sign	#45	-
Digit	#48..#57	0..9
At sign	#64	@
Upper case letter	#65..#90	A..Z
Left bracket	#91	[
Right bracket	#93]
Circumflex	#94	^
Underscore	#95	_
Lower case letter	#97..#122	a..z
Left brace	#123	{
Right brace	#125	}
Tilde	#126	~

Regular identifiers

A regular identifier is an unquoted string of characters.

- Regular identifiers cannot contain spaces or any of the following characters:

Percent (#37)
 Ampersand (#38)
 Left parenthesis (#40)
 Right parenthesis (#41)
 Plus sign (#43)
 Minus sign (#45)
 Left bracket (#91)
 Right bracket (#93)

- Regular identifiers cannot begin with a decimal digit.
- Regular identifiers are case-insensitive.
- The maximum length of regular identifiers is 128 characters.

Delimited identifiers

A delimited identifier is a string of characters enclosed by double quotation marks.

- Delimited identifiers can contain spaces, but not as leading or trailing characters.
- Delimited identifiers are case-insensitive in NexusDB SQL, while they are case-sensitive in SQL:2003.
- The maximum length of delimited identifiers is 128 characters.

Schema-qualified names

All persistent database objects, such as tables, views, triggers and user-defined procedures and functions, have a two-part name consisting of the schema name and the object name separated by a period. For example, a table with the name "students" that is stored in a database called "students database", can be referenced in either of the following ways:

- "students database"."students"
- "students database".students
- "students"
- students
- Schema and database object names shall be a valid regular or delimited identifier.
- NexusDB table names must additionally follow the requirements for valid file names in Windows.
- A database object that is stored in the current database may be referenced by its unqualified one-part name, or the schema-qualified two-part name for clarity.
- A database object that is stored in a database other than the current database, must be referenced by its schema-qualified two-part name.

17.3 Comments

Comments

Comments are text embedded in SQL statements without being executed as part of the statement.

Simple comments

NexusDB SQL supports two variations of simple comments. The first variation, which is the syntax defined by SQL:2003, starts with two minus signs (--). The second variation, which is known from traditional programming languages like C and others, starts with two slashes (//). Simple comments can only appear in single lines and are terminated by the end of the line.

The following example illustrates the use of simple comments:

```
// The three pound symbols define a global temporary table
CREATE TABLE ###Temp (
    c1 AUTOINC NOT NULL PRIMARY KEY, -- We
    don't allow null values in this column
    c2 VARCHAR(50) // This is an ordinary character
    string column
)
```

Bracketed comments

Bracketed comments start with a slash followed by an asterisk and end with an asterisk followed by a slash. Bracketed comments can be embedded anywhere in an SQL statement and may span multiple lines.

The following example illustrates the use of bracketed comments:

```
/*
SQL statements must be terminated with a
semicolon when they're executed as a sequence
(statement block).
We need to drop the table before recreating it.
*/
/* The IF EXISTS clause prevents an exception if the table doesn't exist */
DROP TABLE IF EXISTS /*IF EXISTS is a NexusDB extension*/ ###Temp;
CREATE TABLE ###Temp (
    c1 AUTOINC NOT NULL PRIMARY KEY,
    c2 VARCHAR(50)
);
```

Conformance

SQL:2003 standard - Core SQL
 - Feature T351 "Bracketed comments"

NexusDB extensions - C-style comments

17.4 Operators

Operators

Operators are symbols and keywords that specify an operation to be performed on one or more value expressions.

Arithmetic operators

The arithmetic operators perform operations on numeric values.

Operator	Symbol	Operation
<addition operator>	+	Addition
<subtraction operator>	-	Subtraction
<multiplication operator>	*	Multiplication
<division operator>	/	Division

Note: The addition (+) and subtraction (-) operators are also used in arithmetic operations on datetime values.

See also: Numeric Value Functions.

Concatenation operators

The concatenation operator is used to concatenate string values.

Operator	Symbol	Operation
<concatenation operator>		String concatenation

Note: The arithmetic addition (+) operator can be used as an alternate string concatenation operator.

See also: String Value Functions.

Comparison operators

The comparison operators are used in some predicates to specify comparison tests.

Operator	Symbol	Operation
<equals operator>	=	Equal to
<not equals operator>	<>	Not equal to
<less than operator>	<	Less than
<greater than operator>	>	Greater than
<less than or equals operator>	<=	Less than or equal to
<greater than or equals operator>	>=	Greater than or equal to

Note: The equals sign (=) is also used as an assignment operator.

Boolean operators

The boolean operators are used to combine predicates in search conditions.

Operator	Keyword	Operation
<boolean NOT operator>	NOT	Invert the result of the boolean test
<boolean AND operator>	AND	Test if both predicates are TRUE
<boolean OR operator>	OR	Test if either of the predicates is TRUE

Note: Check the truth tables to see how combined boolean expressions are evaluated.

Assignment operator

The equals sign is used for value assignments.

Operator	Symbol	Operation
<assignment operator>	=	Value assignment

Operator precedence

Operator precedence determines the order in which operations are performed during the execution of a complex value expression with multiple operators.

Operators are evaluated in the following order, ranked from highest to lowest precedence level:

- * (multiplication), / (division)

- + (addition), - (subtraction)
- ||, + (concatenation)
- =, <>, <, >, <=, >= (comparison operators)
- NOT
- AND
- OR

Operators on the same precedence level are evaluated left to right based on their position in the value expression.

If a value expression has nested parentheses, the innermost expression is evaluated first.

Tip: Instead of relying on the defined operator precedence, it is recommended to use parentheses to explicitly specify the order of operations in complex value expressions with many operators. All operations inside parentheses are evaluated first to yield a single value before that value is used as an operand in other operations.

Conformance

SQL:2003 standard - Core SQL

NexusDB - Plus sign as string concatenation operator
extensions

17.5 Null Values

Null Values

SQL uses a special value, the **null value**, to indicate that a data element does not have a value. Every data element, such as table columns, variables, parameters, value expressions, the result of functions, etc., can have a null value. An exception to this rule are columns with the NOT NULL constraint, which prohibits columns from containing the null value in any row.

The null value is different from any valid value, and is therefore not the same as zero, blanks or an empty string. This concept is often confusing to programmers more familiar with programming languages like C and Pascal that define a "null string" to mean a string with length 0. In SQL, an empty string is a **known** value (it has no characters), while null is an **unknown** value.

SQL provides the keyword NULL to represent a null value in SQL statements. However, since the keyword does not associate a data type with the null value, SQL also provides a syntactic ability to associate a data type with the null value by using an explicit cast:

CAST(NULL AS data-type)

As a consequence of the null value concept, SQL uses a three-valued logic (TRUE, FALSE, UNKNOWN) when evaluating boolean expressions.

The main characteristics of the null value are summarized below:

- The null value is an unknown value.
 - Null values are not distinct, meaning that it's not possible to distinguish between two null values.
 - Null values are treated as a single group in grouping operations.
 - Null values are sorted before all non-null values by default. The null ordering sort option may however specify that nulls are sorted last.
 - Null values as operands of arithmetic operations and string concatenations, or as function arguments, will cause the result to be null.
 - Some of the predicates, but not all, will evaluate to UNKNOWN if one of the predicands is the null value.
-

Conformance

SQL:2003 standard - Core SQL

17.6 Literals

Literals

Specify a non-null value.

Syntax

```

<literal> ::= 
  | <signed numeric literal>
  | <general literal>

<unsigned literal> ::= 
  | <unsigned numeric literal>
  | <general literal>

<general literal> ::= 
  | <character string literal>
  | <national character string literal>
  | <guid literal>
  | <binary string literal>
  | <datetime literal>
  | <interval literal>
  | <boolean literal>

```

Usage

Literals, commonly perceived as constants, can be used in SQL everywhere a value can be specified. Since literals have a data type, they can only be used in places where data of the particular type is permitted.

Conformance

SQL:2003 standard - Core SQL

- Feature F421 "National character"
 - Feature T041 "Basic LOB data type support"
 - Feature F052 "Intervals and datetime arithmetic"
 - Feature T031 "BOOLEAN data type"
-

Numeric Literals

String Literals

Date/Time Literals

Interval Literals

Boolean Literals

17.6.1 Numeric Literals

Numeric Literals

Syntax

```

<signed numeric literal> ::= [ + | - ] <unsigned numeric literal>

<unsigned numeric literal> ::=
| <exact numeric literal>
| <approximate numeric literal>

<exact numeric literal> ::=
| <unsigned integer> [ . [ <unsigned integer> ] ]
. <unsigned integer>

<unsigned integer> ::= <digit>...
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<approximate numeric literal> ::= <mantissa> E <exponent>

<mantissa> ::= <exact numeric literal>

<exponent> ::= <signed integer>

<signed integer> ::= [ + | - ] <unsigned integer>

```

Notes

- An exact numeric literal without a decimal fraction specified denotes an integer value. The implicit data type is BIGINT.
- An exact numeric literal with a decimal fraction specified denotes a BCD value. The implicit data type is DECIMAL.
- The number of digits following the decimal point is determining the implicit scale of an exact numeric literal. If the specified scale exceeds the maximum scale of 4, then the value is truncated to scale 4.

Examples

Data Type	Literal Examples
SHORTINT	96
SMALLINT	5247
INTEGER	258476
BIGINT	2147483648
NUMERIC	14.99 15.

DECIMAL	14.99 15.
FLOAT	1.56E-4 200E10
REAL	1.56E-4
DOUBLE PRECISION	3.1415929265432E00
EXTENDED	3.1415929265432E00

Conformance

SQL:2003 standard - Core SQL

17.6.2 String Literals

String Literals

Syntax

```

<character string literal> ::= 
    <quote> [ <character representation>... ] <quote>
    [ { separator <quote> [ <character representation>... ] <quote> }... ]

<national character string literal> ::= 
    [ N ] <quote> [ <character representation>... ]
    [ { separator <quote> [ <character representation>... ] <quote> }... ]

<guid literal> ::= 
    [ GUID ] <quote> <character representation>... <quote>

<binary string literal> ::= 
    X <quote> [ { <hexit> <hexit> }... ] <quote>
    [ { separator <quote> [ { <hexit> <hexit> }... ] <quote> }... ]

<character representation> ::= 
    | nonquote-character
    <quote symbol>

<quote symbol> ::= <quote><quote>

<quote> ::= '

<hexit> ::= <digit> | A | B | C | D | E | F | a | b | c | d | e | f

```

Notes

- The notation for embedding a quote in a character string literal is two quote characters without space between them.
- The N prefix is needed to implicitly cast a literal to a national character string, but is optional if the assignment target is a national character string type.
- Character string literals are comparable to any character string independent of the collation, and they are implicitly converted to the character set and collation of the target site in an assignment.
- The character representation of a GUID literal shall conform to the format of a GUID value, see the example below. The GUID prefix is optional if the assignment target is a GUID type.
- The V1 syntax of specifying a binary string literal as hexadecimal values inside square brackets is still supported, but the extension has been deprecated in favor of standard SQL syntax using the X prefix, and may be removed from a future version of NexusDB SQL.
- Feature F271 "Compound character literals" provides the syntactical ability to split string literals onto multiple lines by separating each part with a line-feed.

Examples

Data Type	Literal Examples
CHAR	'A character string literal is a sequence of characters enclosed in single quotes.'
VARCHAR	'The quote character (") is embedded in a literal by using two quote characters.'
CLOB	'Character string literals are also used with CLOB types.' 'String literals may be split onto multiple lines like this example.'
NCHAR	N'In formal SQL:2003 syntax, national character string literals are prefixed with a N.'
NCHAR	'NexusDB SQL supports national character string literals without the N prefix.'
GUID	GUID '{E5C37F97-0F9F-47FB-8B55-30FDEC0026E1}'
BLOB	X4E6578757344422056322069732061206772656174205244424D532E'

Conformance

SQL:2003 standard - Core SQL

- Feature F421 "National character"
- Feature T041 "Basic LOB data type support"
- Feature F271 "Compound character literals"

NexusDB
extensions

- The N prefix in a national character string literal is optional
- GUID literal
- Binary literal notation with hexadecimal values inside square brackets

17.6.3 DateTime Literals

DateTime Literals

Syntax

```

<datetime literal> ::= 
  | <date literal>
  | <time literal>
  | <timestamp literal>

<date literal> ::= DATE <date string>

<time literal> ::= TIME <time string>

<timestamp literal> ::= TIMESTAMP <timestamp string>

<date string> ::= <quote> year-month-day <quote>

<time string> ::= <quote> hour:minute:second [ . [ millisecond ] ] <quote>

<timestamp string> ::= <quote> year-month-day hour:minute:second [ . [ millisecond ] ] <quote>
  
```

Examples

Data Type	Literal Examples
DATE	DATE '2004-09-25'
TIME	TIME '14:28:36'
TIMESTAMP	TIMESTAMP '2004-09-25 14:28:36'

Conformance

SQL:2003 standard - Core SQL

17.6.4 Interval Literals

Interval Literals

Syntax

```
<interval literal> ::=  
    INTERVAL [ + | - ] <interval string> <interval qualifier>  
  
<interval string> ::=  
    <quote> [ + | - ] { <year-month literal> | <day-time literal> } <quote>  
  
<year-month literal> ::=  
    | years [ -months ]  
    months  
  
<day-time literal> ::=  
    | <day-time interval>  
    <time interval>  
  
<day-time interval> ::=  
    day [ space hours [ :minutes [ :seconds [ . [ milliseconds ] ] ] ] ]  
  
<time interval> ::=  
    | hours [ :minutes [ :seconds [ . [ milliseconds ] ] ] ]  
    | minutes [ :seconds [ . [ milliseconds ] ] ]  
    seconds [ . [ milliseconds ] ]  
  
<interval qualifier> ::=  
    | <non-second primary datetime field> TO <primary datetime field>  
    <primary datetime field>  
  
<primary datetime field> ::=  
    | <non-second primary datetime field>  
    SECOND  
  
<non-second primary datetime field> ::=  
    | YEAR  
    | MONTH  
    | DAY  
    | HOUR  
    | MINUTE  
    | DAYOFYEAR  
    | ISODAYOFWEEK  
    | ISO_WEEK  
    | ISOMONTH  
    | ISOYEAR  
    | ISO_WEEKOFMONTH  
    | ISODAYOFMONTH
```

ISODAYOFYEAR

Notes

- Interval literals can only be used in datetime arithmetic.
- If the interval string contains a year-month literal, then the interval qualifier shall not specify DAY, HOUR, MINUTE or SECOND. If the interval string contains a day-time literal, then the interval qualifier shall not specify YEAR or MONTH.
- If TO is specified, then the start field shall be more significant than the end field, and the start field shall not specify MONTH. If the start field specifies YEAR, then the end field shall specify MONTH.
- DAYOFYEAR - day of the calendar year (1 to 365/366 starting on year-01-01)
ISODAYOFWEEK - day of week according to ISO 8601 (1 = monday to 7 = sunday)
ISO_WEEK - week of year according to ISO 8601
ISOMONTH - month of year according to ISO 8601x (can be different from calendar month)
ISOYEAR - year of week according to ISO 8601 (can be different from calendar year)
ISO_WEEKOFMONTH - week of month according to ISO 8601x
ISODAYOFMONTH - day of month according to ISO 8601x
ISODAYOFYEAR - day of year according to ISO 8601

Examples

Literal Examples

CURRENT_TIMESTAMP + INTERVAL '2' YEAR
CURRENT_TIMESTAMP + INTERVAL '2-6' YEAR TO MONTH
CURRENT_TIMESTAMP + INTERVAL '6' MONTH
CURRENT_TIMESTAMP + INTERVAL '15' DAY
CURRENT_TIMESTAMP + INTERVAL '15 12' DAY TO HOUR
CURRENT_TIMESTAMP + INTERVAL '12' HOUR
CURRENT_TIMESTAMP + INTERVAL '12:25' HOUR TO MINUTE
CURRENT_TIMESTAMP + INTERVAL '12:25:30' HOUR TO SECOND
CURRENT_TIMESTAMP + INTERVAL '25' MINUTE
CURRENT_TIMESTAMP + INTERVAL '25:30' MINUTE TO SECOND

The following SQL can be used to check out samples of the returned values:

```
CREATE LOCAL TEMPORARY TABLE #dates (
    dt DATE
);
```

```
DECLARE StartDT, EndDT, CurrentDT DATE;

SET StartDT = CURRENT_TIMESTAMP;
SET EndDT = StartDT + INTERVAL '5' YEAR;

SET CurrentDT = StartDT;

WHILE CurrentDT < EndDT DO
    INSERT INTO #dates (dt) VALUES (CurrentDT);
    SET CurrentDT = CurrentDT + INTERVAL '1' DAY;
END WHILE;

SELECT
    *,
    EXTRACT(DAYOFYEAR FROM dt) AS "DAYOFYEAR",

    EXTRACT(ISODAYOFWEEK FROM dt) AS "ISODAYOFWEEK",
    EXTRACT(ISO_WEEK FROM dt) AS "ISO_WEEK",
    EXTRACT(ISOMONTH FROM dt) AS "ISOMONTH",
    EXTRACT(ISOYEAR FROM dt) AS "ISOYEAR",
    EXTRACT(ISO_WEEK_OF_MONTH FROM dt) AS "ISO_WEEK_OF_MONTH",
    EXTRACT(ISODAY_OF_MONTH FROM dt) AS "ISODAY_OF_MONTH",
    EXTRACT(ISODAY_OF_YEAR FROM dt) AS "ISODAY_OF_YEAR"
FROM
    #dates;
```

Conformance

SQL:2003 standard - Feature F052 "Intervals and datetime arithmetic"

17.6.5 Boolean Literals

Boolean Literals

Syntax:

```
<boolean literal> ::=  
| TRUE  
| FALSE  
UNKNOWN
```

Notes

- The boolean literal UNKNOWN is equivalent to the null value.
-

Conformance

SQL:2003 standard - Feature T031 "BOOLEAN data type"

17.7 Value Expressions

Value Expressions

Specify a scalar value.

Syntax

```
<value expression> ::=  
  | <common value expression>  
  | <boolean value expression>  
  | <row value expression>  
  
<common value expression> ::=  
  | <numeric value expression>  
  | <string value expression>  
  | <datetime value expression>  
  
<value expression primary> ::=  
  | ( <value expression> )  
  | <nonparenthesized value expression primary>  
  
<nonparenthesized value expression primary> ::=  
  | <unsigned value specification>  
  | <general value specification>  
  | column-reference  
  | <aggregate function>  
  | <scalar subquery>  
  | <case expression>  
  | <cast specification>  
  | <routine invocation>  
  
<value specification> ::=  
  | <literal>  
  | <general value specification>  
  
<unsigned value specification> ::=  
  | <unsigned literal>  
  | <general value specification>  
  
<general value specification> ::=  
  | SQL-parameter-reference  
  | SQL-variable-reference  
  | <dynamic parameter specification>
```

```

| USER
| CURRENT_USER
| SESSION_USER
| SYSTEM_ROW#
| NEWGUID
ROWSREAD
ROWSAFFECTED

<target specification> ::=

| SQL-parameter-reference
| SQL-variable-reference
| column-reference
<dynamic parameter specification>

```

Usage

Value expressions return values with a data type declared by the expression, and can be used in SQL everywhere a value is expected and the particular data type is permitted. Value expressions can be as simple as a literal or a column reference, or complex expressions involving arithmetic operations, combined boolean logic, scalar subqueries, case expressions and SQL functions.

Conformance

SQL:2003 standard - Core SQL

- Feature F052 "Intervals and datetime arithmetic"
 - Feature T031 "BOOLEAN data type"
 - Feature F471 "Scalar subquery values"
-

Numeric Value Expressions
String Value Expressions
DateTime Value Expressions
Boolean Value Expressions
Row Value Expressions
Subqueries
Case Expressions
Cast Specification

17.7.1 Numeric Value Expressions

Numeric Value Expressions

Specify a numeric value.

Syntax

```
<numeric value expression> ::=  
  | <term>  
  | <numeric value expression> + <term>  
  | <numeric value expression> - <term>  
  
<term> ::=  
  | <factor>  
  | <term> * <factor>  
  | <term> / <factor>  
  
<factor> ::= [ { + | - } ] <numeric primary>  
  
<numeric primary> ::=  
  | <value expression primary>  
  | <numeric value function>
```

Notes

- The declared type of a numeric primary shall be a numeric type.
- If both operands are exact numeric types, then the data type of the result is an exact numeric type with precision not less than the maximum precision of the two operands and the scale determined as follows:
 - If addition or subtraction is specified, then the scale is the maximum of the two operands.
 - If multiplication is specified, then the scale is the sum of the two operand scales, limited by the maximum scale of 4.
 - If division is specified, then the scale is the maximum scale 4.
- If either of the operands are an approximate numeric type, then the result is an approximate numeric type with precision not less than the maximum precision of the two operands.
- Normal operator precedence is applied.

See also: Numeric Value Functions

Examples

- 1) The following example raises all teacher salaries by a given percent:

```
UPDATE teachers  
SET salary = salary * ( 1 + ( :percent / 100 ) )
```

- 2) Some examples of valid numeric value expressions:

- -5
 - 3 + 1
 - 7/3-16
 - 15 * (5-4)
-

Conformance

SQL:2003 standard - Core SQL

17.7.2 String Value Expressions

String Value Expressions

Specify a character string value or a binary string value.

Syntax

```
<string value expression> ::=  
| <character value expression>  
<blob value expression>  
  
<character value expression> ::=  
| <character value expression> { || | + } <character primary>  
<character primary>  
  
<character primary> ::=  
| <value expression primary>
```

```
<string value function>

<blob value expression> ::=  
    <blob value expression> || <blob primary>

<blob primary> ::=  
    | <value expression primary>  
    <string value function>
```

Notes

- The declared type of a character primary shall be a character string type.
- The declared type of a blob primary shall be a binary string type.
- Two character strings can only be concatenated if they share the same character set and are comparable.
- The concatenation operator and the arithmetic plus operator can be used interchangeably to concatenate string values. If character string concatenation is specified, then the length of the result is the combined length of the two values, up to the maximum length for the data type.
- If either of the operands are a character large object type, then the result type of the concatenation is CHARACTER LARGE OBJECT or NATIONAL CHARACTER LARGE OBJECT.
- If either of the operands are a variable-length character string, then the result type of the concatenation is CHARACTER VARYING or NATIONAL CHARACTER VARYING.
- If both operands are fixed-length character strings, then the result type of the concatenation is CHARACTER or NATIONAL CHARACTER.
- If either of the operands are a national character string type, then the result of the concatenation is a national character string type.
- If either of the operands are null, then the result of the concatenation is null.

See also: String Value Functions

Examples

1) String concatenation using the concatenation operator:

```
UPDATE students  
SET studentName = firstName || ' ' || lastName
```

2) String concatenation using the plus operator:

```
UPDATE students  
SET studentName = firstName + ' ' + lastName
```

Conformance

SQL:2003 standard - Core SQL

NexusDB - Plus sign as string concatenation operator
extensions

17.7.3 DateTime Value Expressions

DateTime Value Expressions

Specify a datetime value.

Syntax

```
<datetime value expression> ::=  
| <datetime primary>  
| <datetime value expression> + <interval literal>  
| <datetime value expression> - <interval literal>  
  
<datetime primary> ::=  
| <value expression primary>  
| <datetime value function>
```

Notes

- The declared type of a datetime primary shall be a datetime type.

See also: DateTime Value Functions

Examples

Interval Literals Examples

Conformance

SQL:2003 standard - Feature F052 "Intervals and datetime arithmetic"

17.7.4 Boolean Value Expressions

Boolean Value Expressions

Specify a boolean value.

Syntax

```
<boolean value expression> ::=  
  | <boolean term>  
    <boolean value expression> OR <boolean term>  
  
<boolean term> ::=  
  | <boolean factor>  
    <boolean term> AND <boolean factor>  
  
<boolean factor> ::= [ NOT ] <boolean test>  
  
<boolean test> ::= <boolean primary> [ IS [ NOT ] <truth value> ]  
  
<truth value> ::=  
  | TRUE  
  | FALSE  
  UNKNOWN  
  
<boolean primary> ::=  
  | <predicate>  
    <boolean predicand>  
  
<boolean predicand> ::=  
  | ( <boolean value expression> )  
    <nonparenthesized value expression primary>
```

Notes

- The declared type of a boolean primary shall be boolean.

- The AND and OR boolean operators are used to combine two or more predicates.
- The NOT operator inverts the meaning of a boolean expression.
- The boolean test:

<boolean primary> IS NOT <truth value>

is equivalent to:

NOT <boolean primary> IS <truth value>

Truth tables

Since SQL supports null values and predicates whose value can be UNKNOWN, a three-valued logic is used to evaluate boolean value expressions. The rules of this logic are summarized in the following tables:

Truth Table for the AND Boolean Operator

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

Truth Table for the OR Boolean Operator

OR	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

Truth Table for the NOT Boolean Operator

NOT	
True	False

<i>False</i>	True
<i>Unknown</i>	Unknown

Truth Table for the IS Boolean Operator

<i>IS</i>	<i>True</i>	<i>False</i>	<i>Unknown</i>
<i>True</i>	True	False	False
<i>False</i>	False	True	False
<i>Unknown</i>	False	False	True

Examples

- 1) The following example selects students from Newport, CA:

```
SELECT studentID, studentName, city, state
FROM students
WHERE city = 'Newport' AND state = 'CA'
```

- 2) The following example selects students from Newport, CA and RI:

```
SELECT studentID, studentName, city, state
FROM students
WHERE city = 'Newport' AND ( state = 'CA' OR
state = 'RI' )
```

- 3) The following example selects students from Massachusetts and Newport, CA:

```
SELECT studentID, studentName, city, state
FROM students
WHERE ( city = 'Newport' AND state = 'CA' ) OR
state = 'MA'
```

- 4) The following example selects all students except those from California and Boston, MA:

```
SELECT studentID, studentName, city, state
FROM students
WHERE NOT ( ( city = 'Boston' AND state = 'MA' )
OR state = 'CA' )
```

Conformance

SQL:2003 standard - Feature T031 "BOOLEAN data type"
- Feature F571 "Truth value tests"

17.7.5 Row Value Expressions

Row Value Expressions

Specify a value or list of values to construct a row value.

Syntax

```
<row value expression> ::= <row value constructor>

<row value constructor> ::=
| <common value expression>
| <boolean value expression>
| <explicit row value constructor>

<explicit row value constructor> ::=
( <row value element> , <row value element list> )

<row value element list> ::=
<row value element> [ { , <row value element> }... ]

<row value element> ::=
| <value expression>
| NULL
| DEFAULT

<row value predicand> ::= <row value constructor predicand>

<row value constructor predicand> ::=
| <common value expression>
| <boolean predicand>
| <explicit row value constructor>
```

Notes

- Row value constructors with more than 1 row value element are only supported in the INSERT statement in NexusDB SQL.

Examples

1) The following example selects students from California and New York:

```
SELECT studentName, state  
FROM students  
WHERE state IN ( 'CA', 'NY' )
```

2) The following example inserts two new rows into the documents table:

```
INSERT into documents ( documentID, title )  
VALUES ( NEWGUID, 'NexusDB main features' ),  
( NEWGUID, 'NexusDB special features' )
```

Conformance

SQL:2003 standard - Core SQL

17.7.6 Subqueries

Subqueries

Specify a scalar value, a row or a table based on a query expression.

Syntax

```
<scalar subquery> ::= <subquery>  
<row subquery> ::= <subquery>  
<table subquery> ::= <subquery>  
<subquery> ::= ( <query expression> )
```

Usage

A subquery is a query that is executed as part of another query expression.

Notes

- Subqueries shall be enclosed in parentheses.

- Table subqueries are commonly used in predicates, but can also appear in the FROM clause to specify a derived table.
- Scalar subqueries can be used anywhere a value expression is allowed, including computed columns in the select list. A scalar subquery shall have a degree of one and a cardinality of one.
- A correlated subquery is a subquery that is referencing data in an outer query. A subquery specified in the select list is typically correlated with the main query to produce meaningful data. Since correlated subqueries depend on data from the outer query, they must normally be evaluated for each row produced by the outer query.

Examples

- 1) The following examples uses a subquery with the IN predicate to select students who are enrolled in courseID 730:

```
SELECT studentID, studentName  
FROM students  
WHERE studentID IN ( SELECT studentID FROM  
enrolls WHERE courseID = 730 )
```

- 2) The following example uses a subquery in the FROM clause instead of a base table:

```
SELECT *  
FROM ( SELECT studentID, studentName, gender  
FROM students ) AS student_list
```

- 3) The following example uses a correlated subquery in the select list to count the number of courses related to the selected student:

```
SELECT  
  
courseID,  
studentID,  
( SELECT COUNT( * ) FROM enrolls WHERE  
studentID = e.studentID ) AS numCourses  
  
FROM enrolls e  
ORDER BY courseID
```

Conformance

SQL:2003 standard - Core SQL

17.7.7 Case Expressions

Case Expressions

Specify a conditional value.

Syntax

```
<case expression> ::=  
| <case abbreviation>  
  <case specification>  
  
<case abbreviation> ::=  
| NULLIF ( <value expression> , <value expression> )  
  COALESCE ( <value expression> { , <value expression> }... )  
  
<case specification> ::=  
| CASE <case operand> <simple when clause>... [ <else clause> ] END  
  CASE <searched when clause>... [ <else clause> ] END  
  
<simple when clause> ::= WHEN <when operand> THEN <result>  
  
<searched when clause> ::= WHEN <search condition> THEN <result>  
  
<else clause> ::= ELSE <result>  
  
<case operand> ::= <row value predicand>  
  
<when operand> ::= <row value predicand>  
  
<result> ::=  
| <value expression>  
  NULL
```

Usage

The CASE expression is similar in concept to the CASE statement found in some programming languages. It is used in SQL to provide a **conditional** value, and can be specified anywhere a value expression is allowed.

Notes

- The searched CASE expression is the main syntactical version, and the most flexible one, since a search condition is specified for each individual WHEN clause.
- The simple CASE expression is a handy shorthand that eliminates the need for repeating the same value in each WHEN clause when comparing the same value against different conditions.
- The NULLIF abbreviation compares two values and returns null if the values are equal, else returning the first value specified. This is shorthand syntax for:

```
CASE WHEN value1 = value2 THEN NULL ELSE
      value1 END
```

- The COALESCE abbreviation returns the first value in a list of comparable values that is not null, or null if all values in the list are the null value. This is shorthand syntax for:

```
CASE
    WHEN value1 IS NOT NULL THEN value1
    WHEN value2 IS NOT NULL THEN value2
    WHEN valuen IS NOT NULL THEN valuen
    ELSE NULL
END
```

- If the optional ELSE clause is omitted, then ELSE NULL is implicit.
- The resulting data type of a CASE expression is determined according to the rules for Result data types of aggregations.

Tip: Use COALESCE in arithmetic expressions to avoid calculations on a null value.

Examples

- 1) The following example uses a searched CASE expression to determine a conditional value for the gender_description column:

```
SELECT
    studentID,
```

```
studentName,  
CASE  
  
WHEN gender = 'F' THEN 'Female'  
WHEN gender = 'M' THEN 'Male'  
ELSE 'Not available'  
  
END AS gender_description  
  
FROM students
```

- 2) The following example uses a simple CASE expression to determine a conditional value for the gender_description column:

```
SELECT  
  
studentID,  
studentName,  
CASE gender  
  
WHEN 'F' THEN 'Female'  
WHEN 'M' THEN 'Male'  
ELSE 'Not available'  
  
END AS gender_description  
  
FROM students
```

- 3) The following example uses the NULLIF function to display a null in the rows where the orderTotal column has a value of 0:

```
SELECT  
  
orderID,  
NULLIF( orderTotal, 0 ) AS orderTotal  
  
FROM orders  
ORDER BY orderID
```

- 4) The following example uses the COALESCE function to avoid calculations on potential null values:

```
SELECT  
  
orderID,
```

```
COALESCE( orderTotal, 0 ) -  
COALESCE( amountPaid, 0 ) AS "Amount Due"
```

```
FROM orders  
ORDER BY orderId
```

Conformance

SQL:2003 standard - Core SQL

17.7.8 Cast Specification

Cast Specification

Specify a data conversion.

Syntax

```
<cast specification> ::= CAST ( <cast operand> AS <data type> )  
  
<cast operand> ::=  
| <value expression>  
NULL
```

Usage

The CAST function converts data of one type to a different type, subject to the conversion rules below.

Notes

- Numeric values can be converted to any other numeric data type. If the target type has a smaller scale than the source value, then the fractional component is truncated. An exception is raised if the conversion would lead to a loss of any leading significant digits.
- Numeric values can be converted to any character string type.
- Character strings can be converted to other character string types. If the string being converted has more characters than the maximum length of the target type, then the string is truncated to fit the length of the target type.

- Character strings can be converted to any other data type, with the restriction that the contents of the character string must make sense for the target type. For example, if the target type is DATE, then the character string must mimic the pattern of a DATE literal – four digits, a hyphen, two digits, another hyphen, and two digits.
- A DATE value can be converted to a character string or a TIMESTAMP with the time part filled in with 00:00:00.
- A TIME value can be converted to a character string or a TIMESTAMP with the date part filled in with the current date.
- A TIMESTAMP value can be converted to a character string, a DATE or a TIME.
- Values of any datetime type can be converted to a numeric type. This functionality is a NexusDB extension.
- Boolean values can be converted to a character string. The boolean value TRUE is converted to 'True' and FALSE is converted to 'False'.
- The expression CAST (NULL AS data-type) is used to associate a data type with NULL.

Examples

1) The following example casts a character string value to DOUBLE PRECISION:

```
CAST( '2.75' AS DOUBLE PRECISION )
```

2) The following example casts an integer value to FLOAT:

```
CAST( 50 AS FLOAT )
```

3) The following example casts a character string value to DATE:

```
CAST( '2004-10-15' AS DATE )
```

4) The following example casts the current datetime value to VARCHAR:

```
CAST( CURRENT_TIMESTAMP AS  
VARCHAR(19) )
```

- 5) The following example casts the null value to DECIMAL:

```
CAST ( NULL AS DECIMAL )
```

Conformance

SQL:2003 standard - Core SQL

NexusDB - Casting datetime values to a numeric type
extensions

17.8 Parameters

Parameters

Specify a dynamic value.

Syntax

```
<dynamic parameter specification> ::= :parameter-name | ?
```

Usage

Parameters are used to specify dynamic values that are passed to the statement by the client application.

Notes

- The parameter-name can be any name you choose, but a common (and good) practice is to use descriptive names.

- The notation of using a question mark to represent parameters is only used with ODBC and ADO.NET data access.

Tip: Preparing statements with parameters will improve the efficiency when the same statement is executed more than once.

Examples

SELECT statement with parameters in the search condition
INSERT statement with parameters in the VALUES clause
UPDATE statement with parameters in the SET clause
DELETE statement with parameters in the WHERE clause

Conformance

SQL:2003 standard - Core SQL

NexusDB - Named dynamic parameters
extensions

18 SQL Data Types

SQL Data Types

Specify a data type.

Syntax

```
<data type> ::=  
| <character string type> [ <character set clause> ] [ <collate clause> ]  
| <national character string type> [ <collate clause> ]  
| <binary large object string type>  
| <numeric type>  
| <boolean type>  
| <datetime type>  
| GUID  
| BYTEARRAY ( length )  
RECREV  
  
<character set clause> ::=  
| CHARACTER SET <character string literal>  
CODEPAGE <unsigned integer>  
  
<collate clause> ::=  
| COLLATE <character string literal> [ <compare flag>... ]  
LOCALE <unsigned integer> [ <compare flag>... ]  
  
<compare flag> ::=  
| IGNORE KANA TYPE  
| IGNORE NONSPACE  
| IGNORE SYMBOLS  
| IGNORE WIDTH  
USE STRING SORT
```

Character sets and collations

NexusDB uses the available Windows OS codepages and locales to define the encoding and collation for the character string types.

- CHARACTER SET and CODEPAGE are used interchangeably to specify the encoding used by the character string type.
- The national character string type implicitly defines codepage UTF16.
- If a character set is not specified, then the default codepage associated with the specified collation is implicit.

- If neither an explicit nor implicit character set is defined, then no encoding takes place.

Tip: You can query the #CODEPAGES system table to get a list of supported character sets.

- COLLATE and LOCALE are used interchangeably to specify an explicit collation for the character string and national character string types.
- If a collation is not specified for the character string type, then the collation is byte order.
- If a collation is not specified for the national character string type, then the collation is locale-neutral.

Tip: You can query the #COLLATIONS system table to get a list of supported collations.

Note: The compare flags are arguments used in standard Windows API calls. Please check your Windows OS documentation for the meaning of these flags.

Result data types of aggregations

The result data type of an aggregation over values of compatible data types, such as CASE expressions and columns in the result of a query expression with UNION, EXCEPT and INTERSECT, is determined as follows:

- All data types in the set of data types shall be comparable.
- If any of the data types in the set are character string, then all data types shall be a character string type. The result data type is the character string type with highest precedence:
 - If at least one of the data types in the set is a character large object type, then the result data type is a character large object type. NATIONAL CHARACTER LARGE OBJECT takes precedence over CHARACTER LARGE OBJECT.
 - If at least one of the data types in the set is a variable-length character string, then the result data type is a variable-length character string with maximum length equal to the maximum length of the data types in the set. NATIONAL CHARACTER VARYING takes precedence over CHARACTER VARYING.

- Otherwise the result is a fixed-length character string with length equal to the maximum length of the data types in the set. NATIONAL CHARACTER takes precedence over CHARACTER.
- If any of the data types in the set are binary string, then all data types shall be binary string. The result data type is BINARY LARGE OBJECT.
- If all data types in the set are exact numeric, then the result data type is an exact numeric type with precision not less than the highest precision of the data types in the set and scale equal to the maximum scale of the data types in the set.
- If any of the data types in the set are approximate numeric, then the result data type is an approximate numeric type with precision not less than the highest precision of the data types in the set.
- If any of the data types in the set are a datetime type, then all data types in the set shall be a datetime type having the same datetime fields, which is also the data type of the result.
- If any of the data types in the set are boolean, then all data types shall be boolean. The result data type is BOOLEAN.

Special data types

- GUID is a NexusDB-specific 16 byte binary string type used to define a Globally Unique Identifier.
The NEWGUID function can be used to generate GUID values.
 - BYTEARRAY is a NexusDB-specific binary string type with a maximum length of 64KB.
 - RECREV is a special NexusDB-specific 32-bit unsigned integer type which is incremented automatically each time the row is updated.
-

Conformance

SQL:2003 standard - Core SQL
- Feature F421 "National character"

	<ul style="list-style-type: none"> - Feature T041 "Basic LOB data type support" - Feature T031 "BOOLEAN data type"
NexusDB extensions	<ul style="list-style-type: none"> - GUID - BYTEARRAY - RECREV - CODEPAGE - LOCALE - Compare flags in the collate clause

Overview of Predefined Data Types
 String Types
 Numeric Types
 Boolean Types
 DateTime Types

18.1 Overview of Predefined Data Types

Overview of Predefined Data Types

Predefined data types in NexusDB SQL

SQL type	NexusDB type	Capacity	Conformance
CHARACTER [(length)]	nxtNullString	8192 characters	Core SQL
CHAR [(length)]	nxtNullString	8192 characters	Core SQL
NULLSTRING [(length)]	nxtNullString	8192 characters	NexusDB ext
SHORTSTRING [(length)]	nxtShortString	255 characters	NexusDB ext
SINGLECHAR	nxtChar	8-bit character	NexusDB ext
CHARACTER VARYING [(length)]	nxtNullString	8192 characters	Core SQL
CHAR VARYING [(length)]	nxtNullString	8192 characters	Core SQL
VARCHAR [(length)]	nxtNullString	8192 characters	Core SQL
CHARACTER LARGE OBJECT [(length)]	nxtBlobMemo	4GB	Feature T041
CHAR LARGE OBJECT [(length)]	nxtBlobMemo	4GB	Feature T041
CLOB [(length)]	nxtBlobMemo	4GB	Feature T041
TEXT [(length)]	nxtBlobMemo	4GB	NexusDB ext
NATIONAL CHARACTER [(length)]	nxtWideString	32767 characters	Feature F421
NATIONAL CHAR [(length)]	nxtWideString	32767 characters	Feature F421

NCHAR [(length)]	nxtWideString	32767 characters	Feature F421
NSINGLECHAR	nxtWideChar	16-bit character	NexusDB ext
NATIONAL CHARACTER VARYING [(length)]	nxtWideString	32767 characters	Feature F421
NATIONAL CHAR VARYING [(length)]	nxtWideString	32767 characters	Feature F421
NCHAR VARYING [(length)]	nxtWideString	32767 characters	Feature F421
NVARCHAR [(length)]	nxtWideString	32767 characters	NexusDB ext
NATIONAL CHARACTER LARGE OBJECT [(length)]	nxtBlobWideMemo	4GB	Feature T041
NCHAR LARGE OBJECT [(length)]	nxtBlobWideMemo	4GB	Feature T041
NCLOB [(length)]	nxtBlobWideMemo	4GB	Feature T041
BINARY LARGE OBJECT [(length)]	nxtBlob	4GB	Feature T041
BLOB [(length)]	nxtBlob	4GB	Feature T041
IMAGE [(length)]	nxtBlobGraphic	4GB	NexusDB ext
NUMERIC [(precision [, scale])]	nxtBCD	Precision 20, scale 4	Core SQL
DECIMAL [(precision [, scale])]	nxtBCD	Precision 20, scale 4	Core SQL
DEC [(precision [, scale])]	nxtBCD	Precision 20, scale 4	Core SQL
BCD [(precision [, scale])]	nxtBCD	Precision 20, scale 4	
FMTBCD [(precision [, scale])]	nxtFmtBCD	Precision 20, scale 4	
BYTE	nxtByte	8-bit unsigned integer	NexusDB ext
TINYINT	nxtByte	8-bit unsigned integer	NexusDB ext
SHORTINT	nxtInt8	8-bit signed integer	NexusDB ext
SMALLINT	nxtInt16	16-bit signed integer	Core SQL
INTEGER	nxtInt32	32-bit signed integer	Core SQL
INT	nxtInt32	32-bit signed integer	Core SQL

AUTOINC [(start-value [, increment-value])]	nxtAutoInc	32-bit unsigned integer	NexusDB ext
BIGINT	nxtInt64	64-bit signed integer	Feature T071
LARGEINT	nxtInt64	64-bit signed integer	NexusDB ext
WORD	nxtWord16	16-bit unsigned integer	NexusDB ext
DWORD	nxtWord32	32-bit unsigned integer	NexusDB ext
FLOAT [(precision)]	nxtSingle	1.5E-45 .. 3.4E38	Core SQL
REAL	nxtDouble	5.0E-324 .. 1.7E308	Core SQL
DOUBLE PRECISION	nxtDouble	5.0E-324 .. 1.7E308	Core SQL
EXTENDED	nxtExtended	3.6E-4951 .. 1.1E4932	NexusDB ext
MONEY	nxtCurrency	Precision 20, scale 4	NexusDB ext
BOOLEAN	nxtBoolean	8-bit boolean flag	Feature T031
BOOL	nxtBoolean	8-bit boolean flag	NexusDB ext
DATE	nxtDate	4 bytes	Core SQL
TIME	nxtTime	4 bytes	Core SQL
TIMESTAMP	nxtDateTime	8 bytes	Core SQL
DATETIME	nxtDateTime	8 bytes	NexusDB ext
GUID	nxtGuid	16 bytes	NexusDB ext
BYTARRAY (length)	nxtByteArray	64KB	NexusDB ext
RECREV	nxtRecRev	32-bit unsigned integer	NexusDB ext

18.2 String Types

String Types

Syntax

<character string type> ::=

| CHARACTER [(length)]
| CHAR [(length)]
| NULLSTRING [(length)]
| SHORTSTRING [(length)]
| SINGLECHAR
| CHARACTER VARYING [(length)]
| CHAR VARYING [(length)]
| VARCHAR [(length)]
<character large object type>

<character large object type> ::=
| CHARACTER LARGE OBJECT [(length)]
| CHAR LARGE OBJECT [(length)]
| CLOB [(length)]
TEXT [(length)]

<national character string type> ::=
| NATIONAL CHARACTER [(length)]
| NATIONAL CHAR [(length)]
| NCHAR [(length)]
| NSINGLECHAR
| NATIONAL CHARACTER VARYING [(length)]
| NATIONAL CHAR VARYING [(length)]
| NCHAR VARYING [(length)]
| NVARCHAR [(length)]
<national character large object type>

<national character large object type> ::=
| NATIONAL CHARACTER LARGE OBJECT [(length)]
| NCHAR LARGE OBJECT [(length)]
NCLOB [(length)]

<binary large object string type> ::=
| BINARY LARGE OBJECT [(length)]
| BLOB [(length)]
IMAGE [(length)]

Notes

- NexusDB stores string data either in fixed or variable length, depending on the STORAGE attribute in the table definition.
- If the length argument is omitted, then a length of 1 is implicit.
- The large object types have a dynamic length in NexusDB, up to the maximum capacity. Specifying the length argument has no effect, and is only included for syntax compatibility with SQL:2003.

- The character string type defines a single-byte ANSI string.
 - CHARACTER is a fixed-length character string type with a maximum length of 8192 characters. Values are padded with trailing spaces on data retrieval to fill the fixed length.
 - CHAR is equivalent to CHARACTER.
 - CHARACTER VARYING is a variable-length character string type with the same maximum length as CHARACTER.
 - CHAR VARYING and VARCHAR are equivalent to CHARACTER VARYING.
 - NULLSTRING is a NexusDB synonym for CHARACTER VARYING.
 - SHORTSTRING is a NexusDB-specific variable-length character string type with a maximum length of 255 characters.
 - SINGLECHAR is a NexusDB-specific type that defines a single 8-bit character.
 - CHARACTER LARGE OBJECT is a large object character string type with a maximum capacity of 4GB.
 - CHAR LARGE OBJECT and CLOB are equivalent to CHARACTER LARGE OBJECT.
 - TEXT is a NexusDB synonym for CHARACTER LARGE OBJECT.
- The national character string type defines a double-byte Unicode string with implicit character set UTF16.

- NATIONAL CHARACTER is a fixed-length character string type with a maximum capacity of 32767 characters. Values are padded with trailing spaces on data retrieval to fill the fixed length.
 - NATIONAL CHAR and NCHAR are equivalent to NATIONAL CHARACTER.
 - NSINGLECHAR is a NexusDB-specific type that defines a single 16-bit character.
 - NATIONAL CHARACTER VARYING is a variable-length character string type with the same maximum length as NATIONAL CHARACTER.
 - NATIONAL CHAR VARYING and NCHAR VARYING are equivalent to NATIONAL CHARACTER VARYING.
 - NVARCHAR is a NexusDB synonym for NATIONAL CHARACTER VARYING.
 - NATIONAL CHARACTER LARGE OBJECT is a large object character string type with a maximum capacity of 4GB.
 - NCHAR LARGE OBJECT and NCLOB are equivalent to NATIONAL CHARACTER LARGE OBJECT.
-
- BINARY LARGE OBJECT is a large object binary string type with a maximum capacity of 4GB.
 - BLOB is equivalent to BINARY LARGE OBJECT.
 - IMAGE is a NexusDB-specific large object binary string type with a maximum capacity of 4GB.
 - Two character strings are comparable only if they use the same collation and compare flags.
 - A character string is assignable to any character string site, including sites having a different character set, provided that data loss does not occur during the translation. An exception is

raised if an assignment leads to truncation of the character string data.

- A binary string is only assignable to a binary string type.

Conformance

SQL:2003 standard - Core SQL

- Feature F421 "National character"
- Feature T041 "Basic LOB data type support"

NexusDB extensions	<ul style="list-style-type: none"> - NULLSTRING - SHORTSTRING - SINGLECHAR - TEXT - NSINGLECHAR - NVARCHAR - IMAGE
-----------------------	---

18.3 Numeric Types

Numeric Types

Syntax

```

<numeric type> ::= 
  | <exact numeric type>
  | <approximate numeric type>

<exact numeric type> ::= 
  | NUMERIC [ ( precision [ , scale ] ) ]
  | DECIMAL [ ( precision [ , scale ] ) ]
  | DEC [ ( precision [ , scale ] ) ]
  | BYTE
  | TINYINT
  | SHORTINT
  | SMALLINT
  | INTEGER
  | INT
  | AUTOINC [ ( start-value [ , increment-value ] ) ]
  | BIGINT
  | LARGEINT
  | WORD
  | DWORD

```

```
<approximate numeric type> ::=  
| FLOAT [ ( precision ) ]  
| REAL  
| DOUBLE PRECISION  
| EXTENDED  
| MONEY  
| BCD  
| FMTBCD
```

Notes

- The exact numeric types can represent a numeric value exactly. The unsigned and signed integer types have an implicit scale of 0, while the NUMERIC and DECIMAL types have both a precision (the number of significant digits) and a scale (the fractional component).
 - DECIMAL is a BCD type with a maximum precision of 16-20, depending on the scale, and a maximum scale of 4. A scale of 0 is implicit if the scale argument is omitted, and a precision of 16 is implicit if also the precision argument is omitted.
 - NUMERIC is equivalent to DECIMAL.
- BYTE and TINYINT are equivalent NexusDB-specific 8-bit unsigned integer types with range 0 .. 255.
- SHORTINT is a NexusDB-specific 8-bit signed integer type with range -128..127.
- SMALLINT is a 16-bit signed integer type with range -32768..32767.
- INTEGER is a 32-bit signed integer type with range -2147483648..2147483647.
- INT is equivalent to INTEGER.
- AUTOINC is a NexusDB-specific 32-bit unsigned integer type with range 0..4294967295, suitable for identity columns. A start value of 1 is implicit if start-value is not specified. An increment value

of 1 is implicit if increment-value is not specified.

- BIGINT is a 64-bit integer type with range -2E63..2E63-1.
 - LARGEINT is a NexusDB synonym for BIGINT.
 - WORD is a NexusDB-specific unsigned 16-bit integer type with range 0..65535.
 - DWORD is a NexusDB-specific unsigned 32-bit integer type with range 0..4294967295
 - The approximate numeric types are floating point values that can only represent numbers approximately.
 - FLOAT is a 4 byte floating point type with range -1.5E-45..3.4E38. The option of specifying the precision is included for syntax compatibility with SQL:2003, but does not have any affect on the actual precision.
 - DOUBLE PRECISION is an 8 byte floating point type with range 5.0E-324..1.7E308.
 - REAL is equivalent to DOUBLE PRECISION.
 - EXTENDED is a NexusDB-specific 10 byte floating point type with range 3.6E-4951..1.1E4932.
 - MONEY is a NexusDB-specific type that is stored as a 64-bit scaled integer with the last 4 digits representing the fractional component.
- Any two numbers are comparable.
 - A number is assignable only to numeric sites. An exception is raised if an assignment of a numeric value would lead to a loss of its most significant digit. When a numeric value is assigned

to a site of exact numeric type, the value is converted to have the precision and scale of the target, with truncation of the fractional component if the scale is reduced.

Conformance

SQL:2003 standard - Core SQL

- Feature T071 "BIGINT data type"

NexusDB
extensions

- BYTE
- TINYINT
- SHORTINT
- AUTOINC
- LARGEINT
- WORD
- DWORD
- EXTENDED
- MONEY
- BCD
- FMTBCD

18.4 Boolean Types

Boolean Types

Syntax

```
<boolean type> :=  
| BOOLEAN  
| BOOL
```

Notes

- The BOOLEAN data type can have one of three different boolean literal values:

TRUE
FALSE
UNKNOWN

- BOOL is a NexusDB synonym for BOOLEAN.
- Two boolean values are comparable.

-
- A boolean value is assignable only to a boolean site.
-

Conformance

SQL:2003 standard - Feature T031 "BOOLEAN data type"

NexusDB - BOOL
extensions

18.5 DateTime Types

DateTime Types

Syntax

```
<datetime type> ::=  
| DATE  
| TIME  
| TIMESTAMP  
DATETIME
```

Notes

- DATE is a 4 byte type that contains the datetime fields YEAR, MONTH and DAY.
- TIME is a 4 byte type that contains the datetime fields HOUR, MINUTE and SECOND.
- TIMESTAMP is a 8 byte type that contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE and SECOND.
- DATETIME is a NexusDB synonym for TIMESTAMP.
- The time precision of the datetime types is milliseconds, 3 digits following the decimal point in the SECOND datetime field.
- Two date time values are comparable only if both have the same datetime fields.

- A datetime value is assignable only to a datetime site having the same datetime fields.
-

Conformance

SQL:2003 standard - Core SQL

NexusDB - DATETIME
extensions

19 Functions

Functions

This section describes the aggregate functions and built-in SQL functions available in NexusDB SQL.

Aggregate Functions
 Numeric Value Functions
 String Value Functions
 DateTime Value Functions
 System Functions

19.1 Aggregate Functions

Aggregate Functions

Specify a value computed from a group of rows.

Syntax

```
<aggregate function> ::=  

    | COUNT( * )  

    <general set function>  
  

<general set function> ::=  

    <set function type> ( [ DISTINCT | ALL ] <value expression> )  
  

<set function type> ::=  

    | AVG  

    | MAX  

    | MIN  

    | SUM  

    | COUNT  

    | MED  

    | STD  

    | LIST
```

Usage

The aggregate functions, also referred to as set functions, perform operations that aggregate data from groups of rows.

Notes

- ALL is implicit if no set quantifier is not specified.
- If DISTINCT is specified, then all duplicate values are removed from the set.
- If the query specifies a GROUP BY clause, then the aggregate functions are applied to each group, else the entire table is treated as a single group.
- If a general set function is specified, other than the COUNT function, and the set is empty, then the result is null.

COUNT function

- If COUNT(*) is specified, then the result is the cardinality of the set, else the COUNT function returns the number of non-null values in the set.
- The data type of the result is BIGINT.

MAX function

- The MAX function returns the highest non-null value in the set.
- The data type of the result is the same type as the argument.

MIN function

- The MIN function returns the lowest non-null value in the set.
- The data type of the result is the same type as the argument.

AVG function

- The AVG function calculates the average of non-null values in the set.
- The argument shall be a numeric type.

- The data type of the result is the same type as the argument.

SUM function

- The SUM function calculates the sum of non-null values in the set.
- The argument shall be a numeric type.
- If the argument is an integer type, then the data type of the result is BIGINT.
- If the argument is a BCD type, then the data type of the result is DECIMAL with maximum possible precision and the same scale as the argument.
- If the argument is an approximate numeric type, then the data type of the result is DOUBLE PRECISION or EXTENDED.

MED function

- The MED function calculates the median value of the non-null values in the set.
- The argument shall be a numeric type.
- The data type of the result is the same type as the argument.

STD function

- The STD function calculates the standard deviation of the non-null values in the set.
- The argument shall be a numeric type.
- The data type of the result is the same type as the argument.

LIST function

- LIST is a special NexusDB set function that compiles a comma-separated list of non-null values in the set.
- The argument shall be a character string type.
- The data type of the result is CHAR(4096).

Examples

1) The following example calculates the number of rows in the students table:

```
SELECT COUNT( * ) FROM students
```

2) The following example returns the number of female students:

```
SELECT COUNT( studentName )  
FROM students  
WHERE gender = 'F'
```

3) The following example calculates the average grade for each student:

```
SELECT studentID, AVG( grade )  
FROM enrolls  
GROUP BY studentID
```

4) The following example selects the highest and lowest salaries of teachers:

```
SELECT MAX( salary ), MIN( salary )  
FROM teachers
```

5) The following example calculates the total cost of each order made:

```
SELECT orderID, SUM( extPrice )  
FROM order_details  
GROUP BY orderID
```

6) The following example retrieves a list of student names grouped by gender:

```
SELECT gender, LIST( studentName )
FROM students
GROUP BY gender
```

Conformance

SQL:2003 standard - Core SQL

NexusDB extensions	- MED function - STD function - LIST function
-----------------------	---

19.2 Numeric Value Functions

Numeric Value Functions

Specify a function that returns a numeric value.

Syntax

```
<numeric value function> ::=  
| <position expression>  
| <extract expression>  
| <length expression>  
| <absolute value expression>  
| <modulus expression>  
| <natural logarithm>  
| <exponential function>  
| <power function>  
| <square root>  
| <floor function>  
| <ceiling function>  
| <arctangent function>  
| <cosine function>  
| <sine function>  
| <ordinal function>  
| <pi function>  
| <round function>  
| <random function>  
| <lastautoinc function>
```

Usage

Numeric value functions can be used in SQL everywhere a numeric value is permitted.

Conformance

SQL:2003 standard - Core SQL

- Feature F052 "Intervals and datetime arithmetic"
- Feature T441 "ABS and MOD functions"
- Feature T621 "Enhanced numeric functions"

NexusDB
extensions

- ATAN function
- ATAN2 function
- COS function
- SIN function
- ORD function
- PI function
- ROUND function
- BROUND function
- RAND function
- LASTAUTOINC function

19.2.1 Position Expression

Position Expression

Syntax

```

<position expression> ::=  
| <string position expression>  
<blob position expression>  
  
<string position expression> ::=  
POSITION ( <string value expression> IN <string value expression>  
[ USING <char length units> ] )  
  
<blob position expression> ::=  
POSITION ( <blob value expression> IN <blob value expression> )  
  
<char length units> ::=  
| CHARACTERS  
OCTETS
  
```

Notes

- The POSITION function returns the ordinal position of the first occurrence of the first string expression within the second string expression.

- The USING clause shall only be specified if the IN expression is a UNICODE character string. CHARACTERS, which is implicit if the clause is omitted, returns the ordinal character position, while OCTETS returns the ordinal byte position.
- The POSITION function returns the ordinal byte position when the IN expression is a binary string.
- If the substring is not found within the second string, then 0 is returned.
- The data type of the result is INTEGER.
- If either of the arguments are null, then the result is null.

Examples

1) The following example lists all courses starting with 'C':

```
SELECT *
FROM courses
WHERE POSITION( 'C' IN courseName ) = 1
```

2) In the following example, the POSITION function returns 3 with the CHARACTERS char length unit implicitly specified:

```
SELECT POSITION( 'a' IN N'Grace' )
FROM #dummy
```

3) In the following example, the POSITION function returns 6 with the OCTETS char length unit explicitly specified:

```
SELECT POSITION( 'a' IN N'Grace' USING
OCTETS )
FROM #dummy
```

Conformance

SQL:2003 standard - Core SQL

19.2.2 Extract Expression

Extract Expression

Syntax

```
<extract expression> ::=  
    EXTRACT ( <extract field> FROM <extract source> )  
  
<extract field> ::= <primary datetime field>  
  
<extract source> ::= <datetime value expression>
```

Notes

- The EXTRACT function returns the datetime field specified by <extract field> from the <extract source>.
- If SECOND is specified as extract field, then the data type of the result is DOUBLE PRECISION, else the data type of the result is INTEGER.
- If either of the arguments are null, then the result is null.

Examples

- 1) The following example extracts the YEAR, MONTH and DAY fields from the orderDate column:

```
SELECT  
  
    orderID,  
    EXTRACT( YEAR FROM orderDate ) AS "Year",  
  
    EXTRACT( MONTH from orderDate ) AS "Month",  
    EXTRACT( DAY FROM orderDate ) AS "Day",  
  
    orderTotal  
  
    FROM orders
```

- 2) The following example groups order totals by year:

```

SELECT
    "Year",
    SUM( orderTotal ) AS "Total Amount"

FROM
(
    SELECT
        EXTRACT( YEAR FROM orderDate ) AS "Year",
        orderTotal
    FROM orders
) AS orders
GROUP BY "Year"

```

Conformance

SQL:2003 standard - Feature F052 "Intervals and datetime arithmetic"

19.2.3 Length Expression

Length Expression

Syntax

```

<length expression> ::=

| <char length expression>
  <octet length expression>

<char length expression> ::=
  { CHAR_LENGTH | CHARACTER_LENGTH } ( <string value expression>
  [ USING <char length units> ] )

<octet length expression> ::=
  OCTET_LENGTH ( <string value expression> )

```

Notes

- CHAR_LENGTH and CHARACTER_LENGTH are equivalent functions that return the number of characters or bytes in a character string according to the implicit or explicit char length unit, or

the number of bytes in a binary string.

- The USING clause shall only be specified if the argument is a UNICODE character string. CHARACTERS is implicit if a char length unit is not specified.
- The OCTET_LENGTH function returns the number of octets (bytes) in a character or binary string.
- The data type of the result is INTEGER.
- If the argument is null, then the result is null.

Examples

- 1) The following example selects students who have a longer last name than first name:

```
SELECT firstName, lastName  
FROM students  
WHERE CHAR_LENGTH( lastName ) >  
CHARACTER_LENGTH( firstName )
```

- 2) In the following example, the CHAR_LENGTH function returns 5 with the CHARACTERS char length unit implicitly specified:

```
SELECT CHAR_LENGTH( N'Grace' )  
FROM #dummy
```

- 3) In the following example, the CHAR_LENGTH function returns 10 with the OCTETS char length unit explicitly specified:

```
SELECT CHAR_LENGTH( N'Grace' USING  
OCTETS )  
FROM #dummy
```

Conformance

SQL:2003 standard - Core SQL

19.2.4 Absolute Value Expression

Absolute Value Expression

Syntax

```
<absolute value expression> ::=  
    ABS ( <numeric value expression> )
```

Notes

- The ABS function returns the unsigned value of the argument.
- The data type of the result is the same type as the argument.
- If the argument is null, then the result is null.

Examples

Expression	Result
ABS(5)	5
ABS(-5.25)	5.25

Conformance

SQL:2003 standard - Feature T441 "ABS and MOD functions"

19.2.5 Modulus Expression

Modulus Expression

Syntax

```
<modulus expression> ::=  
    MOD ( <numeric value expression dividend> , <numeric value expression divisor> )
```

<numeric value expression dividend> ::= <numeric value expression>

<numeric value expression divisor> ::= <numeric value expression>

Notes

- The MOD function returns the remainder of the first argument divided by the second argument.
- Both dividend and divisor shall be whole numbers (scale 0).
- The divisor shall not be 0.
- The data type of the result is the same type as the second argument.
- If either of the arguments are null, then the result is null.

Examples

Expression	Result
MOD(7, 2)	1
MOD(11, 4.0)	3.0

Conformance

SQL:2003 standard - Feature T441 "ABS and MOD functions"

19.2.6 Natural Logarithm

Natural Logarithm

Syntax

<natural logarithm> ::=
LN (<numeric value expression>)

Notes

- The LN function returns the natural logarithm of the argument.
- The argument shall be greater than 0.
- The data type of the result is DOUBLE PRECISION.
- If the argument is null, then the result is null.

Examples

Expression	Result
LN(2)	0.693147..
LN(7.5)	2.014903..

Conformance

SQL:2003 standard - Feature T621 "Enhanced numeric functions"

19.2.7 General Logarithm

General Logarithm

Syntax

```
<general logarithm> ::=  
    LOG ( <numeric value expression>, <numeric value expression> )
```

Notes

- The LOG function returns the logarithm of the 2nd argument, using the 1st argument as the base.
- The arguments shall be greater than 0.
- The data type of the result is DOUBLE PRECISION.

- If either argument is null, then the result is null.

Examples

Expression	Result
LOG(10, 100)	2
LOG(2, 8)	3

Conformance

SQL:2003 standard - Feature T621 "Enhanced numeric functions"

19.2.8 Exponential Function

Exponential Function

Syntax

```
<exponential function> ::=  
    EXP ( <numeric value expression> )
```

Notes

- The result of the EXP function is e (the base of natural logarithms) raised to the power of the argument.
- The data type of the result is DOUBLE PRECISION.
- If the argument is null, then the result is null.

Examples

Expression	Result

EXP(2)	7.389056..
EXP(-2)	0.135335..

Conformance

SQL:2003 standard - Feature T621 "Enhanced numeric functions"

19.2.9 Power Function

Power Function

Syntax

```
<power function> ::=  
    POWER ( <numeric value expression> , <numeric value expression> )
```

Notes

- The POWER function returns the value of the first argument raised to the power of the second argument.
- If both arguments are 0, then the result is 1.
- If the first argument is 0 and the second argument is positive, then the result is 0.
- If the first argument is negative, then the second argument shall be an exact numeric value with scale 0.
- The data type of the result is DOUBLE PRECISION.
- If either of the arguments are null, then the result is null.

Examples

Expression	Result

POWER(0, 0)	1
POWER(0, 3)	0
POWER(-2, 3)	-8
POWER(2, 3)	8

Conformance

SQL:2003 standard - Feature T621 "Enhanced numeric functions"

19.2.10 Square Root Function

Square Root Function

Syntax

```
<square root> ::=  
    SQRT ( <numeric value expression> )
```

Notes

- The SQRT function returns the square root of the argument.
- The data type of the result is DOUBLE PRECISION.
- If the argument is null, then the result is null.

Examples

Expression	Result
SQRT(4)	2
SQRT(5)	2.236068..

Conformance

SQL:2003 standard - Feature T621 "Enhanced numeric functions"

19.2.11 Floor Function

Floor Function

Syntax

<floor function> ::=
 FLOOR (<numeric value expression>)

Notes

- The FLOOR function returns the greatest whole number (scale 0) that is less than or equal to the argument.
- The data type of the result is the same type as the argument.
- If the argument is null, then the result is null.

Examples

Expression	Result
FLOOR(5)	5
FLOOR(5.25)	5
FLOOR(-5.25)	-6

Conformance

SQL:2003 standard - Feature T621 "Enhanced numeric functions"

19.2.12 Ceiling Function

Ceiling Function

Syntax

<ceiling function> ::=
 { CEIL | CEILING } (<numeric value expression>)

Notes

- The CEILING function returns the smallest whole number (scale 0) that is greater than or equal to the argument.
- The data type of the result is the same type as the argument.
- If the argument is null, then the result is null.

Examples

Expression	Result
CEILING(5)	5
CEILING(5.25)	6
CEILING(-5.25)	-5

Conformance

SQL:2003 standard - Feature T621 "Enhanced numeric functions"

19.2.13 Arctangent Function

Arctangent Function

Syntax

<arctangent function> ::=
 | ATAN (<numeric value expression>)
 { ATN2 | ATAN2 } (<numeric value expression> , <numeric value expression>)

Notes

- The ATAN function returns the angle in radians, whose tangent is the value of the argument.

- The ATAN2 function returns the angle in radians, whose tangent is between the values of the two arguments.
- The data type of the result is DOUBLE PRECISION.
- If either of the arguments are null, then the result is null.

Examples

Expression	Result
ATAN(2.75)	1.222025..
ATAN2(2.75, 7.3525)	0.357913..

Conformance

NexusDB
extensions - ATAN function
 - ATAN2 function

19.2.14 Cosine Function

Cosine Function

Syntax

```
<cosine function> ::=  
                  COS ( <numeric value expression> )
```

Notes

- The COS function returns the trigonometric cosine of the argument (in radians).
- The data type of the result is DOUBLE PRECISION.
- If the argument is null, then the result is null.

Examples

Expression	Result
COS(5.25)	0.512085..
COS(-2)	-0.416147..

Conformance

NexusDB - COS function
extensions

19.2.15 Sine Function

Sine Function

Syntax

<sine function> ::=
 SIN (<numeric value expression>)

Notes

- The SIN function returns the trigonometric sine of the argument (in radians).
- The data type of the result is DOUBLE PRECISION.
- If the argument is null, then the result is null.

Examples

Expression	Result
SIN(5.25)	-0.858935..
SIN(-2)	-0.909297..

Conformance

NexusDB
extensions - SIN function

19.2.16 Ordinal Function

Ordinal Function

Syntax

<ordinal function> ::=
ORD (<character value expression>)

Notes

- The ORD function returns the ordinal value of the character passed by the argument.
- The data type of the result is INTEGER.
- If the argument is null, then the result is null.

Examples

Expression	Result
ORD('A')	65
ORD('a')	97
ORD('5')	53

Conformance

NexusDB
extensions - ORD function

19.2.17 PI Function

PI Function

Syntax

<pi function> ::= PI

Notes

- The PI function returns the numeric value **pi** (3.141592..).
 - The data type of the result is DOUBLE PRECISION.
-

Conformance

NexusDB - PI function
extensions

19.2.18 Round Function

Round Function

Syntax

<round function> ::=
| ROUND (<numeric value expression>)
| BROUND (<numeric value expression>)

Notes

- The ROUND function returns the value of the argument rounded to the closest whole number (scale 0).
- The BROUND function uses the "Banker's Rounding" algorithm to calculate the result.
- The data type of the result is the same type as the argument.
- If the argument is null, then the result is null.

Examples

Expression	Result
ROUND(6.5)	7.0
BROUND(6.5)	6.0
ROUND(7.5)	8.0
BROUND(7.5)	8.0

Conformance

NexusDB - ROUND function
 extensions - BROUND function

19.2.19 Random Function

Random Function

Syntax

<random function> ::= RAND

Notes

- The RAND function returns a random floating point value between 0 and 1.
- The data type of the result is DOUBLE PRECISION.

Conformance

NexusDB - RAND function
 extensions

19.2.20 LastAutoinc Function

LastAutoinc Function

Syntax

```
<lastautoinc function> ::=  
| LASTAUTOINC  
IDENTITY
```

Notes

- The LASTAUTOINC function returns the most recent Autolnc value generated by the current SQL session.
- The data type of the result is DWORD.
- If no Autolnc value has been generated during the current SQL session, then the result is null.
- The LASTAUTOINC function should always be executed inside a transaction when the result is used for assignments.
- The IDENTITY keyword, which is a synonym for the LASTAUTOINC function, is deprecated in V2 and may be deleted in a future version of NexusDB SQL.

Examples

- 1) Assuming master.masterID is an AUTOINC column, the following example first inserts a new row into master, which causes the AUTOINC column to be incremented, and then inserts a new row into the detail table using the LASTAUTOINC function to assign the last generated Autolnc value to the detail.masterID column:

```
START TRANSACTION;  
INSERT INTO master ( masterID ) VALUES  
( 100 );  
INSERT INTO detail ( masterID, detailID ) VALUES  
( LASTAUTOINC, 1 );  
COMMIT;
```

Conformance

NexusDB extensions - LASTAUTOINC function

19.3 String Value Functions

String Value Functions

Specify a function that returns a character string value.

Syntax

```
<string value function> ::=  
    <character value function>  
  
<character value function> ::=  
    | <character substring function>  
    | <fold>  
    | <trim function>  
    | <char function>  
    | <tostring function>  
    | <tostringlen function>  
    | OVERLAY Function
```

Usage

String value functions can be used in SQL everywhere a character string value is permitted.

Conformance

SQL:2003 standard - Core SQL

NexusDB extensions - CHR function
- TOSTRING function
- TOSTRINGLEN function

19.3.1 Substring Function

Substring Function

Syntax

```
<character substring function> ::=  
  SUBSTRING ( <character value expression>  
    FROM [ LEFT | RIGHT ] <integer expression>  
    [ { FOR | TO } [ LEFT | RIGHT ] <integer expression> ]  
    [ USING { CHARACTERS | OCTETS } ]  
  )
```

Alternative syntax supporting regular expression matching:

```
SUBSTRING ( <character value expression>  
  SIMILAR [ TO ] <character value expression> [ "IGNORE" "CASE" ]  
  [ "GROUP" <character value or integer expression> ]  
)
```

Notes

- The SUBSTRING function returns a subset of characters from the source string.
- The start-position argument specifies the ordinal position within the source string where the substring starts.
- The string-length argument specifies how many characters or octets shall be copied from the source string.
- If no direction is specified for FROM, the LEFT is the default.
If no direction is specified for FOR | TO then the direction from FROM is the default.

FROM and TO are absolute positions, counting from the start (LEFT) or end (RIGHT) of the string. With 1 being the first or last character in the string. (Keeping in mind that for CHAR instead of VARCHAR the string always has the full length and is padded with spaces at the end).

FOR counts from the FROM position either forward (LEFT) or backward (RIGHT) to determine the absolute position. If backward, this effectively calculates the start position and the value from FROM becomes the end position.

If the calculated absolute start or end position is beyond the start or end of the string, it is clamped to fall within the string.

The string is always copied left-to-right from the absolute start to the absolute end position. (So if the end position is before the start position, an empty string is returned).

- If CHARACTERS is specified, or the USING clause is omitted, then start-position is the ordinal character position within the source string. If OCTETS is specified, then start-position is the byte position.

- If the declared type of the character value expression is a fixed-length or variable-length character string type, then the data type of the result is CHARACTER VARYING or NATIONAL CHARACTER VARYING with a length equal to the fixed length or maximum variable length of the character value expression, otherwise the data type of the result is CHARACTER LARGE OBJECT or NATIONAL CHARACTER LARGE OBJECT.
- When the alternative SIMILAR [TO] regular expression syntax is used, the regular expressions are evaluated using the PCRE library.

NOTE! The required library to support regular expressions does not exist in Rad Studio/Delphi versions before the XE version. An exception will be raised when using the SIMILAR [TO] syntax If the SQL engine is compiled with a pre-XE version. The pre-compiled NexusDB executables fully support regular expressions.

A quick overview of the supported syntax can be found at
<https://www.pcre.org/original/doc/html/pcresyntax.html>

The full specification of valid patterns can be found at
<https://www.pcre.org/original/doc/html/pcrepattern.html>

The optional GROUP parameter can be either an integer or a string to extract only the value of a numbered subpattern or named subpattern as described in the "SUBPATTERNS" and "NAMED SUBPATTERNS" section of the full pattern specification.

See below for an example of what is now possible.

- If either of the arguments are null, then the result is null.

Examples

Expression	Result
SUBSTRING('Hello World' FROM 1)	'Hello World'
SUBSTRING('Hello World' FROM 1 FOR 5)	'Hello'
SUBSTRING('Hello World' FROM 7)	'World'
SUBSTRING('Hello World' FROM 7 FOR 5)	'World'
SUBSTRING('this is a test' FROM /*LEFT*/ 5)	is a test
SUBSTRING('this is a test' FROM /*LEFT*/ 1 FOR /*LEFT*/ 4)	this
SUBSTRING('this is a test' FROM LEFT 1 FOR /*LEFT*/ 4)	this
SUBSTRING('this is a test' FROM LEFT 4 FOR RIGHT 2)	is
SUBSTRING('this is a test' FROM RIGHT 1 FOR /*RIGHT*/ 4)	test
SUBSTRING('this is a test' FROM RIGHT 4 FOR LEFT 4)	test

SUBSTRING('this is a test' FROM RIGHT 5)	this is a
SUBSTRING('this is a test' FROM /*LEFT*/ 2 TO /*LEFT*/ 3)	hi
SUBSTRING('this is a test' FROM LEFT 2 TO LEFT 3)	hi
SUBSTRING('this is a test' FROM RIGHT 3 TO /*RIGHT*/ 2)	es
SUBSTRING('this is a test' FROM RIGHT 10 TO LEFT 10)	" is a "

Regular expression style:

```
DECLARE s NVARCHAR(53);
SET S = 'the blue king and the White king and the white queen';

SELECT
    SUBSTRING(s SIMILAR TO 'the ((red|white) (king|queen))' AS "FULL",
    SUBSTRING(s SIMILAR TO 'the ((red|white) (king|queen))' GROUP 0) AS
"GROUP_0", -- same as not specifying GROUP
    SUBSTRING(s SIMILAR TO 'the ((red|white) (king|queen))' GROUP 1) AS
"GROUP_1",
    SUBSTRING(s SIMILAR TO 'the ((red|white) (king|queen))' GROUP 2) AS
"GROUP_2",
    SUBSTRING(s SIMILAR TO 'the ((red|white) (king|queen))' GROUP 3) AS
"GROUP_3",
    SUBSTRING(s SIMILAR TO 'the ((?<color>red|white) (?<gender>king|queen))'
GROUP 'color') AS "GROUP_COLOR",
    SUBSTRING(s SIMILAR TO 'the ((?<color>red|white) (?<gender>king|queen))'
GROUP 'gender') AS "GROUP_GENDER",
    SUBSTRING(s SIMILAR TO 'the (red|white) (?<gender>king|queen)' IGNORE
CASE GROUP 'gender') AS "GROUP_GENDER_IGNORECASE"
FROM
    #dummy;
```

which will return the following values:

```
FULL: the white queen
GROUP_0: the white queen
GROUP_1: white queen
GROUP_2: white
GROUP_3: queen
GROUP_COLOR: white
GROUP_GENDER: queen
GROUP_GENDER_IGNORECASE: king
```

Conformance

SQL:2003 standard - Core SQL

19.3.2 OVERLAY Function

OVERLAY function

The syntax has been extended from what the standard allows to allow the same target specification as the extended SUBSTRING function (see that function for details).

Syntax

```
OVERLAY (
    <character value expression> PLACING <character value expression>
    { FROM [ LEFT | RIGHT ] <integer expression>
        [ { FOR | TO } [ LEFT | RIGHT ] <integer expression> ]
        [ USING { CHARACTERS | OCTETS } ]
    | SIMILAR [ TO ] <character value expression> [ "IGNORE" "CASE" ]
        [ "GROUP" <character value or integer expression> ]
    }
)
```

Furthermore, OVERLAY has been extended to allow operating on DATE, TIME, and TIMESTAMP values.

In this case the syntax is:

```
OVERLAY (
    <date, time, or timestamp value expression> PLACING <integer or float value expression>
    IN {YEAR | MONTH | DAY | HOUR | MINUTE | SECOND }
```

Usage

The function allows to replace parts of a string with another string directly in place.

Here is an example of what is now possible:

```
DECLARE s NVARCHAR(70);
SET S = 'the blue king and the White king and the white queen';

SELECT
    OVERLAY(s PLACING 'the joker' SIMILAR TO 'the ((red|white) (king|queen))' IGNORE CASE),
    OVERLAY(s PLACING ' - gone - ' FROM LEFT 5 TO RIGHT 5),
    OVERLAY(s PLACING ' - gone - ' FROM 5)
FROM
    #dummy;
```

Resulting in the following 3 string:

```
the blue king and the joker and the white queen
the - gone - ueen
the - gone - and the White king and the white queen
```

19.3.3 Fold Function

Fold Function

Syntax

```
<fold> ::=  
    { UPPER | LOWER } ( <character value expression> )
```

Notes

- The UPPER function converts the source string to upper case.
- The LOWER function converts the source string to lower case.
- The data type of the result is the declared type of the argument.
- If the argument is null, then the result is null.

Examples

Expression	Result
UPPER('Hello World')	'HELLO WORLD'
LOWER('Hello World')	'hello world'

Conformance

SQL:2003 standard - Core SQL

19.3.4 Trim Function

Trim Function

Syntax

```
<trim function> ::=  
    TRIM ( [ [ <trim specification> ] [ <trim character> ] FROM ] <trim source>
```

```

<trim specification> ::=  

  | LEADING  

  | TRAILING  

  BOTH

<trim character> ::= <character value expression>

<trim source> ::= <character value expression>

```

Notes

- The TRIM function removes leading and/or trailing characters from the source string.
- If no trim specification is specified, then BOTH is implicit.
- If no trim character is specified, then space is implicit.
- If the declared type of the trim source is a fixed-length or variable-length character string type, then the data type of the result is CHARACTER VARYING or NATIONAL CHARACTER VARYING with a length equal to the fixed length or maximum variable length of the trim source, otherwise the data type of the result is CHARACTER LARGE OBJECT or NATIONAL CHARACTER LARGE OBJECT.
- If either of the arguments are null, then the result is null.

Examples

Expression	Result
TRIM(' Meet the Dream Team ')	'Meet the Dream Team'
TRIM(BOTH FROM ' Meet the Dream Team ')	'Meet the Dream Team'
TRIM(LEADING 'M' FROM 'Meet the Dream Team')	'eet the Dream Team'
TRIM(TRAILING 'M' FROM 'Meet the Dream Team')	'Meet the Dream Tea'
TRIM(BOTH 'M' FROM 'Meet the Dream Team')	'eet the Dream Tea'

Conformance

SQL:2003 standard - Core SQL

19.3.5 Char Function

Char Function

Syntax

```
<char function> ::=  
    CHR ( <numeric value expression> )
```

Notes

- The CHR function returns a character corresponding to the value of the argument.
- The argument shall be within the range 1..255.
- If the argument is null, then the result is null.

Examples

- 1) The following example selects all customer rows with a CR/LF pair in the Memo column:

```
SELECT *  
FROM customers  
WHERE memo LIKE '%' || CHR( 13 ) || CHR( 10 ) ||  
'%'
```

Conformance

NexusDB - CHR function
extensions

19.3.6 Tostring Function

Tostring Function

Syntax

```
<tostring function> ::=  
    TOSTRING ( <value expression> )
```

Notes

- The TOSTRING function returns a text representation of the argument.
- The data type of the result is CHARACTER.
- If the argument is null, then the result is null.

Examples

- 1) The following example converts the studentID column to a character string:

```
SELECT studentName, TOSTRING( studentID )
AS student_code
FROM students
ORDER BY student_code
```

Conformance

NexusDB - TOSTRING function
extensions

19.3.7 Tostringlen Function

Tostringlen Function

Syntax

```
<tostringlen function> ::=  
    TOSTRINGLEN ( <value expression> , max-length )
```

Notes

- The TOSTRINGLEN function is equal to the TOSTRING function, except that the result is limited to the length specified by the *max-length* argument.

- The data type of the result is CHARACTER.
- If the value expression is null, then the result is null.

Examples

- 1) The following example shows a computed column with 100 characters extracted from a BLOB column:

```
SELECT documentID, title,  
TOSTRINGLEN( content, 100 ) AS hex_content  
FROM documents  
ORDER BY documentID
```

Conformance

NexusDB
extensions

- TOSTRINGLEN function

19.4 DateTime Value Functions

DateTime Value Functions

Specify a function that returns a datetime value.

Syntax

```
<datetime value function> ::=  
| <current date value function>  
| <current time value function>  
| <current timestamp value function>  
| OVERLAY Function
```

Usage

DateTime value functions can be used in SQL everywhere a datetime value is permitted.

Conformance

SQL:2003 standard - Core SQL

19.4.1 Current Date Function

Current Date Function

Syntax

<current date value function> ::= CURRENT_DATE

Notes

- The CURRENT_DATE function returns the system date on the server computer.

Examples

- The following example selects all future appointments:

```
SELECT *
FROM appointments
WHERE bookingDate >= CURRENT_DATE
```

Conformance

SQL:2003 standard - Core SQL

19.4.2 Current Time Function

Current Time Function

Syntax

<current time value function> ::=
 | CURRENT_TIME
 LOCALTIME

Notes

- The CURRENT_TIME function has an implicit time zone specification equal to the time offset on the server computer, and is therefore equivalent to LOCALTIME in this version of NexusDB SQL.
- The LOCALTIME function returns the local time on the server computer.

Examples

- 1) The following example selects all remaining appointments for today:

```
SELECT *
FROM appointments
WHERE bookingDate = CURRENT_DATE AND
apptTime >= CURRENT_TIME
```

Conformance

SQL:2003 standard - Core SQL

19.4.3 Current Timestamp Function

Current Timestamp Function

Syntax

```
<current timestamp value function> ::=
| CURRENT_TIMESTAMP
LOCALTIMESTAMP
```

Notes

- The CURRENT_TIMESTAMP function has an implicit time zone specification equal to the time offset on the server computer, and is therefore equivalent to LOCALTIMESTAMP in this version of NexusDB SQL.
- The LOCALTIMESTAMP function returns the system date and local time on the server computer.

Examples

- 1) The following example selects all new appointments made today:

```
SELECT *
FROM appointments
WHERE bookingDate BETWEEN
CURRENT_DATE AND CURRENT_TIMESTAMP
```

Conformance

SQL:2003 standard - Core SQL

19.4.4 OVERLAY Function

OVERLAY function

OVERLAY, which is commonly used for strings, has also been extended to allow operating on DATE, TIME, and TIMESTAMP values.

Syntax

```
OVERLAY (
<date, time, or timestamp value expression> PLACING <integer or float value expression>
IN {YEAR | MONTH | DAY | HOUR | MINUTE | SECOND } )
```

Usage

The function allows to replace parts of a datetime value with another value directly in place.

For YEAR..MINUTE the PLACING expression is interpreted as an integer expression, for SECOND as a float expression, containing both seconds and milliseconds as a fraction.

The value being placed must result in a valid date/time (so HOUR must be 0..23, MINUTE or SECOND 0..59.9999, DAY can't be beyond the end of the specific month, and so on).

When a YEAR or MONTH is set that would result in the existing DAY component to be beyond the valid range for that month (e.g. 29th Feb when not a leap year, or 31st Apr) then the DAY value is clamped to the last day of that month.

Here is an example of what is now possible:

```
SELECT
    OVERLAY(CURRENT_DATE PLACING 2020 IN YEAR),
    OVERLAY(CURRENT_DATE PLACING 1 IN MONTH),
    OVERLAY(CURRENT_DATE PLACING 2 IN DAY),
    OVERLAY(CURRENT_TIME PLACING 3 IN HOUR),
    OVERLAY(CURRENT_TIME PLACING 4 IN MINUTE),
    OVERLAY(CURRENT_TIME PLACING 5.1234 IN SECOND),
    OVERLAY(DATE'2004-02-29' PLACING 2005 IN YEAR),
    OVERLAY(DATE'2004-01-31' PLACING 4 IN MONTH)
FROM
    #dummy;
```

19.5 System Functions

System Functions

The following table shows the special system functions supported in NexusDB SQL.

Syntax	Data type	Notes
USER	VARCHAR	Returns the user name associated with the current session.
CURRENT_USER	VARCHAR	Equivalent to USER.
SESSION_USER	VARCHAR	Equivalent to USER.
SYSTEM_ROW#	DWORD	The function can only be defined as the first select column in query specifications, and will assign a sequential (zero-based) row-id to the rows in the result set. Cannot be used inside other functions.
NEWGUID	GUID	Returns a generated GUID value.
ROWSREAD		Returns the number of rows in the result set of a SELECT statement executed prior to calling ROWSREAD.
ROWSAFFECTED		Returns the number of rows affected by a data-change statement executed prior to calling ROWSaffected.
ERROR_MESSAGE	VARCHAR	Returns the last error message from the system. The function can only be used in the CATCH clause of the TRY statement.
SYS_REPARSEALL	CLOB	<p>Returns the definition of a stored function that, when run via EXECUTE IMMEDIATE SYS_REPARSEALL(), regenerates all stored procedures, functions, and triggers in the database.</p> <p>The "EXECUTE IMMEDIATE" is important as SYS_REPARSEALL only returns the text of the SQL function to execute.</p> <p>The statement will return a resultset with the following format:</p>

```

CREATE TABLE #ROUTINES (
    TYPE NVARCHAR(9),
    NAME NVARCHAR(255),
    SOURCE NCLOB,
    SUCCESS BOOLEAN,
    ERROR_MSG NCLOB,
    PRIMARY KEY (TYPE, NAME)
);

TYPE is PROCEDURE, FUNCTION or TRIGGER
NAME is the name
SOURCE is the source code of the function
SUCCESS indicates if the function was successfully
reparsed
ERROR_MSG is whatever error was SIGNALed during
the reparse in case of failure.

```

Examples

- 1) The following example selects all appointments made by the current user:

```

SELECT *
FROM orders
WHERE takenBy = CURRENT_USER

```

- 2) The following example retrieves a unique row-id and the customer name:

```

SELECT SYSTEM_ROW# AS row_id,
customer_name
FROM customers

```

- 3) The following example uses the NEWGUID function to assign a value to the documentID column:

```

INSERT INTO documents ( documentID, title )
VALUES ( NEWGUID, 'NexusDB features' )

```

- 4) The following example shows how ERROR_MESSAGE is used in the TRY statement:

```

START TRANSACTION;
TRY

```

```
DROP TABLE garbage;  
COMMIT;  
  
CATCH POSITION( 'Unable to open table' IN  
ERROR_MESSAGE ) <> 0  
  
ROLLBACK;  
SIGNAL ERROR_MESSAGE;  
  
END;
```

Conformance

SQL:2003 standard - Core SQL

NexusDB extensions	- SYSTEM_ROW# - NEWGUID - ROWSREAD - ROWSAFFECTED - ERROR_MESSAGE
-----------------------	---

20 Predicates

Predicates

Specify a condition that evaluates to a boolean value.

Syntax

```
<predicate> ::=  
| <comparison predicate>  
| <between predicate>  
| <in predicate>  
| <like predicate>  
| <>null predicate>  
| <quantified comparison predicate>  
| <exists predicate>  
| <unique predicate>  
| <match predicate>  
| <equivalent predicate>  
| <odd predicate>  
| <SIMILAR Predicate>  
| <inserting predicate>  
| <updating predicate>  
| <deleting predicate>  
| SIMILAR Predicate  
| CONTAINS Predicate
```

<search condition> ::= <boolean value expression>

Usage

Predicates are boolean value expressions, and can be used in SQL everywhere a boolean value is permitted.

Search conditions

The main function of the predicates is to specify a **search condition** that is used in data statements to join tables and filter out unwanted rows from a set, and elsewhere in SQL a conditional computation is specified. The search condition can be a single predicate or multiple predicates combined with the AND and OR boolean operators.

Conformance

SQL:2003 standard - Core SQL

- Feature F561 "Full value expressions"
- Feature F281 "LIKE enhancements"
- Feature F481 "Expanded NULL predicate"

- Feature T501 "Enhanced EXISTS predicate"
- Feature F291 "UNIQUE predicate"
- Feature F741 "Referential MATCH types"

NexusDB extensions	<ul style="list-style-type: none">- EQUIVALENT predicate- ODD predicate- INSERTING predicate- UPDATING predicate- DELETING predicate
-----------------------	--

20.1 Comparison Predicate

Comparison Predicate

Specify a comparison of two row values.

Syntax

```
<comparison predicate> ::=  
    <row value predicand> <comparison predicate part 2>  
  
<comparison predicate part 2> ::= <comp op> <row value predicand>  
  
<comp op> ::=  
    | <equals operator>  
    | <not equals operator>  
    | <less than operator>  
    | <greater than operator>  
    | <less than or equals operator>  
    | <greater than or equals operator>
```

Notes

- The degree of the row value predicands shall be 1.
- The data types shall be comparable.
- The predicate evaluates to TRUE if the comparison criteria is met, else to FALSE.
- If either of the row values are null, then the predicate evaluates to UNKNOWN.

Examples

- 1) The following example selects students who are not from California:

```
SELECT studentName, state  
FROM students  
WHERE state <> 'CA'
```

- 2) The following example selects teachers who earn \$35,000 or more:

```
SELECT teacherName, salary  
FROM teachers  
WHERE salary >= 35000
```

Conformance

SQL:2003 standard - Core SQL

20.2 Between Predicate

Between Predicate

Specify a range comparison.

Syntax

```
<between predicate> ::=  
    <row value predicand> <between predicate part 2>  
  
<between predicate part 2> ::=  
    [ NOT ] BETWEEN <row value predicand> AND <row value predicand>
```

Notes

- The BETWEEN predicate is equal to:

$$\text{value}_1 \geq \text{value}_2 \text{ AND } \text{value}_1 \leq \text{value}_3$$

- The degree of the row value predicands shall be 1.

- The data types shall be comparable.
- The result of evaluating the predicate is shown in the Truth table for the AND boolean operator.
- The NOT operator inverts the meaning of the comparison and is equal to:

NOT (<row value predicand> BETWEEN <row value predicand> AND <row value predicand>)

Examples

- 1) The following example selects teachers who earn between \$30,000 and \$35,000:

```
SELECT teacherName, salary  
FROM teachers  
WHERE salary BETWEEN 30000 AND 35000
```

Conformance

SQL:2003 standard - Core SQL

20.3 In Predicate

In Predicate

Specify a quantified comparison.

Syntax

```
<in predicate> ::=  
    <row value predicand> <in predicate part 2>  
  
<in predicate part 2> ::= [ NOT ] IN <in predicate value>  
  
<in predicate value> ::=  
    | <table subquery>  
    ( <row value expression> [ { , <row value expression> }... ] )
```

Notes

- The degree of the row value predicand, table subquery and row value expressions shall be 1.
- The data types shall be comparable.
- The predicate evaluates to TRUE if the row value predicand has a matching value in the list of values specified by the in predicate value, else to FALSE.
- If the row value predicand is null, then the predicate evaluates to UNKNOWN.
- The NOT operator inverts the meaning of the comparison and is equal to:

NOT (<row value predicand> IN <in predicate value>)

Examples

- 1) The following example selects students from California and New York:

```
SELECT studentName, state  
FROM students  
WHERE state IN ( 'CA', 'NY' )
```

- 2) The following examples uses a subquery with the IN predicate to select students who are enrolled in courseID 730:

```
SELECT studentID, studentName  
FROM students  
WHERE studentID IN ( SELECT studentID FROM  
enrolls WHERE courseID = 730 )
```

Conformance

SQL:2003 standard - Core SQL
- Feature F561 "Full value expressions"

20.4 Like Predicate

Like Predicate

Specify a pattern-match comparison.

Syntax

```
<like predicate> ::=  
    <row value predicand> <character like predicate part 2> [ IGNORE CASE ]  
  
<character like predicate part 2> ::=  
    [ NOT ] LIKE <character pattern> [ ESCAPE escape-character ]  
  
<character pattern> ::= <character value expression>
```

Notes

- The data types of the row value predicand, character pattern and escape-character shall be comparable character string types.
- The predicate evaluates to TRUE if the row value predicand string matches the character pattern string, else to FALSE.
- Underscore (_) is used as a wildcard character representing an arbitrary single character specifier at the location in character pattern.
- Percent (%) is used as a wildcard character representing an arbitrary string specifier at the location in character pattern.
- The ESCAPE clause causes the character following the escape-character to be treated as a normal character, which allows the special wildcard characters to be used as single character specifiers in character pattern.
- If any of the row value predicand, character pattern or escape-character values are null, then the predicate evaluates to UNKNOWN.
- The NOT operator inverts the meaning of the comparison and is equal to:

NOT (<row value predicand> LIKE <character pattern> [ESCAPE escape-character])

- IGNORE CASE makes the comparison operation case-insensitive.

Examples

- 1) The following example selects students whose first name starts with 'Jo':

```
SELECT studentName, gender  
FROM students  
WHERE studentName LIKE 'Jo%'
```

- 2) The following example selects students whose first name has an 'o' in the second position:

```
SELECT studentName, gender  
FROM students  
WHERE studentName LIKE '_o%'
```

- 3) The following example selects orders with '10%' as a portion of the notes column value:

```
SELECT orderID, notes  
FROM orders  
WHERE notes LIKE '%10\%%' ESCAPE '\'
```

Conformance

SQL:2003 standard - Core SQL
- Feature F281 "LIKE enhancements"

NexusDB - IGNORE CASE
extensions

20.5 Null Predicate

Null Predicate

Specify a test for a null value.

Syntax

```
<null predicate> ::=  
    <row value predicand> <null predicate part 2>  
  
<null predicate part 2> ::= IS [ NOT ] NULL
```

Notes

- The degree of the row value predicand shall be 1.
- IS NULL evaluates to TRUE if the row value is null, else to FALSE.
- IS NOT NULL evaluates to TRUE if the row value is different from null, else to FALSE.

Examples

- 1) The following example selects students who don't take any courses:

```
SELECT studentID, studentName  
FROM students s  
left join enrolls e ON e.studentID = s.studentID  
WHERE e.studentID IS NULL
```

Conformance

SQL:2003 standard - Core SQL
- Feature F481 "Expanded NULL predicate"

20.6 Quantified Comparison Predicate

Quantified Comparison Predicate

Specify a quantified comparison.

Syntax

```

<quantified comparison predicate> ::= 
    <row value predicand> <quantified comparison predicate part 2>

<quantified comparison predicate part 2> ::= 
    <comp op> <quantifier> <table subquery>

<quantifier> ::= 
    | ALL
    { SOME | ANY }

```

Notes

- The degree of the row value predicand and table subquery shall be 1.
- The data types shall be comparable.
- If ALL is specified, then the predicate is evaluated as follows:
 - a) If the result of the subquery is empty or the comparison is true for every row in the subquery, then the result is TRUE.
 - b) If the comparison is false for at least one row in the subquery, then the result is FALSE.
 - c) If any of the values returned by the subquery is null and none of the comparisons are false, then the result is UNKNOWN.
- If SOME or ANY is specified, then the predicate is evaluated as follows:
 - a) If the comparison is true for at least one row in the subquery, then the result is TRUE.
 - b) If the result of the subquery is empty or the comparison is false for every row in the subquery, then the result is FALSE.
 - c) If any values returned by the subquery is null and none of the comparisons are true, then the result is UNKNOWN.

- If the row value predicand is null, then the predicate evaluates to UNKNOWN.

Examples

- 1) The following example selects students enrolled in all available courses:

```
SELECT studentID, studentName
FROM students s
WHERE studentID = ALL (
    SELECT e.studentID
    FROM courses c
    LEFT JOIN enrolls e on
        e.courseID = c.courseID AND e.studentID =
        s.studentID
)
```

- 2) The following example selects students enrolled in at least one course:

```
SELECT studentID, studentName
FROM students s
WHERE studentID = ANY (
    SELECT e.studentID
    FROM courses c
    LEFT JOIN enrolls e on
        e.courseID = c.courseID AND e.studentID =
        s.studentID
)
```

Conformance

SQL:2003 standard - Core SQL

20.7 Exists Predicate

Exists Predicate

Specify a test for a non-empty set.

Syntax

<exists predicate> ::= EXISTS <table subquery>

Notes

- The predicate evaluates to TRUE if the cardinality of the subquery result is greater than 0, else to FALSE.

Examples

- The following example selects students enrolled in more than 2 courses:

```
SELECT studentID, studentName
FROM students s
WHERE EXISTS (
    SELECT COUNT( * )
    FROM enrolls
    WHERE studentID = s.studentID
    HAVING COUNT( * ) > 2
)
```

- The following example selects students not enrolled in any courses:

```
SELECT studentID, studentName
FROM students s
WHERE EXISTS (
    SELECT COUNT( * )
    FROM enrolls
    WHERE studentID = s.studentID
    HAVING COUNT( * ) = 0
)
```

Conformance

SQL:2003 standard - Core SQL
- Feature T501 "Enhanced EXISTS predicate"

20.8 Unique Predicate

Unique Predicate

Specify a test for the absence of duplicate rows.

Syntax

<unique predicate> ::= UNIQUE <table subquery>

Notes

- The predicate evaluates to TRUE if the subquery result is empty or it contains no duplicate rows, else to FALSE.

Examples

- 1) The following example selects students enrolled in none or only 1 course:

```
SELECT studentID, studentName
FROM students s
WHERE UNIQUE (
    SELECT studentID
    FROM enrolls
    WHERE studentID = s.studentID
)
```

Conformance

SQL:2003 standard - Feature F291 "UNIQUE predicate"

20.9 Match Predicate

Match Predicate

Specify a test for matching rows.

Syntax

```
<match predicate> ::=  
    <row value predicand> <match predicate part 2>  
  
<match predicate part 2> ::=  
    MATCH [ UNIQUE ] [ SIMPLE | PARTIAL | FULL ] <table subquery>
```

Notes

- The degree of the row value predicand and table subquery shall be 1.
- The data types shall be comparable.
- If neither SIMPLE, PARTIAL, nor FULL is specified, then SIMPLE is implicit. Since this version of NexusDB only allows row value constructors with a degree of 1 (one column), specifying SIMPLE, PARTIAL or FULL will evaluate to exactly the same result.
- The predicate evaluates to TRUE if the row value predicand is null, or the row value predicand is different from null and there is at least one row in the subquery result matching the row value predicand, else to FALSE.
- If UNIQUE is specified, then the predicate evaluates to TRUE only if there is exactly one row in the subquery result matching the row value predicand.

Examples

- 1) The following example selects students enrolled in any number of courses:

```
SELECT studentID, studentName  
FROM students  
WHERE studentID MATCH (  
  
    SELECT studentID  
    FROM enrolls  
  
)
```

2) The following example selects students enrolled in exactly 1 course:

```
SELECT studentID, studentName  
FROM students  
WHERE studentID MATCH UNIQUE (  
  
    SELECT studentID  
    FROM enrolls  
  
)
```

Conformance

SQL:2003 standard - Feature F741 "Referential MATCH types"

20.10 Equivalent Predicate

Equivalent Predicate

Compares two values for equivalence.

Syntax

```
<equivalent predicate> ::=  
    EQUIVALENT ( <value expression>, <value expression> )
```

Notes

- The data types shall be comparable.
- The predicate evaluates to TRUE if the two values are equal, else to FALSE.
- Null values are considered equal in this predicate.

Examples

- 1) The following example selects students from Newport:

```
SELECT studentID, studentName, city
FROM students
WHERE EQUIVALENT ( city, 'Newport' )
```

Conformance

NexusDB - EQUIVALENT predicate
extensions

20.11 Odd Predicate

Odd Predicate

Specify a test for odd numbers.

Syntax

<odd predicate> ::= ODD (<numeric value expression>)

Notes

- The data type shall be numeric.
- The predicate evaluates to TRUE if the value is an odd number, else to FALSE.
- If the given value is null, then the predicate evaluates to UNKNOWN.

Examples

- 1) The following example selects students with odd studentID:

```
SELECT studentID, studentName
FROM students
WHERE ODD( studentID )
```

Conformance

NexusDB - ODD predicate
extensions

20.12 Inserting Predicate

Inserting Predicate

Testing if a trigger event is INSERT.

Syntax

<inserting predicate> ::= INSERTING

Notes

- The INSERTING predicate can only be specified in the body of a trigger definition to check the trigger event.
- The predicate evaluates to TRUE if the trigger was activated by an INSERT operation on the table, else to FALSE.

Examples

- 1) The following example shows how the INSERT predicate is used in the WHEN clause of a triggered action:

WHEN (INSERTING)
triggered SQL statements

- 2) The following example shows how the INSERT predicate is used in a triggered SQL statement:

IF INSERTING THEN ... END IF;

Conformance

NexusDB - INSERTING predicate
extensions

20.13 Updating Predicate

Updating Predicate

Testing if a trigger event is UPDATE.

Syntax

<updating predicate> ::= UPDATING

Notes

- The UPDATING predicate can only be specified in the body of a trigger definition to check the trigger event.

- The predicate evaluates to TRUE if the trigger was activated by an UPDATE operation on the table, else to FALSE.

Examples

- 1) The following example shows how the UPDATE predicate is used in the WHEN clause of a triggered action:

WHEN (UPDATING)
triggered SQL statements

- 2) The following example shows how the UPDATE predicate is used in a triggered SQL statement:

IF UPDATING THEN ... END IF;

Conformance

NexusDB extensions - UPDATING predicate

20.14 Deleting Predicate

Deleting Predicate

Testing if a trigger event is DELETE.

Syntax

<deleting predicate> ::= DELETING

Notes

- The DELETING predicate can only be specified in the body of a trigger definition to check the trigger event.
- The predicate evaluates to TRUE if the trigger was activated by an DELETE operation on the table, else to FALSE.

Examples

- 1) The following example shows how the DELETING predicate is used in the WHEN clause of a triggered action:

WHEN (DELETING)
triggered SQL statements

- 2) The following example shows how the DELETING predicate is used in a triggered SQL statement:

IF DELETING THEN ... END IF;

Conformance

NexusDB
extensions - DELETING predicate

20.15 SIMILAR Predicate

SIMILAR predicate

This predicate allows for regular expression matching. The exact syntax that NexusDB implements diverges from the SQL standard.

Syntax

```
<row value predicand> [ NOT ] SIMILAR [ TO ] <similar pattern> [ IGNORE CASE ]
```

Usage

The regular expressions are evaluated using the PCRE library.

A quick overview of the supported syntax can be found at
<https://www.pcre.org/original/doc/html/pcresyntax.html>

The full specification of valid patterns can be found at
<https://www.pcre.org/original/doc/html/pcrepattern.html>

SIMILAR evaluates to TRUE if the pattern evaluated against the predicand produces one or more matches, it evaluates to FALSE of there are 0 matches, if predicand or pattern is NULL, it returns NULL.

Here is an example of what is now possible:

```
CREATE LOCAL TEMPORARY TABLE #strings (
    string NVARCHAR(30)
);

INSERT INTO #strings VALUES ('the blue king');
INSERT INTO #strings VALUES ('the Red king');
INSERT INTO #strings VALUES ('the white prince');
INSERT INTO #strings VALUES ('the white queen');
INSERT INTO #strings VALUES ('red queen');

SELECT
    *,
    string SIMILAR TO 'the (red|white) (king|queen)',
    string SIMILAR TO 'the (red|white) (king|queen)' IGNORE CASE,
    string SIMILAR TO '(red|white) (king|queen)'
FROM
    #strings;
```

The produced output is:

the blue king - FALSE FALSE FALSE
 the Red king - FALSE TRUE FALSE

the white prince - FALSE FALSE FALSE
the white queen - TRUE TRUE TRUE
red queen - FALSE FALSE TRUE

20.16 CONTAINS Predicate

CONTAINS predicate

This predicate allows for searching on the contents of a fulltext index. The named fulltext index must exist in the table in order to use this predicate.

Syntax

```
<contains predicate> ::= CONTAINS(<SqlName>, <FullTextSearchExpr>)

<SqlName> ::= name of existing fulltext index

<FullTextSearchExpr> ::= { <FullTextSearchTerm> "OR" }

<FullTextSearchFactor> ::= [ "NOT" ] [ "STARTS" "WITH" ] <SimpleExpression>

<FullTextSearchTerm> ::= { <FullTextSearchFactor> "AND" }
```

Usage

```
SELECT * FROM Customers
WHERE CONTAINS(idx_ft_name, 'jack' OR 'jill')
```

21 Stored Procedures and Functions

Stored Procedures and Functions

The support for user-defined procedures and functions, commonly known as stored procedures, makes NexusDB SQL a programmable database language, allowing users to automate tasks and extend the built-in functionality with custom routines.

NexusDB SQL supports user-defined procedures and functions written in SQL, called SQL routines, or hosted in a .NET assembly, called CLR routines (Enterprise Edition only). Since user-defined procedures and functions share many common characteristics and syntactic elements, they are both called SQL-invoked routines in standard SQL.

The procedure language in NexusDB adds power and flexibility to the functionality of user-defined procedures and functions written in SQL.

User-defined procedures

A user-defined procedure (UDP) can have both IN and OUT parameters. The OUT parameter is used to return a value from the procedure, and is equivalent to the result of a scalar-valued function. Procedures can also return a cursor by specifying a SELECT statement as the last statement in the routine body.

User-defined procedures are defined with the CREATE PROCEDURE statement and are invoked in SQL by the CALL statement.

The following example shows a procedure definition and how it is called in SQL:

```
CREATE PROCEDURE addCourse (
    // Parameter declarations with implicit IN mode
    p_courseID SMALLINT,
    p_courseName CHAR(20),
    p_department CHAR(20),
    p_numCredits TINYINT
)
MODIFIES SQL DATA -- We need write-access to
update the courses table
// We prefer to use a compound statement in the
routine body, even with a single statement
BEGIN
    INSERT INTO courses
    VALUES ( p_courseID, p_courseName,
    p_department, p_numCredits );
END;
```

```
CALL  
addCourse( :courseID, :courseName, :department,  
:numCredits );
```

User-defined functions

Like functions in traditional programming languages, a user-defined function (UDF) always returns a value. The function result can be either a scalar value (scalar-valued function) or a table derived from a cursor specification (table-valued function), by specifying TABLE in the RETURNS clause of the function definition. A scalar-valued function can be referenced everywhere in SQL a scalar value is expected, while a table-valued function can be referenced in the FROM clause of a SELECT statement or invoked by calling the function as a separate SQL statement.

User-defined functions are defined with the CREATE FUNCTION statement and are invoked in SQL by the function name, see routine invocation.

The following example shows a function definition and how it is invoked in SQL:

```
CREATE FUNCTION getFullName ( firstName  
VARCHAR(30), lastName VARCHAR(30) )  
RETURNS VARCHAR(61)  
/*  
LANGUAGE SQL is implicit  
NOT DETERMINISTIC is implicit  
CONTAINS SQL is implicit  
*/  
RETURNS NULL ON NULL INPUT -- We don't  
invoke the function if any of the arguments are null  
  
// Multiple statements in the routine body must  
appear inside a compound statement  
BEGIN  
  
DECLARE name VARCHAR(61);  
SET name = firstName || ' ' || lastName;  
RETURN name;  
  
END;  
  
UPDATE students  
SET studentName = getFullName( firstName,  
lastName )  
WHERE studentID = 211;
```

SQL-Invoked Routines
SQL Procedure Statements

21.1 Procedure Language

Procedure Language

The procedure language in NexusDB SQL consists of a set of control statements that extends the SQL language with compound statement blocks (BEGIN .. END), SQL variable declarations, conditional statements, iterating statements and error handling. This functionality adds computational completeness to NexusDB SQL and makes it a fully programmable database language, which is particularly useful in the definition of triggers, user-defined procedures and user-defined functions.

CALL statement
RETURN statement
Compound statement
DECLARE variable statement
SET statement
IF statement
ITERATE statement
LEAVE statement
REPEAT statement
WHILE statement
SIGNAL statement
TRY statement

21.2 SQL-Invoked Routines

SQL-Invoked Routines

Define an SQL-invoked routine.

Syntax

```
<SQL-invoked routine> ::=  
| <user-defined procedure>  
| <user-defined function>
```

The following syntax elements are common to both procedures and functions:

```
<SQL parameter declaration list> ::=  
( [ <SQL parameter declaration> [ { , <SQL parameter declaration> }... ] ] )
```

```
<SQL parameter declaration> ::=  
[ <parameter mode> ] parameter-name <data type>
```

```

<parameter mode> ::=  

  | IN  

  | OUT  

  | INOUT

<routine characteristics> ::= [ <routine characteristic>... ]  

<routine characteristic> ::=  

  | <language clause>  

  | <deterministic characteristic>  

  | <SQL-data access indication>  

  | <null-call clause>

<language clause> ::= LANGUAGE { SQL | CLR }

<deterministic characteristic> ::=  

  | DETERMINISTIC  

  | NOT DETERMINISTIC

<SQL-data access indication> ::=  

  | NO SQL  

  | CONTAINS SQL  

  | READS SQL DATA  

  | MODIFIES SQL DATA

<null-call clause> ::=  

  | RETURNS NULL ON NULL INPUT  

  | CALLED ON NULL INPUT

<routine body> ::=  

  | <SQL routine body>  

  | <external body reference>

<SQL routine body> ::= <SQL procedure statement>

<external body reference> ::= EXTERNAL [ NAME <external routine name> ]  

<external routine name> ::= assembly-name [ .class-name [ .method-name ] ]  

<routine invocation> ::= routine-name <SQL argument list>

<SQL argument list> ::= ( [ <SQL argument> [ { , <SQL argument> }... ] ] )  

<SQL argument> ::=  

  | <value expression>  

  | <target specification>

```

SQL parameter declaration

- A parameter mode of IN is implicit if parameter mode is not specified.

- The OUT parameter has an initial value of null, and is used to output values from a procedure.
- The INOUT parameter is a combined IN and OUT parameter.
- Function definitions shall not specify a parameter mode.
- SQL parameter names shall conform to the rules for identifiers in NexusDB SQL.

Routine characteristics

- Language clause

NexusDB SQL supports user-defined procedures and functions written in SQL or hosted in a .NET assembly.

- LANGUAGE SQL defines an SQL routine, and is implicit if the clause is omitted. SQL routines shall specify an SQL routine body.
- LANGUAGE CLR defines a .NET CLR routine. CLR routines shall specify an external body reference, which is a reference to a class method in a .NET assembly.

Note: CLR routines are supported in the Enterprise Edition only.

- Deterministic characteristic

If neither DETERMINISTIC nor NOT DETERMINISTIC is specified, then NOT DETERMINISTIC is implicit. A DETERMINISTIC routine is guaranteed to produce the same result every time it is invoked with the same argument values, and may therefore be optimized by caching the result. NOT DETERMINISTIC routines are re-evaluated each time.

- SQL-data access indication

Defines whether or not the routine contains SQL statements, and if the routine is allowed to read or change SQL data.

- NO SQL can only be used with CLR routines, and means that the external routine contains no SQL statements. NO SQL is implicit for CLR routines if no SQL-data access indication is specified.
 - CONTAINS SQL is implicit for SQL routines if no SQL-data access indication is specified, and means that the routine may contain SQL statements, except any of the data statements.
 - READS SQL DATA means that the routine may contain SQL statements, including the SELECT statement, but is not allowed to specify any of the data change statements.
 - MODIFIES SQL DATA means that the routine may contain any of the SQL procedure statements, including the data-change statements.
 - A routine is only allowed to call other routines that specify the same or a higher SQL-data access level.
-
- Null-call clause

Is used to specify whether or not a function should be invoked if any of the arguments are null.

- CALLED ON NULL INPUT, which is implicit if the clause is not specified, means that the function is invoked regardless of null values in the arguments.

- RETURNS NULL ON NULL INPUT instructs the engine to return null as the function result if any of the arguments are null, without invoking the function itself.
- Procedure definitions shall not specify a null-call clause.

Routine body

- SQL Routines

The body of an SQL routine is defined using an SQL routine body, which is a single SQL procedure statement. The Compound statement can be used to specify multiple SQL statements inside a BEGIN..END block.

- CLR Routines

The body of a CLR routine is defined by providing an EXTERNAL body reference, which is a reference to a .NET assembly that has been registered on the NexusDB Server. Assemblies can be registered directly in the NexusDB Server GUI, or in SQL by using the CREATE ASSEMBLY statement.

- The NAME clause may be omitted in the rare case of the assembly name, the class name and the method name being the same as the routine name.
- assembly-name is the name used to register the assembly. The assembly can only be referenced by its owner and users with REFERENCES privilege on the assembly.
- class-name is the name of an existing class in the assembly. If the class has a multi-part name separated with periods, then the name must be delimited with double quotation marks, for example:

AssemblyName."Namespace.ClassName". The class name may be omitted if the name is the same as the assembly name.

- method-name is the name of a method defined in the specified class. The method name may be omitted if the name is the same as the class name.

Note: CLR routines are supported in the Enterprise Edition only.

Routine invocation

- User-defined procedures are invoked by the CALL statement followed by the routine name and the argument list.
- User-defined functions are invoked by the routine name followed by the argument list only.
- The number and types of arguments shall match the parameter declarations in the routine definition.
- The parentheses in the argument list shall be specified even if the routine has no parameters.
- Any arguments that represent an OUT parameter must be a target specification.

Examples

- 1) The following example calls a user-defined procedure passing two dynamic parameter values and one local SQL variable as arguments:

```
CALL setStudentName( :lastName, :firstName,  
studentName )
```

- 2) The following example calls a user-defined function without arguments:

```
getSystemTime()
```

- 3) The following example calls a user-defined function that takes two arguments:

```
getFullName( 'Tiger', 'Woods' )
```

Conformance

SQL:2003 standard - Core SQL

- Feature B128 "Routine language SQL"
- Feature T653 "SQL-schema statements in external routines"

NexusDB
extensions

- LANGUAGE CLR

21.3 SQL Procedure Statements

SQL Procedure Statements

Define all of the SQL statements that are allowed in SQL routines.

Syntax

```
<SQL procedure statement> ::=  
| <SQL schema statement>  
| <SQL data statement>  
| <SQL control statement>  
| <SQL transaction statement>  
| <SQL session statement>  
  
<SQL schema statement> ::=  
| <SQL schema definition statement>  
| <SQL schema manipulation statement>  
  
<SQL schema definition statement> ::=  
| <table definition>  
| <index definition>  
| <view definition>  
| <SQL-invoked routine>  
| <trigger definition>  
  
<SQL schema manipulation statement> ::=  
| <alter table statement>  
| <drop table statement>  
| <drop index statement>  
| <drop view statement>
```

```

| <drop routine statement>
| <drop trigger statement>

<SQL data statement> ::= 
| <select statement>
| <SQL data change statement>

<SQL data change statement> ::= 
| <insert statement>
| <update statement: searched>
| <delete statement: searched>

<SQL control statement> ::= 
| <call statement>
| <return statement>
| <compound statement>
| <SQL variable declaration>
| <assignment statement>
| <if statement>
| <iterate statement>
| <leave statement>
| <repeat statement>
| <while statement>
| <signal statement>
| <try statement>

<SQL transaction statement> ::= 
| <start transaction statement>
| <commit statement>
| <rollback statement>

<SQL session statement> ::= 
| <set passwords statement>

```

Conformance

SQL:2003 standard - Core SQL

- Feature T651 "SQL-schema statements in SQL routines"
- Feature P002-01 "Compound statement"
- Feature P002-04 "SQL variable declaration"
- Feature P002-05 "Assignment statement"
- Feature P002-07 "IF statement"
- Feature P002-08 "ITERATE statement"
- Feature P002-09 "LEAVE statement"
- Feature P002-11 "REPEAT statement"
- Feature P002-12 "WHILE statement"
- Feature P002-14 "SIGNAL statement"

- CREATE INDEX statement

NexusDB
extensions - DROP INDEX statement
 - TRY statement
 - SET PASSWORDS statement

22 SQL Statements

SQL Statements

An SQL statement is a sequence of characters that conforms to the syntax rules for the statement and the general rules for keywords and identifiers.

The execution of an SQL statement triggers database actions as implied by the content of the statement. Statements can be executed one by one or as a sequence of statements, either in the main execution block or in a Compound statement block. When SQL statements are executed as a sequence, each individual statement must be terminated with a semicolon.

SQL statements are categorized according to type of function:

- Schema Statements
- Data Statements
- Transaction Statements
- Session Statements
- Control Statements
- Special Statements

Prepared Statements
Statement switches

22.1 Prepared Statements

Prepared Statements

NexusDB preserves the execution context of an SQL statement if a prepared statement is requested by the client application, and stays prepared until explicitly unprepared. A prepared statement can therefore be reused without the overhead of creating a new execution context every time the statement is executed.

Usage

Prepared statements are typically used in combination with parameters, which is an efficient method for reusing data manipulation statements that are frequently used. For example, instead of executing a batch of UPDATE statements with explicit values, it is more efficient to prepare a parameterized UPDATE statement and provide the parameter values before each execution.

Examples

SELECT statement with parameters in the search condition

INSERT statement with parameters in the VALUES clause

UPDATE statement with parameters in the SET clause

DELETE statement with parameters in the WHERE clause

22.2 Statement Switches

Statement Switches

The following statement switches are NexusDB extensions that can be specified in a statement block to set execution characteristics:

Syntax

```
<blob switch> ::= #B { + | - }  
<index switch> ::= #I { + | - }  
<simplify switch> ::= #S { + | - }  
<log switch> ::= #L { + | - }  
<timeout switch> ::= #T integer-literal  
<feature switch> ::= #F feature-name { + | - }
```

Blob Switch

This switch controls the behavior when copying columns that contain BLOBs. In the default #B- state, the query engine only copies a link to the original BLOB in the result set which can save on both memory and execution time. The exception to this is if the source table is a temporary table, in which case the query engine will copy the full BLOB from the temporary table to the result set.

If switched on (#B+) the query engine will always copy the full BLOB to the result set. Since NexusDB allows for query results drawn on tables that are dropped within the same statement block, it is possible to write a SQL block so that BLOB links would refer to a table which had since been dropped, so understanding the implications of this setting is important.

```
SELECT * FROM BlobTable;  
DROP BlobTable;
```

In this case, the SELECT statement must be prefixed with #B+ to force BLOBs to be copied from BlobTable into the query result.

Index Switch

Indexing optimization turns the use of indexes on and off for the following statement. This switch defaults to #I+ (on), and is usually only turned off when trying to troubleshoot the query optimizer.

Simplify Switch

#S- turns query simplification off for the statement that follows the switch (default is on). Query simplification is a process where the structure of a parsed query is simplified. E.g. x BETWEEN y AND z would be simplified to x >= y AND x <= z .

As simplification occurs before the SQL statement is bound to the tables it works on, occasionally, simplification can convert a query into a less efficient alternative. For such cases, the #S- switch allows you turn simplification off. When logging is enabled (see #L+), the individual stages in the simplification process are logged.

Log Switch

#L+ turns on logging for the statement block as a whole and should be placed before any other noncomment line. When logging is enabled, (default is off) the query engine collects information about simplifications, index use, and query metrics, and builds a text report which is returned to the client with the query result and status. The report can be accessed through the Log property of the nxQuery component and can be used to analyze the query for opportunities to rephrase statements for better performance.

Timeout Switch

#T sets the query timeout period to the value specified by integer-literal in milliseconds.

Feature Switch

#F sets the named feature switch to the value specified by the switch value.

22.3 Schema Statements

Schema Statements

The schema statements, the part of the SQL language commonly known as DDL, Data Definition Language, are used to create, alter and drop schema (database) objects.

CREATE TABLE statement
CREATE INDEX statement
CREATE VIEW statement
CREATE TRIGGER
CREATE PROCEDURE statement
CREATE FUNCTION statement
CREATE ASSEMBLY statement

ALTER TABLE statement

DROP TABLE statement
DROP INDEX statement
DROP VIEW statement
DROP TRIGGER statement
DROP ROUTINE statement
DROP ASSEMBLY statement

22.3.1 CREATE TABLE statement

CREATE TABLE statement

Define a persistent base table, a global temporary table, or a local temporary table.

Syntax

```
<table definition> ::=  
    CREATE [ { GLOBAL | LOCAL } TEMPORARY ] TABLE [ schema-name. ] table-name  
    [ DESCRIPTION <character string literal> ]  
    [ { BLOCKSIZE | BLOCK } <unsigned integer> ]  
    [ { INITIALSIZE | INITIAL } <unsigned integer> ]  
    [ { GROWSIZE | GROW } <unsigned integer> ]  
    [ STORAGE [ ENGINE ] <character string literal> ]  
    [ ENCRYPTION [ ENGINE ] <character string literal> ]  
    [ ENCRYPT WITH <character string literal> ]  
    [ DEFAULT <character set clause> ]  
    [ DEFAULT <collation clause> ]  
    <table contents source>  
  
<collation clause> ::=  
    | COLLATION <character string literal> [ <compare flag>... ]  
    LOCALE <unsigned integer> [ <compare flag>... ]  
  
<table contents source> ::=  
    | ( <table element> [ { , <table element> }... ] )
```

```
<as subquery clause>

<table element> ::=  
| <column definition>  
| <table constraint definition>  
| <like clause>

<like clause> ::= LIKE [ schema-name. ] table-name

<as subquery clause> ::=  
[ ( column-name [ { , column-name }... ] ) ] AS <subquery>  
WITH { NO DATA | DATA }

<column definition> ::=  
column-name <data type>  
[ <default clause> ]  
[ <column constraint>... ]  
[ DESCRIPTION <character string literal> ]

<default clause> ::= DEFAULT <default option>

<default option> ::=  
| <literal>  
| NULL  
| EMPTY  
| NEWGUID  
| CURRENT_DATE  
| { CURRENT_TIME | LOCALTIME }  
| { CURRENT_TIMESTAMP | LOCALTIMESTAMP }  
| { USER | CURRENT_USER | SESSION_USER }

<column constraint definition> ::= [ <constraint name definition> ] <column constraint>

<constraint name definition> ::= CONSTRAINT constraint-name

<column constraint> ::=  
| NOT NULL  
| <unique constraint definition>  
| <references specification>  
| <check constraint definition>

<table constraint definition> ::= [ <constraint name definition> ] <table constraint>

<table constraint> ::=  
| <unique constraint definition>  
| <referential constraint definition>  
| <check constraint definition>

<unique constraint definition> ::= { UNIQUE | PRIMARY KEY } ( column-name [ { , column-name }... ] )  
>

<referential constraint definition> ::=
```

```
FOREIGN KEY ( column-name [ { , column-name }... ] )  
<references specification>  
  
<references specification> ::=  
    REFERENCES [ schema-name. ] table-name [ ( column-name [ { , column-name }... ] ) ]  
    [ <referential triggered action> ]  
  
<referential triggered action> ::=  
    | <update rule> [ <delete rule> ]  
    | <delete rule> [ <update rule> ]  
  
<update rule> ::= ON UPDATE <referential action>  
  
<delete rule> ::= ON DELETE <referential action>  
  
<referential action> ::=  
    | CASCADE  
    | SET NULL  
    | SET DEFAULT  
    | RESTRICT  
  
<check constraint definition> ::= CHECK ( <search condition> )
```

Usage

The CREATE TABLE statement creates a new table in the database with metadata according to the table definition.

- The schema and table names shall conform to the format rules for identifiers and schema-qualified names in NexusDB SQL. The table is created in the database specified by schema-name. The current database is implicit if a schema name is not specified.

Table attributes

The following table descriptor attributes are NexusDB extensions that can be used to tailor the table definition:

- DESCRIPTION is a free text attribute used to store a comment in the table descriptor.
- BLOCKSIZE sets the physical page size to one of the valid values 4, 8, 16, 32 or 64 kilobytes. The default value is 4. The page size is automatically adjusted to fit the total row size.
- INITIALSIZE sets the initial number of allocated pages. The default value is 4.

- GROWSIZE sets the number of new pages allocated when more space is needed. The default value is 1.
- STORAGE ENGINE specifies the name of the storage engine used for storing data in the table. The names of the built-in storage engines are 'Static', which stores data in fixed length, and 'Variable', which stores data in variable length. The default storage engine is 'Static'.
- ENCRYPTION ENGINE specifies the name of the encryption engine used to encrypt the table. The name of the built-in encryption engine is 'nx1xDefault'. NexusDB tables are not encrypted if an encryption engine is not explicitly specified.
- If ENCRYPT WITH is specified, then the encryption engine will use the password as part of the encryption algorithm. A password-encrypted table cannot be accessed unless the password has been added to the list of passwords available in the current session, see the SET PASSWORDS statement.
- DEFAULT CHARACTER SET or CODEPAGE can be used interchangeably to override the implicit codepage associated with the default collation for the table.
- DEFAULT COLLATION or LOCALE can be used interchangeably to set a default collation for the table. If the clause is omitted, then the default collation is byte order for character string columns and a locale-neutral collation for national character string columns.

Note: Table attributes must be specified in the order they appear in the syntax specification.

Table contents

The following elements can be used to define the table contents:

- Column definition
- Table constraint definition
- LIKE clause
- AS subquery clause

Column definition

A column definition consists of the following column descriptor attributes:

- Column name
 - Data type
 - Default clause (optional)
 - Column constraints (optional)
 - DESCRIPTION (optional)
-
- The column name shall conform to the format rules for identifiers in NexusDB SQL.
 - The data type can be any of the predefined data types in NexusDB SQL.
 - A character set can be specified for character string columns to override the default codepage associated with the implicit or explicit collation for the column.
 - A collation to be used in sort and comparison operations on the column, can be specified for character string and national character string columns. The default collation for the table is implicit if a collation is not explicitly defined.
 - The DEFAULT clause is used to assign a default value to the column when inserting new rows, with an implicit null value if the clause is omitted.
 - DESCRIPTION is a free text attribute that can be used to store comments in the column descriptor.

Column and table constraints

- The NOT NULL constraint can only be defined as a column constraint, and prohibits the null value to be assigned to the column in any row of the table.
- The UNIQUE constraint can be defined either as a column constraint, in case of an one-column key, or as a table constraint with one ore more columns specified in the column list. Rows that have the null value in any of the columns contained in the UNIQUE constraint are not included in the uniqueness check implied by the constraint, hence UNIQUE alone, without also specifying NOT NULL, permits multiple null values in a column.

- The PRIMARY KEY constraint is equal to the UNIQUE constraint, except that null values are not allowed in any column of the primary key. NOT NULL is implicit for primary key columns if not explicitly specified. Only one PRIMARY KEY constraint can be defined on a table.
- A FOREIGN KEY is a single column or a set of columns within a table that is referencing equivalent columns in another table through its values, a relationship called Referential Integrity. The purpose of the FOREIGN KEY constraint is to ensure that rows in the referencing table always have corresponding rows in the referenced table. The FOREIGN KEY constraint can be defined either as a column constraint using the REFERENCES clause only, in case of an one-column key, or as a table constraint with one ore more referencing columns specified in the column list.
- If the REFERENCES clause specifies a column list, then there shall be a one-to-one correspondence between the set of column names specified in the REFERENCES clause and the set of column names contained in a PRIMARY KEY or UNIQUE constraint of the referenced table. If the column list is not specified, then the referenced table shall have a PRIMARY KEY constraint containing a set of columns that matches the set of referencing columns.
- The data types of referencing and referenced columns shall be comparable.
- The REFERENCES clause may optionally specify an update rule, a delete rule, or both rules, each defining one of the following referential actions:
 - RESTRICT, the default referential action, aborts the current transaction with a foreign key violation error if the referential constraint is violated during the execution of an UPDATE or DELETE statement on the referenced table.
 - If CASCADE is specified, then data changes in the referenced table are replicated to the corresponding rows in the referencing table.
 - If SET NULL is specified, then referencing columns are set to null when a corresponding row in the referenced table is deleted or the referenced column is changed.
 - If SET DEFAULT is specified, then referencing columns are set to their default value when a corresponding row in the referenced table is deleted or the referenced column is changed.

- The CHECK constraint is a general purpose constraint that can be defined either as a column constraint or a table constraint. The search condition can be any valid predicate. Columns of the table defining the constraint are referenced by the column names, while columns of other tables can be accessed by using a subquery. The CHECK constraint is satisfied if the search condition is not False for any row in the table, as opposed to having to be True for every row in the table.
- A user-defined constraint name can be specified for both column and table constraints, except the NOT NULL constraint. The constraint name shall conform to the format rules for identifiers in NexusDB SQL. If the constraint name is not specified, then the constraint being defined is given a system-generated name. The main advantage of using named constraints, is that this feature makes constraint management a lot easier.
- The implicit constraint mode is INITIALLY IMMEDIATE NOT DEFERRABLE.

Note: The ALTER TABLE statement is used for constraint management.

LIKE clause

- The LIKE clause works like a shorthand for manually defining new columns based on the columns of an existing table, and can be mixed with other column definitions and table constraints.
- The LIKE clause defines a new column for each column in the source table and copies the following column descriptor attributes to the new table:
 - Column name
 - Data type (including explicit character set and collation for character string columns)
 - DEFAULT value
 - NOT NULL constraint
- Column constraints other than the NOT NULL constraint are not copied to the new table.

AS subquery clause

- The AS subquery clause creates a new table based on one or more existing tables using a subquery.
- The optional column-name list is used to rename the columns specified in the select list of the subquery. If the column-name list is specified, then the number of columns shall be the same as the degree of the subquery.
- For each column in the subquery result set, the following column descriptor attributes are copied to the new table:
 - Column name
 - Data type (including explicit character set and collation for character string columns)
- No table constraints or indexes are copied to the new table.
- If WITH DATA is specified, then the new table is populated with the rows from the subquery result set.

Temporary tables

By supporting temporary tables, NexusDB provides an efficient concept for storing sets of data temporarily, without worrying about conflicts with persistent data in the database. Temporary tables are useful in many situations, for example to create fast lookup tables locally on the client, to combine data from multiple tables, and to store intermediate result sets that can be accessed and further processed by other statements.

NexusDB creates temporary tables in memory. Once created, temporary tables can be manipulated and queried just like ordinary tables, except that the visibility is limited to the scope of the temporary table type. NexusDB SQL supports three types of temporary tables with different scopes and duration:

- A **context-local** temporary table is defined by prefixing the table name with a single pound symbol (#table-name). Context-local temporary tables are visible only in the execution block where they are created, and are automatically dropped when the execution block terminates. Context-local temporary tables cannot be restructured if they contain data.
- A **session-local** temporary table is defined by prefixing the table name with two pound symbols (##table-name). Session-local temporary tables are visible to all processes that are executed in the context of the session and database object that created them, and are automatically dropped when the session terminates.

- A **global** temporary table is defined by prefixing the table name with three pound symbols (###table-name). Global temporary tables are visible to all sessions accessing the server, and stay in memory on the server until explicitly dropped or the server shuts down.
- The V1 syntax of defining a global temporary table by enclosing the table name in angle brackets (<table-name>) is still supported, but has been deprecated and may be removed in a future version of NexusDB SQL.
- NexusDB SQL **requires** that the names of temporary tables are prefixed with pound symbols. The optional GLOBAL | LOCAL TEMPORARY TABLE syntax can be used when defining temporary tables, but is only supported for clarity and syntax compatibility with SQL:2003.

Tip: The AS subquery clause and the SELECT INTO statement are both convenient ways of creating temporary tables based on existing tables.

Examples

- 1) The following example shows a table definition with column constraints:

```
CREATE TABLE customers
/*
```

We're defining a default collation for this table.
The default collation is used in sort and
comparison operations on character string
columns that don't explicitly specify a collation on
the column level.

```
*/
DEFAULT LOCALE 1033
(
customerID AUTOINC NOT NULL PRIMARY
KEY,
companyName VARCHAR(40),
address1 VARCHAR(30),
address2 VARCHAR(30),
city VARCHAR(20),
postalCode VARCHAR(10),
country VARCHAR(20)
)
```

- 2) The following example shows a table definition with default column values and column constraints:

```

CREATE TABLE orders
DESCRIPTION 'Master sales orders table'
(
    orderID AUTOINC NOT NULL PRIMARY KEY,
    customerID INTEGER NOT NULL REFERENCES
        customers ( customerID ),
    orderDate DATE DEFAULT CURRENT_DATE,
    shippingDate DATE,
    freight DECIMAL(0,2) DEFAULT 0,
    orderTotal DECIMAL(0,2) DEFAULT 0,
    amountPaid DECIMAL(0,2) DEFAULT 0
)

```

- 3) The following example shows a table definition with named column and table constraints:

```

CREATE TABLE order_details
(
    orderID INTEGER NOT NULL CONSTRAINT
        uc_orderID UNIQUE,
    productID AUTOINC NOT NULL,
    unitPrice DECIMAL(0,4) DEFAULT 0,
    quantity SMALLINT DEFAULT 0,
    discount FLOAT DEFAULT 0,
    CONSTRAINT pk_orderID_productID PRIMARY
    KEY ( orderID, productID ),
    CONSTRAINT fk_orders_orderID FOREIGN KEY
    ( orderID ) REFERENCES orders ( orderID )
    ON UPDATE CASCADE ON DELETE
    RESTRICT,
    CONSTRAINT ck_discount CHECK ( discount >=
    0 )
)

```

- 4) The following example illustrates how the LIKE clause is used in a table definition:

/*

The LOCAL TEMPORARY keywords have been included for syntax clarity.

What's actually defining the table as a session-local temporary table in NexusDB are the two pound signs prefixing the table name.

```

*/  

CREATE LOCAL TEMPORARY TABLE  

##students  

(  

    LIKE students,  

    dob DATE,  

    notes CLOB LOCALE 1033, -- A collation is  

    explicitly specified for this column  

    primary key ( studentID )  

)
```

- 5) The following example illustrates how the AS subquery clause is used in a table definition:

```

/*  

The three pound signs prefixing the table name  

defines the table as a global temporary table.  

The GLOBAL TEMPORARY keywords have been  

omitted, but could have been specified for syntax  

clarity.  

*/  

CREATE TABLE ###students_list  

( studentID, name, city, gender ) AS  

( SELECT studentID, studentName, city, gender  

FROM students )  

WITH DATA
```

Conformance

SQL:2003 standard - Core SQL

- Feature F692 "Enhanced collation support"
- Feature T171 "LIKE clause in table definition"
- Feature T172 "AS subquery clause in table definition"
- Feature F531 "Temporary tables"
- Rows of temporary tables are implicitly preserved
- Feature F491 "Constraint management"
- Feature T591 "UNIQUE constraints of possibly null columns"
- Feature F191 "Referential delete actions"
- Feature F701 "Referential update actions"
- Feature T191 "Referential action RESTRICT"
- Feature T201 "Comparable data types for referential constraints"

	<ul style="list-style-type: none"> - Feature F671, "Subqueries in CHECK constraints" - Feature F672 "Retrospective check constraints"
NexusDB extensions	<ul style="list-style-type: none"> - DESCRIPTION - BLOCKSIZE BLOCK - INITIALSIZE INITIAL - GROWSIZE GROW - STORAGE [ENGINE] - ENCRYPTION [ENGINE] - ENCRYPT WITH - DEFAULT CHARACTER SET - DEFAULT COLLATION - NEWGUID - #table-name defines a context-local temporary table - ##table-name defines a session-local temporary table - ###table-name defines a global temporary table

22.3.2 CREATE INDEX statement

CREATE INDEX statement

Define a primary or secondary index.

Syntax

```

<index definition> ::= 
    CREATE [ UNIQUE ] INDEX index-name ON [ schema-name. ] table-name
    ( <index element> [ { , <index element> }... ] )

<index element> ::= column-name <sort options>

<sort options> ::= 
    [ <collate clause> ]
    [ IGNORE CASE ]
    [ <sort direction> ]
    [ <null ordering> ]

<sort direction> ::= { ASC | DESC }

<null ordering> ::= NULLS { FIRST | LAST }

```

Usage

The CREATE INDEX statement creates a compiled index based on the columns and sort options of the specified index elements. Indexes are used to optimize search and filter conditions, and it is generally recommended to define indexes on all columns that are used as keys in JOIN and WHERE clause criteria. The speed of grouping and ordering operations will also improve greatly if the columns in the GROUP BY and ORDER BY clauses are supported by matching indexes.

Notes

- The index, schema and table names shall conform to the format rules for identifiers and schema-qualified names in NexusDB SQL. The current database is implicit if a schema name is not specified.
- If UNIQUE is specified, then an UNIQUE constraint is implicitly defined for the table.
- The collate clause is used to specify a Windows OS collation for character string and national character string index columns. If the clause is omitted, then the explicit or implicit collation in the column definition is used for the sort order of the index key.
- IGNORE CASE makes the sort order case-insensitive.
- ASC is implicit if a sort direction is not specified. Specify DESC to sort the index key in descending order.
- NULLS FIRST sorts null values before all non-null values in the set, and is implicit if the null ordering clause is not specified. Specify NULLS LAST to have null values sorted at the bottom of the set.

Examples

- 1) The following example creates a compound unique index on the sections table:

```
CREATE UNIQUE INDEX ix_courseID_teacherID  
ON sections ( courseID, teacherID )
```

- 2) The following example creates a case-insensitive index on the students table with an explicit collation for the studentName column:

```
CREATE INDEX ix_studentName ON students  
( studentName locale 1033 IGNORE CASE )
```

Conformance

NexusDB extensions - CREATE INDEX statement

22.3.3 CREATE VIEW statement

CREATE VIEW statement

Define a persistent viewed table.

Syntax

```
<view definition> ::=  
    CREATE VIEW [ schema-name. ] view-name <view specification> AS <query expression>  
  
<view specification> ::= [ ( column-name [ { , column-name }... ] ) ]
```

Usage

The CREATE VIEW statements creates a persistent view in the database based on the query expression. Views are a useful mechanism for defining complex queries once and to use them repeatedly in SQL statements. Views can also be used to enhance database security, for example by exposing only parts of a base table to other users.

Notes

- The schema and view names shall conform to the format rules for identifiers and schema-qualified names in NexusDB SQL. The view is created in the database specified by schema-name. The current database is implicit if a schema name is not specified.
- The optional view specification can be used to rename the columns processed by the query expression. If view specification is specified, then the degree shall be equal to the degree of the query expression.
- All table references in the subquery shall identify persistent base tables or views.
- Only the metadata about the view is stored in the database. The data itself is processed from the base tables when the view is referenced in SQL statements.
- Once created, views can be queried just like ordinary base tables.
- Views are read-only.

Examples

- 1) The following example creates a view that exposes only some of the columns in the Teachers base table:

```
CREATE VIEW teachers_view AS (
    SELECT teacherID, teacherName, phone
    FROM teachers
)
```

- 2) The following example references teachers_view in a SELECT statement:

```
SELECT courseID, teacherID, teacherName
FROM sections s
JOIN teachers_view t on t.teacherID =
s.teacherID
ORDER BY courseID
```

Conformance

SQL:2003 standard - Core SQL
- Views are read-only

22.3.4 CREATE TRIGGER statement

CREATE TRIGGER statement

Define a trigger.

Syntax

```
<trigger definition> ::=  
    CREATE [OR ALTER] TRIGGER [ schema-name. ] trigger-name  
    { BEFORE | AFTER } <trigger event> [ { , <trigger event> }... ]  
    ON [ schema-name. ] table-name [ REFERENCING <transition variable>... ]  
    [ DESCRIPTION <character string literal> ]  
    [ AS ] <triggered action>
```

```

<trigger event> ::=  

  | INSERT  

  | DELETE  

  UPDATE [ OF <trigger column list> ]  
  

<trigger column list> ::= column-name [ { , column-name }... ]  
  

<transition variable> ::=  

  | OLD [ ROW ] [ AS ] correlation-name  

  NEW [ ROW ] [ AS ] correlation-name  
  

<triggered action> ::=  

  [ FOR EACH { ROW } ]  

  [ WHEN ( <search condition> ) ]  

  <triggered SQL statement>  
  

<triggered SQL statement> ::= <SQL procedure statement>

```

Usage

The CREATE OR ALTER TRIGGER statement creates a trigger that is stored in the database and associated with the subject table identified by table-name.

When using CREATE OR ALTER, if the trigger already exists, it is being replaced in-place with the new definition instead of raising an error.

If there is an error with the new routine, the existing one remains unchanged.

This prevents the situation where a sequence of
 DROP ... [IF EXISTS]
 CREATE ...

executes the DROP but then fails the CREATE statement.

Notes

- The schema and trigger names shall conform to the format rules for identifiers and schema-qualified names in NexusDB SQL. The trigger is created in the database specified by schema-name, which must be the same database as where the associated table is stored. The current database is implicit if a schema name is not specified.
- The subject table of the trigger definition shall be a persistent base table.
- The BEFORE and AFTER keywords define whether the trigger is fired immediately before or immediately after the effects of the triggering SQL statement are applied to the table.

- The trigger event specifies which data-change statements that will cause the trigger to fire. The trigger event can specify INSERT, DELETE or UPDATE, or any combination of the event types separated by a comma. If the optional OF clause is specified for an UPDATE event, then the event is only triggered when the specified columns are updated.
- The REFERENCING clause can be specified to give correlation names to the OLD and NEW transition variables. OLD is a reference to the original row values that existed before an UPDATE or DELETE operation. NEW is a reference to the row values that will exist after an INSERT or UPDATE operation. The NEW variable can be used in a BEFORE trigger to assign column values to the new or modified row, else both variables are read-only.
- DESCRIPTION is a free text attribute used to store a comment in the trigger descriptor.
- FOR EACH ROW, which is implicit if not specified, means that the trigger is fired for each row being processed by a data-change statement.
- The optional WHEN clause is used to specify a condition that must evaluate to true before the triggered SQL statement is executed.
- The triggered SQL statement can be any valid SQL procedure statement, including the Compound statement that allows multiple SQL statements to be specified.
- The special INSERTING, UPDATING and DELETING predicates can be referenced in the WHEN clause or the triggered SQL statement to check the current trigger event type.

Examples

- 1) The following example creates a trigger that deletes related rows in the order_details table after a row in the orders table has been deleted:

```
CREATE TRIGGER orders_after_delete
AFTER DELETE ON orders
WHEN ( DELETING )
DELETE FROM order_details WHERE orderID =
OLD.orderID;
```

- 2) The following example creates a multi-event trigger that stores information about changes to the enrolls table in the enrolls_log table:

```
CREATE OR ALTER TRIGGER enrolls_changes
AFTER INSERT, DELETE, UPDATE ON enrolls
```

```

REFERENCING OLD AS o NEW AS n
BEGIN

IF INSERTING THEN

INSERT INTO enrolls_log ( action, n1, n2, n3, n4,
stamp )
VALUES ( 'Insert', n.courseID, n.sectionID,
n.studentID, n.grade, CURRENT_TIMESTAMP );

ELSEIF UPDATING THEN

INSERT INTO enrolls_log ( action, o1, n1, o2, n2,
o3, n3, o4, n4, stamp )
VALUES ('Update', o.courseID, n.courseID,
o.sectionID, n.sectionID,
o.studentID, n.studentID, o.grade, n.grade,
CURRENT_TIMESTAMP );

ELSEIF DELETING THEN

INSERT INTO enrolls_log ( action, o1, o2, o3, o4,
stamp )
VALUES ( 'Delete', o.courseID, o.sectionID,
o.studentID, o.grade, CURRENT_TIMESTAMP );

END IF;

END

```

Conformance

SQL:2003 standard - Feature T211 "Basic trigger capability"

NexusDB extensions	- DESCRIPTION - More than one trigger event can be specified - AS clause before the triggered action specification.
-----------------------	---

22.3.5 CREATE PROCEDURE statement

CREATE PROCEDURE statement

Define an SQL-invoked procedure.

Syntax

```

<user-defined procedure> ::= 
    CREATE [OR ALTER] PROCEDURE [ schema-name. ] procedure-name
    <SQL parameter declaration list>
    <routine characteristics>
    [ DESCRIPTION <character string literal> ]
    [ AS ] <routine body>

```

Usage

The CREATE OR ALTER PROCEDURE statement creates or changes a user-defined procedure that is stored in the database.

When using CREATE OR ALTER, if the stored routine already exists, it is being replaced in-place with the new definition instead of raising an error.

If there is an error with the new routine, the existing one remains unchanged.

This prevents the situation where a sequence of
 DROP ... [IF EXISTS]
 CREATE ...

executes the DROP but then fails the CREATE statement.

Notes

- The schema and procedure names shall conform to the format rules for identifiers and schema-qualified names in NexusDB SQL. The procedure is created in the database specified by schema-name. The current database is implicit if a schema name is not specified.
- SQL parameter declaration
 - Procedure parameters can be specified with a parameter mode of IN, OUT or INOUT. In is implicit if a parameter mode is not specified.
 - SQL parameter names shall conform to the rules for identifiers in NexusDB SQL.
- Routine characteristics
 - LANGUAGE SQL is implicit if not specified. LANGUAGE CLR defines a CLR routine.

- NOT DETERMINISTIC is implicit if neither DETERMINISTIC nor NOT DETERMINISTIC is specified.
 - If an SQL-data access indication is not specified, then CONTAINS SQL is implicit for SQL routines and NO SQL is implicit for CLR routines. The procedure cannot be called by another routine with a lower SQL-data access level.
 - The null-call clause shall not be specified in a procedure definition.
-
- DESCRIPTION is a free text attribute used to store a comment in the procedure descriptor.
 - Routine body

SQL procedures shall specify a SQL routine body, and CLR procedures shall specify an external body reference.

An SQL procedure can return a cursor by specifying a SELECT statement as the last statement in the routine body.

Examples

- 1) The following example creates a procedure that inserts a new row in the courses table with values passed through four parameters:

```
CREATE OR ALTER PROCEDURE addCourse
(
    // Parameter declarations with implicit IN mode
    p_courseID SMALLINT,
    p_courseName CHAR(20),
    p_department CHAR(20),
    p_numCredits TINYINT
)
MODIFIES SQL DATA -- We need write-access to
update the courses table
// We prefer to use a compound statement in the
routine body, even with a single statement
```

```
BEGIN  
  
    INSERT INTO courses  
    VALUES ( p_courseID, p_courseName,  
             p_department, p_numCredits );  
  
END
```

- 2) The following example creates a procedure that raises all teacher salaries by a given percent and uses an OUT parameter to signal whether or not the transaction succeeded:

```
CREATE OR ALTER PROCEDURE raiseSalaries  
(  
  
    IN percent FLOAT,  
    OUT done BOOLEAN  
  
)  
/*  
LANGUAGE SQL is implicit  
NOT DETERMINISTIC is implicit  
*/  
MODIFIES SQL DATA -- We need write-access to  
update the teachers table  
// Multiple statements in the routine body must  
appear inside a compound statement  
BEGIN  
  
    DECLARE rate FLOAT;  
    SET rate = 1 + ( percent / 100 );  
    START TRANSACTION;  
    TRY  
  
        UPDATE teachers SET salary = ROUND( salary *  
            rate );  
        COMMIT;  
        SET done = TRUE;  
  
    CATCH ( TRUE )  
  
        ROLLBACK;  
        SET done = FALSE;  
  
    END;  
  
END
```

- 3) The following example creates a procedure that returns a cursor derived from the students table:

```

CREATE PROCEDURE studentNames
( IN isFullName BOOLEAN )
READS SQL DATA -- We need read-access to
execute the SELECT statement
BEGIN

IF isFullName THEN

SELECT lastName, firstName FROM students;

ELSE

SELECT firstName FROM students;

END IF;

END

```

Conformance

SQL:2003 standard - Core SQL
- Feature B128 "Routine language SQL"

NexusDB extensions	- DESCRIPTION - AS clause before the routine body
-----------------------	--

22.3.6 CREATE FUNCTION statement

CREATE FUNCTION statement

Define an SQL-invoked function.

Syntax

```

<user-defined function> ::=

CREATE [OR ALTER] FUNCTION [ schema-name. ] function-name
<SQL parameter declaration list>
RETURNS { <data type> | TABLE }
<routine characteristics>
[ DESCRIPTION <character string literal> ]
[ AS ] <routine body>

```

Usage

The CREATE FUNCTION statement creates a user-defined function that is stored in the database.

When using CREATE OR ALTER, if the stored function already exists, it is being replaced in-place with the new definition instead of raising an error.

If there is an error with the new routine, the existing one remains unchanged.

This prevents the situation where a sequence of

DROP ... [IF EXISTS]

CREATE ...

executes the DROP but then fails the CREATE statement.

Notes

- The schema and procedure names shall conform to the format rules for identifiers and schema-qualified names in NexusDB SQL. The procedure is created in the database specified by schema-name. The current database is implicit if a schema name is not specified.
- SQL parameter declaration
 - A parameter mode shall not be specified. All function parameters are implicitly IN parameters.
 - SQL parameter names shall conform to the rules for identifiers in NexusDB SQL.
- The RETURNS clause can either specify the data type of a scalar-valued function result, or TABLE to define a table-valued function that returns a cursor.
- Routine characteristics
 - LANGUAGE SQL is implicit if not specified. LANGUAGE CLR defines a CLR routine.
 - NOT DETERMINISTIC is implicit if neither DETERMINISTIC nor NOT DETERMINISTIC is specified.
 - If an SQL-data access indication is not specified, then CONTAINS SQL is implicit for SQL routines and NO SQL is implicit for CLR routines. The

function cannot be called by another routine with a lower SQL-data access level.

- CALLED ON NULL INPUT is implicit if the null-call clause is not specified.
- DESCRIPTION is a free text attribute used to a store comment in the function descriptor.
- Routine body

SQL functions shall specify a SQL routine body, and CLR functions shall specify an external body reference.

The RETURN statement is required in an SQL routine body to specify the result of the function.

The RETURN clause of a table-valued SQL function shall specify a SELECT statement.

Examples

- 1) The following example creates a function that concatenates two character string values, separated by a space, to form a full name:

```
CREATE FUNCTION getFullName
(
    firstName VARCHAR(30),
    lastName VARCHAR(30)

)
RETURNS VARCHAR(61)
/*
LANGUAGE SQL is implicit
NOT DETERMINISTIC is implicit
CONTAINS SQL is implicit
*/
RETURNS NULL ON NULL INPUT -- We don't
invoke the function if any of the arguments are null

// Multiple statements in the routine body must
// appear inside a compound statement
BEGIN

    DECLARE name VARCHAR(61);
    SET name = firstName || ' ' || lastName;
```

```
    RETURN name;
```

```
END
```

- 2) The following example creates a function that returns the system date and time:

```
CREATE OR ALTER FUNCTION getSystemTime ()  
RETURNS TIMESTAMP  
// A single statement in the routine body doesn't  
need to appear inside a compound statement  
RETURN CURRENT_TIMESTAMP;
```

- 3) The following example creates a function that returns a cursor derived from the students table:

```
CREATE FUNCTION getStudents ()  
RETURNS TABLE  
READS SQL DATA -- We need read-access to  
execute the SELECT statement  
// We prefer to use a compound statement in the  
routine body, even with a single statement  
BEGIN  
  
RETURN SELECT * FROM students;  
  
END
```

Conformance

SQL:2003 standard - Core SQL

- Feature B128 "Routine language SQL"
- Feature T326 "Table functions"

NexusDB extensions - DESCRIPTION
 - AS clause before the routine body

22.3.7 CREATE ASSEMBLY statement

CREATE ASSEMBLY statement

Define an assembly registration.

Syntax

```
<assembly definition> ::=  
    CREATE [ SCRIPT | NET ] ASSEMBLY assembly-name  
        [ AUTHORIZATION owner-name ]  
        FROM <assembly specifier>  
  
<assembly specifier> ::=  
    | <quote> [ \\computer-name\ ] share-name\ [ path\ ] manifest-file-name <quote>  
        <quote> [ local_path\ ] manifest-file-name <quote>
```

Usage

The CREATE ASSEMBLY statement registers a .NET assembly on the NexusDB Server. After registration, the methods contained in the assembly can be referenced by user-defined CLR procedures and functions.

Notes

- The assembly name shall conform to the format rules for identifiers in NexusDB SQL. The name must be different from other registered assembly names, and is the name used by the server to identify the assembly when invoking user-defined CLR routines.
- The AUTHORIZATION clause specifies the name of a valid user or role. If the clause is not specified, then ownership is given to the current user.
- The FROM clause specifies the local path or network location where the assembly is located, and the manifest file name that corresponds to the assembly.

Note: The CREATE ASSEMBLY and DROP ASSEMBLY statements are supported in the Enterprise Edition only.

Examples

- 1) The following example registers an assembly located in a local folder on the NexusDB Server computer:

```
CREATE ASSEMBLY dotNET_routines  
FROM 'C:\Program  
Files\NexusDB\Assemblies\DBFuncs.dll'
```

Conformance

NexusDB extensions - CREATE ASSEMBLY statement

22.3.8 ALTER TABLE statement

ALTER TABLE statement

Change the definition of a table.

Syntax

```

<alter table statement> ::= 
    ALTER TABLE [ schema-name. ] table-name <alter table action>

<alter table action> ::= 
    | SET DESCRIPTION <character string literal>
    | <add column definition>
    | <alter column definition>
    | <drop column definition>
    | <add table constraint definition>
    | <drop table constraint definition>

<add column definition> ::= ADD [ COLUMN ] <column definition>

<alter column definition> ::= 
    ALTER [ COLUMN ] column-name <alter column action>

<alter column action> ::= 
    | SET <default clause>
    | DROP DEFAULT
    | ADD CONSTRAINT NOT NULL
    | DROP CONSTRAINT NOT NULL
    | SET DESCRIPTION <character string literal>
    | CAST [AS] <data type> [ WITH BESTFIT ]

<drop column definition> ::= DROP [ COLUMN ] column-name [ RESTRICT ]

<add table constraint definition> ::= ADD <table constraint definition>

<drop table constraint definition> ::= DROP CONSTRAINT [ IF EXISTS ] constraint-name [ RESTRICT ]

```

Usage

The ALTER TABLE statement is used to change the definition of an existing table in the database, and requires exclusive access to that table.

Notes

- The current database is implicit if a schema name is not specified.
- Use the SET DESCRIPTION clause to change the content of the table description attribute.
- Use the ADD COLUMN clause to add a new column to the table.
- Use the ALTER COLUMN clause to change one of the following column attributes:
 - DEFAULT value
 - NOT NULL constraint
 - DESCRIPTION
- Use the DROP COLUMN clause to remove a column from the table. RESTRICT, which is implicit if not explicitly specified, prevents the column from being removed if the column is referenced by other database objects.
- Use the ADD table constraint clause to add a PRIMARY KEY, UNIQUE or FOREIGN KEY constraint to the table.
- Use the DROP CONSTRAINT clause to remove a table constraint.
 - The IF EXISTS clause can be specified to avoid an exception when attempting to delete a constraint that does not exist.
 - RESTRICT, which is implicit if not explicitly specified, prevents a PRIMARY KEY or UNIQUE constraint from being removed if the constraint is referenced by a FOREIGN KEY constraint in another table.
- The ALTER TABLE statement requires exclusive access to the table being altered.

Examples

- 1) The following example adds a new column to the students table:

```
ALTER TABLE students
ADD COLUMN picture IMAGE
```

- 2) The following example adds a NOT NULL constraint to the lastName column of the students table:

```
ALTER TABLE students
ALTER COLUMN lastName
ADD CONSTRAINT NOT NULL
```

- 3) The following example removes the picture column from the students table:

```
ALTER TABLE students
DROP COLUMN picture
```

- 4) The following example adds a FOREIGN KEY constraint to the enrolls table:

```
ALTER TABLE enrolls
ADD CONSTRAINT fk_students_studentID
FOREIGN KEY ( studentID ) REFERENCES
students ( studentID )
```

- 5) The following example removes the constraint named fk_students_studentID from the enrolls table:

```
ALTER TABLE enrolls
DROP CONSTRAINT fk_students_studentID
```

Conformance

SQL:2003 standard - Core SQL
- Feature F381 "Extended schema manipulation"

NexusDB extensions	- ADD CONSTRAINT NOT NULL - DROP CONSTRAINT NOT NULL - SET DESCRIPTION - IF EXISTS clause
-----------------------	--

22.3.9 DROP TABLE statement

DROP TABLE statement

Delete a table.

Syntax

```
<drop table statement> ::=  
    DROP TABLE [ IF EXISTS ] [ schema-name. ] table-name [ RESTRICT ]
```

Usage

The DROP TABLE statement deletes the table specified by table-name from the database.

Notes

- The current database is implicit if a schema name is not specified.
- The IF EXISTS clause can be specified to avoid an exception when attempting to delete a table that does not exist in the database.
- Dropping a table automatically deletes all indexes and triggers owned by the table.
- RESTRICT, which is implicit and the only supported drop behavior, prevents the table from being deleted if the table is referenced by other tables, views, triggers, procedures or functions.
- The DROP TABLE statement requires exclusive access to the table being deleted.

Examples

- 1) The following example deletes the ##students table:

```
DROP TABLE IF EXISTS ##students
```

Conformance

SQL:2003 standard - Core SQL

NexusDB - IF EXISTS clause
extensions

22.3.10 DROP INDEX statement

DROP INDEX statement

Delete an index.

Syntax

```
<drop index statement> ::=  
    DROP INDEX [ IF EXISTS ] [ schema-name. ] table-name.index-name
```

Usage

The DROP INDEX statement deletes the index specified by index-name from the table.

Notes

- The current database is implicit if a schema name is not specified.
- The IF EXISTS clause can be specified to avoid an exception when attempting to delete an index that does not exist.
- Deleting a UNIQUE index will automatically remove the associated table constraint.
- The DROP INDEX statement requires exclusive access to the table on which the index being deleted is defined.

Examples

- 1) The following example deletes the ix_studentName index from the students table:

```
DROP INDEX IF EXISTS students.ix_studentName
```

Conformance

NexusDB extensions	- DROP INDEX statement - IF EXISTS clause
--------------------	--

22.3.11 DROP VIEW statement

DROP VIEW statement

Delete a view.

Syntax

```
<drop view statement> ::=  
    DROP VIEW [ IF EXISTS ] [ schema-name. ] view-name [ RESTRICT ]
```

Usage

The DROP VIEW statement deletes the view specified by view-name from the database.

Notes

- The current database is implicit if a schema name is not specified.
- The IF EXISTS clause can be specified to avoid an exception when attempting to delete a view that does not exist in the database.
- RESTRICT, which is implicit if not explicitly specified, prevents the view from being deleted if the view is referenced by other database objects.
- The DROP VIEW statement requires exclusive access to the view being deleted.

Examples

- 1) The following example deletes the teachers_view view:

```
DROP VIEW IF EXISTS teachers_view
```

Conformance

SQL:2003 standard - Core SQL

NexusDB - IF EXISTS clause
extensions

22.3.12 DROP TRIGGER statement

DROP TRIGGER statement

Delete a trigger.

Syntax

```
<drop trigger statement> ::=  
    DROP TRIGGER [ IF EXISTS ] [ schema-name. ] trigger-name
```

Usage

The DROP TRIGGER statement deletes the trigger specified by trigger-name from the database.

Notes

- The current database is implicit if a schema name is not specified.
- The IF EXISTS clause can be specified to avoid an exception when attempting to delete a trigger that does not exist in the database.
- The DROP TRIGGER statement requires exclusive access to the subject table of the trigger definition.

Examples

- 1) The following example deletes the enrolls_changes trigger:

```
DROP TRIGGER IF EXISTS enrolls_changes
```

Conformance

SQL:2003 standard - Core SQL

- Feature T211 "Basic trigger capability"

NexusDB
extensions

- IF EXISTS clause

22.3.13 DROP ROUTINE statement

DROP ROUTINE statement

Delete a user-defined procedure or function.

Syntax

<drop routine statement> ::=

 DROP <routine type> [IF EXISTS] [schema-name.] routine-name [RESTRICT]

<routine type> ::=

 | ROUTINE
 | FUNCTION
 | PROCEDURE

Usage

The DROP ROUTINE statement deletes the procedure or function specified by routine-name from the database.

Notes

- The current database is implicit if a schema name is not specified.
- ROUTINE can be specified to indicate that the routine type can be either procedure or function.

- If PROCEDURE or FUNCTION is specified, then the routine being deleted must be the specified type.
- The IF EXISTS clause can be specified to avoid an exception when attempting to delete a routine that does not exist in the database.
- RESTRICT, which is implicit if not explicitly specified, prevents the routine from being deleted if the routine is referenced by other database objects.
- The DROP ROUTINE statement requires exclusive access to the routine being deleted.

Examples

- 1) The following example deletes the getStudents function:

```
DROP FUNCTION IF EXISTS getStudents
```

- 2) The following example deletes the getStudents function using the ROUTINE keyword:

```
DROP ROUTINE IF EXISTS getStudents
```

- 3) The following example deletes the studentNames procedure:

```
DROP PROCEDURE IF EXISTS studentNames
```

Conformance

SQL:2003 standard - Core SQL

NexusDB - IF EXISTS clause
extensions

22.3.14 DROP ASSEMBLY statement

DROP ASSEMBLY statement

Remove an assembly registration.

Syntax

```
<drop assembly statement> ::=  
    DROP ASSEMBLY [ IF EXISTS ] assembly-name
```

Usage

The DROP ASSEMBLY statement removes the specified assembly registration on the NexusDB Server.

Notes

- The IF EXISTS clause can be specified to avoid an exception when attempting to delete an assembly that is not registered on the NexusDB Server.
- The DROP ASSEMBLY statement returns an error if the assembly is referenced by a CLR routine.

Note: The CREATE ASSEMBLY and DROP ASSEMBLY statements are supported in the Enterprise Edition only.

Examples

- 1) The following example removes the registration of DBFuncs.dll which has been registered with the name dotNET_routines:

```
DROP ASSEMBLY IF EXISTS dotNET_routines
```

Conformance

NexusDB extensions	- DROP ASSEMBLY statement - IF EXISTS clause
-----------------------	---

22.4 Data Statements

Data Statements

The data statements, the part of the SQL language commonly known as DML, Data Manipulation Language, are used to retrieve, insert, update and delete data in the database.

- SELECT statement
- INSERT statement
- UPDATE statement
- DELETE statement

22.4.1 SELECT statement

SELECT statement

Specify a cursor that retrieves a selection of rows from one or more tables.

While the full syntax is rather complex, the main syntactical elements can be simplified as:

```
SELECT select-list
[ INTO table-name ]
FROM table-reference
[ WHERE search-condition ]
[ GROUP BY grouping-element ]
[ HAVING search-condition ]
[ ORDER BY sort-specification ]
```

Syntax

```
<select statement> ::= <cursor specification>

<cursor specification> :=
    <query expression>
    [ <order by clause> ]

<query expression> ::=
    | <query term>
    | <query expression> UNION [ ALL | DISTINCT ] <query term>
    | <query expression> EXCEPT [ ALL | DISTINCT ] <query term>

<query term> ::=
    | <query specification>
    | ( <query expression> )
    <query term> INTERSECT [ ALL | DISTINCT ] <query expression>
```

```

<query specification> ::=

    SELECT [ DISTINCT | ALL ] [ TOP n [ PERCENT ] [ { , start-position } ] ] <select list>
    [ INTO table-name ]
    <table expression>

<select list> ::=
| *
| <select sublist> [ { , <select sublist> }... ]

<select sublist> ::=
| derived-column [ [ AS ] column alias ]
| qualified-asterisk

<table expression> ::=
<from clause>
[ <where clause> ]
[ <group by clause> ]
[ <having clause> ]

<from clause> := FROM <table reference> [ { , <table reference> }... ]

<table reference> ::=
| <table factor>
| <joined table>

<table factor> ::=
| [ schema-name. ] table-name [ [ AS ] correlation-name [ ( <derived column list> ) ] ]
| <table subquery> [ AS ] correlation-name [ ( <derived column list> ) ]
| <table function derived table> [ AS ] correlation-name [ ( <derived column list> ) ]

<derived column list> ::= column alias [ { , column alias }... ]

<table function derived table> ::= TABLE ( <routine invocation> )

<joined table> ::=
| <cross join>
| <qualified join>
| <natural join>

<cross join> ::= <table reference> CROSS JOIN <table factor>

<qualified join> ::= <table reference> [ <join type> ] JOIN <table reference> <join specification>

<natural join> ::= <table reference> NATURAL [ <join type> ] JOIN <table factor>

<join type> ::=
| INNER
| LEFT [ OUTER ]
| RIGHT [ OUTER ]
| FULL [ OUTER ]

<join specification> ::=

```

```

| ON <search condition>
  USING ( column-name [ { , column-name }... ] )

<where clause> ::= WHERE <search condition>

<group by clause> ::= GROUP BY <grouping element> [ { , <grouping element> }... ]

<grouping element> ::= column-reference <collate clause>

<having clause> ::= HAVING <search condition>

<order by clause> ::= ORDER BY <sort specification> [ { , <sort specification> }... ]

<sort specification> ::= column-reference <sort options>

```

Live and static cursors

NexusDB supports both live (updateable) and static (read-only) cursors as the result of executing a SELECT statement. A live cursor is effectively a direct cursor on the underlying base table referenced in the table expression, with a filter set to the source columns specified in the select list and the rows resulting from evaluating the query specification. A SELECT statement will return a static result set (a special in-memory table) if the query expression does not satisfy the requirements of a live cursor, or the client access software executing the query does not support live cursors.

The requirements for a live cursor are as follows:

- The FROM clause shall reference a single source table.
- A DISTINCT quantifier shall not be specified.
- A TOP quantifier shall not be specified.
- A GROUP BY clause shall not be specified.
- If an ORDER BY clause is specified, then a supporting index that includes the same columns and sort options must exist.

Table operators

- The UNION, EXCEPT and INTERSECT set operators are implicitly DISTINCT if no set quantifier is specified. The result of a table operation with DISTINCT implied or specified contains no duplicate rows.

- The UNION operator is used to combine the tables derived from two query expressions. The two tables shall have the same number of columns (degree), and each corresponding column shall have the same ordinal position and a compatible data type. The result of an UNION operation is a combination of the two tables without duplicate rows. If ALL is specified, then the duplicate rows are preserved.
- The EXCEPT operator is used to return all rows in the first table except those that also appear in the second table. If ALL is specified, then the number of duplicates of any specific row is equal to the number of such duplicates in the first table less the number of duplicates in the second table.
- The INTERSECT operator is used to return rows that appear in both tables. If ALL is specified, then the number of duplicates of any specific row is equal to the lesser number of such duplicates in the two tables.
- INTERSECT has a higher precedence than UNION and EXCEPT which are executed left to right.
- The data type of columns in the result of a query expression with UNION, EXCEPT and INTERSECT is determined according to the rules for Result data types of aggregations.

Joined tables

NexusDB supports all common types of joins between tables. While the old-style join syntax with a comma-separated list of table names in the FROM clause and the join criteria contained in the WHERE clause, is still supported in SQL, we recommend the use of the new syntax introduced in SQL-92 and carried on by the latest revisions of the standard.

- The CROSS join is effectively a cross-product of the two tables, commonly referred to as the Cartesian product, containing all possible combinations of rows from the first and second tables. Like its equivalent, the classic comma-separated join, a more meaningful result can be obtained by specifying a join criteria in the WHERE clause.
- A NATURAL join is based on all columns in the referenced tables with the same name, and retrieves the rows that have equal values in the corresponding columns. If there are no corresponding columns, the result is a cross-join between the two tables.
- A qualified join connects two tables based on an explicit join specification. The join specification can be expressed as a condition join with the ON keyword followed by a search condition, or as a column name join with the USING keyword followed by a list of corresponding columns.
- The INNER keyword is implicit if no join type is specified. An INNER JOIN retrieves only the rows that match the join criteria. The OUTER keyword is implicit for outer joins, but can optionally be included for clarity. A LEFT OUTER JOIN will preserve all rows from the left table with null values

in the columns of non-matching rows that correspond to the right table. A RIGHT OUTER JOIN is similar to a LEFT JOIN, except that the right table has its rows preserved. Finally, a FULL OUTER JOIN will preserve the rows from both tables.

SELECT clause

- The select list specifies the source columns to be retrieved from the table expression and additional columns computed as the result of a value expression.
- An asterisk (*) is a short-hand syntax for specifying all columns that are part of the table expression. An asterisk qualified by a table name specifies all the fields from that particular table.
- A derived-column can be a column reference, optionally qualified by the table name, or any other valid value expression, including aggregate functions, value functions, user-defined functions, scalar subqueries, and case expressions.
- The optional AS column-alias clause is used to give computed columns meaningful names or to rename source columns.
- The ALL keyword is implicit if no set quantifier is specified. If DISTINCT is specified, then all duplicate rows are removed from the result set.
- The TOP quantifier is used to restrict the result set to the number of rows specified by the *n* argument, or a percentage of all rows in the result set if the *n* argument is followed by the PERCENT keyword. The optional start-position argument indicates the position of the first row to be included in the result set.

Tip: A syntactical way of retrieving the last *n* rows, is to ORDER BY a descending sort specification using the TOP quantifier.

INTO clause

- The INTO clause is used to create a new table with columns according to the select list and data resulting from executing the query.
- Each column in the new table has the same name, data type, ordinal position and row values as the corresponding columns in the select list. Computed columns in the select list will be ordinary columns in the new table, with the column values computed by the query execution.

- If a table with the same name already exists in the database, then the old table will be implicitly dropped before the new table is created.
- A new table without data can be created by having a FALSE condition in the WHERE clause.
- Using the INTO clause to create a new table is an alternate syntax for the AS subquery clause in the CREATE TABLE statement.

FROM clause

- The FROM clause specifies one or more source tables from which to retrieve rows. The name of base tables, views and user-defined functions may optionally be qualified by the schema name to reference such objects in databases other than the current, or just for syntax clarity.
- A table reference can specify a base table, a view, a subquery, or a table-valued function.
- A correlation name is required for source tables derived from a subquery or a user-defined function, but is optional for base tables and views.
- A derived column list may be specified to rename the columns of the table reference. The number of columns in the list shall match the number of source columns contained in the table reference.
- If two or more tables are referenced in the FROM clause without an explicit join or a WHERE clause joining the tables, the result will be a cross-join between the tables.

WHERE clause

- The WHERE clause is a filter that is applied to the result of evaluating the FROM clause.
- Only the rows that satisfy the criteria specified in the search condition are included in the result set.
- The scope of the WHERE clause is the tables referenced in the FROM clause. Expression columns or renamed columns in the select list cannot be referenced in the WHERE clause by their column aliases.

GROUP BY clause

- The effect of the GROUP BY clause is to partition the result of evaluating the table expression so far into one or more groups, such that for each grouping column, no two rows of any group have different values for the grouping column. Furthermore, no other group has the same value for the complete set of grouping columns.
- Each group is represented in the result set by a single row with aggregate functions, if specified, applied to the group.
- All non-aggregate source columns that are referenced in the select list shall be specified as grouping columns.
- The GROUP BY clause may also specify source columns that are not referenced in the select list.
- Each grouping column shall be a reference to a source column in the table expression.

HAVING clause

- The HAVING clause works like the WHERE clause, except that the filter criteria is applied to the groups of rows in the grouped table resulting from evaluating the GROUP BY clause.
- If no GROUP BY clause is specified, then the result of evaluating the preceding clause is considered a single group.
- Columns referenced in the search condition shall be a grouping column, a source column contained in an aggregate function or an outer reference.

ORDER BY clause

- The ORDER BY clause specifies the order of rows in the result set.
- The optional sort options can be specified to override the default sort behavior of the individual columns.
- The sort keys shall be a reference to columns in the select list.

- If DISTINCT, GROUP BY, UNION, EXCEPT or INTERSECT is not specified in the main query expression, then the ORDER BY clause may also specify source columns that are not included in the select list.
- Columns that are present in the select list can be referenced by the column name or the ordinal position.

Note: The option of referencing a column by its ordinal position has been deprecated, and may be removed in a future version of NexusDB SQL.

Examples

- 1) The following example selects all students ordered by names:

```
SELECT studentID, studentName, gender  
FROM students  
ORDER BY lastName, firstName
```

- 2) The following example selects either female or male students depending on the value of the gender parameter:

```
SELECT studentID, studentName, gender  
FROM students  
WHERE gender = :gender  
ORDER BY lastName, firstName
```

- 3) The following example selects a list of students with a computed column called nickName:

```
SELECT studentID, city || '-' || firstName AS  
nickName, lastName, gender  
FROM students  
ORDER BY lastName, firstName
```

- 4) The following example computes how many students are enrolled in the available courses:

```
SELECT courseID, count( * ) AS total_enrolled  
FROM enrolls  
GROUP BY courseID  
ORDER BY total_enrolled DESC
```

- 5) The following example select courses with more than 5 students enrolled:

```
SELECT courseID, count( * ) AS total_enrolled
FROM enrolls
GROUP BY courseID
HAVING COUNT( * ) > 5
ORDER BY total_enrolled DESC
```

- 6) The following example uses joins to select columns from three tables:

```
SELECT
    e.courseID,
    c.courseName,
    s.studentName,
    e.grade
FROM enrolls e
JOIN courses c ON c.courseID = e.courseID
JOIN students s USING ( studentID )
ORDER BY courseID
```

- 7) The following example combines rows from two tables using the UNION ALL table operator:

```
SELECT orderID, orderDate, orderTotal,
amountPaid
FROM orders_2002
UNION ALL
SELECT orderID, orderDate, orderTotal,
amountPaid
FROM orders_2003
ORDER BY orderDate, orderID
```

- 8) The following example uses a subquery as table reference:

```
SELECT studentID, studentName, city, gender
FROM ( SELECT * FROM students ) AS s
ORDER BY lastName, firstName
```

- 9) The following example uses a table function as table reference:

```
SELECT studentID, studentName, city, gender
```

```
FROM TABLE( getStudents() ) AS s
ORDER BY lastName, firstName
```

Conformance

SQL:2003 standard - Core SQL

- Feature F302 "INTERSECT table operator"
- Feature F304 "EXCEPT ALL table operator"
- Feature F401 "Extended joined table"
- Feature F591 "Derived tables"
- Feature T326 "Table functions"
- Feature T551 "Optional key words for default syntax"

NexusDB extensions	<ul style="list-style-type: none"> - TOP quantifier in SELECT clause - INTO clause in query specification - Sort options
-----------------------	---

22.4.2 INSERT statement

INSERT statement

Create new rows in a table.

Syntax

```
<insert statement> ::=  
    INSERT INTO [ schema-name. ] table-name <insert columns and source>  
  
<insert columns and source> ::=  
    | <from subquery>  
    | <from constructor>  
    DEFAULT VALUES  
  
<from subquery> ::=  
    [ ( column-name [ { , column-name }... ] ) ] <query expression>  
  
<from constructor> ::=  
    [ ( column-name [ { , column-name }... ] ) ] <table value constructor>  
  
<table value constructor> ::=  
    VALUES ( <row value expression> [ { , <row value expression> }... ] )
```

Usage

The INSERT statement is used to add new rows to the table identified by table-name.

Notes

- The current database is implicit if schema-name is not specified.
- There are two main variations of the INSERT statement, the first is using a query expression to add rows based on data from existing tables in the database, and the second is using a table value constructor to specify a single row or a comma-separated list of row values.
- The DEFAULT VALUES clause is a third option that can be used to insert a new row with default values in every column.
- The optional comma-separated list of insert columns is particularly useful when inserting rows into a wide table and only a few columns will have initial data. Columns that are not specified in the insert column list will get the default value if specified in the table definition, otherwise the null value.
- The number of values in the inserting rows, whether retrieved by a query expression or specified in the VALUES clause, shall match the number of insert columns or, if the insert column list is omitted, the degree of the target table.
- The values assigned in the VALUES clause can be any valid value expression, including value functions, user-defined functions, scalar subqueries, case expressions, DEFAULT and NULL.
- Parameters can be used to represent values supplied by the application.

Tip: Using multiple row values in the VALUES clause is more efficient than executing a sequence of single row INSERT statements.

Examples

- 1) The following example inserts a new row into the students table, only assigning values to some of the columns:

```
INSERT INTO students
( studentID, lastName, firstName , address, state,
zip, gender)
VALUES ( 211, 'Smith', 'Joanne', NULL, 'QL',
45678, 'F' )
```

- 2) The following example shows a generic INSERT statement for the students table using parameters to assign column values:

```
INSERT INTO students
VALUES (
    :studentID,
    :firstName || ' ' || :lastName,
    :lastName,
    :firstName,
    :address,
    :city,
    :state,
    :zip,
    :gender
)
```

- 3) The following example uses a subquery to insert all the rows of the students table into the ##students table:

```
DELETE FROM ##students;
INSERT INTO ##students
SELECT * FROM students
```

- 4) The following example inserts two new rows into the documents table:

```
INSERT into documents ( documentID, title )
VALUES ( NEWGUID, 'NexusDB main features' ),
( NEWGUID, 'NexusDB special features' )
```

Conformance

SQL:2003 standard - Core SQL

- Feature F222 "INSERT statement: DEFAULT VALUES clause"

22.4.3 UPDATE statement

UPDATE statement

Update rows of a table.

Syntax

```
<update statement: searched> ::=  
    UPDATE [ schema-name. ] table-name [ [ AS ] correlation-name ]  
        SET <set clause list>  
        [ WHERE <search condition> ]  
  
<set clause list> ::=  
    column-name = <value expression> [ { , column-name = <value expression> }... ]
```

Usage

The UPDATE statement is used to modify the value of one or more columns in a selection of rows in the table identified by table-name.

Notes

- The current database is implicit if schema-name is not specified.
- The SET clause is a comma-separated list of column assignments. The assigned value can be any valid value expression, including value functions, user-defined functions, scalar subqueries, case expressions, DEFAULT and NULL.
- Parameters can be used to represent values supplied by the application.
- The WHERE clause restricts the rows that are updated to those satisfying the search condition. If no WHERE clause is specified, all rows in the table are updated.

Tip: Using a prepared statement with parameters is more efficient than executing a sequence of individual UPDATE statements with explicit values.

Examples

- 1) The following example uses literals and value expressions to update studentID 211:

```
UPDATE students  
SET  
  
studentName = firstName || ' ' || lastName,  
address = '983 Park Avenue',  
city = 'Boston',
```

```
state = ( SELECT state FROM students WHERE  
studentID = 473 ),  
zip = '02169'
```

```
WHERE studentID = 211
```

- 2) The following example shows a generic UPDATE statement for the students table using parameters to update column values and identify the row:

```
UPDATE students  
SET  
  
studentName = :firstName || ' ' || :lastName,  
lastName = :lastName,  
firstName = :firstName,  
address = :address,  
city = :city,  
state = :state,  
zip = :zip,  
gender = :gender  
  
WHERE studentID = :studentID
```

Conformance

SQL:2003 standard - Core SQL

22.4.4 DELETE statement

DELETE statement

Delete rows of a table.

Syntax

```
<delete statement: searched> ::=  
    DELETE FROM [ schema-name. ] table-name [ [ AS ] correlation-name ]  
    [ WHERE <search condition> ]
```

Usage

The DELETE statement is used to delete a selection of rows from the table identified by table-name.

Notes

- The current database is implicit if schema-name is not specified.
- The WHERE clause restricts the rows that are deleted to those satisfying the search condition.
- If no WHERE clause is specified, then all the rows of the table are deleted.

Tip: A more efficient way of deleting all the rows in a table is to drop the table and then recreate it.

Examples

- 1) The following example deletes students who are not enrolled in any courses:

```
DELETE FROM students
WHERE studentID IN (
    SELECT DISTINCT studentID
    FROM students s
    LEFT JOIN enrolls e ON e.studentID =
        s.studentID
    WHERE e.studentID IS NULL
)
```

- 2) The following example uses a parameter to identify the row to delete:

```
DELETE FROM students
WHERE studentID = :studentID
```

- 3) The following example deletes all rows in the ##students table:

```
DELETE FROM ##students
```

Conformance

SQL:2003 standard - Core SQL

22.4.5 EXECUTE IMMEDIATE Statement

EXECUTE IMMEDIATE statement

Execute dynamic SQL statements from within triggers etc

Syntax

<ExecuteStatement> ::= EXECUTE IMMEDIATE <SimpleExpression>

Usage

Note: the example shows how to work with single quoted parts of the statement

```
declare s varchar(100);

set s = 'select ' + chr(39) + 'a string' + chr(39) + ' from #dummy';

execute immediate (s);
```

22.5 CURSOR Statements

DECLARE CURSOR Statement
OPEN Statement
FETCH Statement
CLOSE statement

22.5.1 CLOSE statement

CLOSE statement

Syntax

<CloseStatement> ::= 'CLOSE' <identifier>

Usage

Closes a previously declared named cursor.

22.5.2 DECLARE CURSOR Statement

DECLARE CURSOR statement

Syntax

<DeclareStatement> ::= DECLARE <identifier> CURSOR FOR <SELECT statement>

Usage

Declares a named cursor. Example:

```
drop function if exists createquery;
create function createquery( tablename varchar( 100))
returns varchar( 32767)
reads sql data
begin
    declare cRet varchar( 32767);
    declare cSep varchar( 5);
    declare myFieldname varchar( 32);
    declare myFieldlength integer;
    declare myFieldType varchar( 32);
    declare nLengthOfOtherFields integer;
    declare nRL integer;
    set cRet = 'select ';
    set cSep = '';
    set nLengthOfOtherFields = 0;
    declare aCursor cursor for select f.field_name, f.field_type_sql,
f.field_length
                                from #tables t, #fields f
                                where f.table_name = t.table_name
                                and lower(t.table_name) =
lower(tablename);
    open aCursor;
    fetch first from aCursor into myFieldName, myFieldType, myFieldLength;
    while @@fetch_status = 0 do
        if myFieldType = 'CHARACTER VARYING' then
            set cRet = cRet || cSep || 'sum( coalesce( char_length( ' ||
myFieldname || '),0)) ';
            set cSep = '+';
        else
            set nLengthOfOtherFields = nLengthOfOtherFields + myFieldLength;
        end if;
        fetch next from aCursor into myFieldName, myFieldType, myFieldLength;
    end while;
    close aCursor;
    set cRet = cRet || '+' || tostring( nLengthOfOtherFields) || '
SizeOfAllTheData ';

    declare aCursor cursor for select record_length from #tables where
lower( table_name ) = lower( tablename );
    open aCursor;
    fetch first from aCursor into nRL;
    set cRet = cRet || ', sum( ' || tostring( nRL ) || ' )
SizeOfAllocatedData ';
    close aCursor;
    set cRet = cRet || ' from ' || tablename || ';';
    return cRet;
end;

select createquery( '<table name>' ) from #dummy;
```

22.5.3 FETCH Statement

Fetch values from a declared cursor.

Syntax

```
<FetchStatement> ::= FETCH [ ( CURRENT | NEXT | PRIOR | FIRST | LAST | ABSOLUTE  
<SimpleExpression> | RELATIVE <SimpleExpression> ) ] FROM <identifier>
```

Usage

See example in DECLARE CURSOR Statement topic

22.5.4 OPEN Statement

OPEN statement

Syntax

```
<OpenStatement> ::= 'OPEN' <identifier>
```

Usage

Opens a previously declared named cursor.

22.6 Transaction Statements

Transaction Statements

Transaction statements are used to start and terminate explicit transactions.

A transaction asserts data integrity by demanding that all statements executed in the context of the transaction must complete successfully before the transaction can be committed, otherwise the transaction is aborted and no changes are updated in the database.

Note: If an explicit transaction is not active, then each SQL statement starts an implicit transaction which is automatically committed when the SQL statement terminates.

Tip: Use transaction blocks to speed up the execution of bulk data-change statements.

START TRANSACTION statement
COMMIT statement
ROLLBACK statement

22.6.1 START TRANSACTION statement

START TRANSACTION statement

Start a transaction.

Syntax

```
<start transaction statement> ::= START TRANSACTION [ <transaction mode> ]  
<transaction mode> ::=  
| SNAPSHOT  
| SERIALIZABLE
```

Usage

The START TRANSACTION statement starts a new transaction in the current session.

Notes

- An isolation level of SERIALIZABLE is implicit if a transaction mode is not specified.
- SNAPSHOT defines a read-only transaction with SERIALIZABLE isolation level. Since SNAPSHOT transactions don't put any locks on the tables involved in the transaction, SERIALIZABLE transactions from other sessions may change data while the SNAPSHOT transaction is active.
- All SQL statements between a START TRANSACTION statement and a COMMIT or ROLLBACK statement will be part of the transaction.
- If an error occurs during the execution of any following SQL statements, then the transaction is aborted and no changes are updated in the database.
- Schema statements (DDL) are executed under control of an internal transaction, and shall not be specified in the context of an explicit transaction.

Examples

- 1) The following example starts a new transaction:

```
START TRANSACTION;
```

- 2) The following example starts a new snapshot transaction:

```
START TRANSACTION SNAPSHOT;
```

Conformance

SQL:2003 standard - Feature T241 "START TRANSACTION statement"
- Transaction access mode is not supported.

NexusDB - SNAPSHOT
extensions

22.6.2 COMMIT statement

COMMIT statement

Commit a transaction.

Syntax

<commit statement> ::= COMMIT [WORK]

Usage

The COMMIT statement terminates the active transaction and makes all changes made during the transaction permanent in the database.

Notes

- The COMMIT statement will return an error if no transaction is active.

Examples

- 1) The following example commits the current transaction:

```
COMMIT;
```

Conformance

SQL:2003 standard - Core SQL

22.6.3 ROLLBACK statement

ROLLBACK statement

Cancel a transaction.

Syntax

<rollback statement> ::= ROLLBACK [WORK]

Usage

The ROLLBACK statement terminates the active transaction and cancels all changes made during the transaction.

Notes

- The COMMIT statement will return an error if no transaction is active.

Examples

- The following example cancels the current transaction:

```
ROLLBACK;
```

- The following example shows the ROLLBACK statement used in the context of a TRY statement:

```
START TRANSACTION;
TRY

// Execute some data-change statements here.
COMMIT;

CATCH POSITION( 'Unable to open table' IN
ERROR_MESSAGE ) <> 0

ROLLBACK;
SIGNAL ERROR_MESSAGE;
```

END;

Conformance

SQL:2003 standard - Core SQL

22.7 Session Statements

Session Statements

Session statements are used to set properties and default values of an SQL-session.

SET PASSWORDS statement
ALTER SYSTEMPASSWORD statement
CREATE ALIAS statement
ALTER ALIAS statement
DROP ALIAS Statement
USE Statement

22.7.1 SET PASSWORDS statement

SET PASSWORDS statement

Set one ore more passwords.

Syntax

```
<set passwords statement> ::=  
    SET PASSWORDS [ [ ADD | REMOVE ] password-list ]
```

Usage

The SET PASSWORDS statement is used to manage a list of encryption passwords for the current session. Tables that are encrypted with a password cannot be accessed in SQL statements unless the encryption password defined for the table is present in the list of passwords assigned to the session.

Notes

- The password-list is a comma separated list of passwords.
- SET PASSWORDS ADD password-list adds new passwords to the list.
- SET PASSWORDS REMOVE password-list removes the passwords specified by password-list from the list.
- SET PASSWORDS password-list sets the password list according to the argument, replacing any existing passwords.
- SET PASSWORDS sets the password list to an empty string.

Examples

1) The following example adds passwords to the password list of the current session:

```
SET PASSWORDS ADD 'pw100, pw101, pw102'
```

2) The following example removes one of the existing passwords in the list:

```
SET PASSWORDS REMOVE 'pw101'
```

3) The following example replaces the existing password list:

```
SET PASSWORDS 'pw103'
```

4) The following example removes all passwords:

```
SET PASSWORDS
```

Conformance

NexusDB extensions - SET PASSWORDS statement

22.7.2 ALTER SYSTEMPASSWORD statement

ALTER SYSTEMPASSWORD statement

Add password to session.

Syntax

```
<alter systempassword statement> ::=  
    ALTER SYSTEMPASSWORD TO <character string literal>
```

Usage

The ALTER SYSTEMPASSWORD statement is used to change the password for the active session.

22.7.3 CREATE ALIAS statement

CREATE ALIAS statement

Create a new alias

Syntax

```
<CreateAliasStatement> ::= CREATE ALIAS <SqlName> FROM <AliasPath>
```

Usage

The Create Alias statement is used to create a new alias.

22.7.4 ALTER ALIAS statement

ALTER ALIAS statement

Change name or path of an alias

Syntax

```
<AlterAliasStatement> ::= " ALTER ALIAS" <SqlName> "SET" ( "PATH" "TO" <AliasPath> | 'NAME'  
    "TO" <SqlName> )
```

Usage

The Alter Alias statement is used to change the path (file system folder) or name of an alias.

22.7.5 DROP ALIAS Statement

DROP ALIAS statement

Drops (removes) an alias

Syntax

```
<DropAliasStatement> ::= DROP ALIAS [ IF EXISTS ] <SqlName>
```

Usage

The Drop Alias statement is used to drop an alias.

22.7.6 USE Statement

USE statement

Changes the database in use to the new alias

Syntax

```
<UseStatement> ::= USE [<SqlName> | DEFAULT_ALIAS ]
```

Usage

The Use statement is used to switch the active database to the database of a different alias. Subsequent SQL operations will operate on the database in the new alias. To return to the original, use the USE DEFAULT_ALIAS statement.

```
--client started operations in 'LiveData' alias  
USE 'backup'; --now operates on the 'backup' database  
----some operations...  
USE DEFAULT_ALIAS; --now operates on 'LiveData' alias again
```

22.8 Control Statements

Control Statements

Control statements, the main part of the procedure language in NexusDB SQL, add computational completeness to the SQL language with the Compound statement, SQL variable declarations, conditional statements, iterating statements and error handling.

CALL statement
RETURN statement
Compound statement
DECLARE variable statement
SET statement
IF statement
ITERATE statement
LEAVE statement
REPEAT statement
WHILE statement
SIGNAL statement
TRY statement

22.8.1 CALL statement

CALL statement

Invoke a user-defined procedure.

Syntax

<call statement> ::= CALL <routine invocation>

Usage

The CALL statement is used to invoke a user-defined procedure.

Notes

- The CALL statement can only invoke a procedure.
- The number and types of arguments passed to the procedure call must match the SQL parameter declarations in the procedure definition.

Examples

- 1) The following example invokes a user-defined procedure using dynamic parameters to pass values:

```
CALL
addCourse( :courseID, :courseName, :department,
:numCredits );
```

Conformance

SQL:2003 standard - Core SQL

22.8.2 RETURN statement

RETURN statement

Return a function result.

Syntax

<return statement> ::= RETURN <return value>

<return value> ::=
 | <value expression>
 | NULL
 <cursor specification>

Usage

The RETURN statement is used to assign the return value of a user-defined function.

Notes

- The RETURN statement can only appear in the routine body of a function definition.
- The return value shall be compatible with the type specified in the RETURNS clause of the function definition.
- The cursor specification option shall only be used if the RETURNS clause specify TABLE.

Examples

- 1) The following example returns a character string value:

```
RETURN 'Hello World';
```

2) The following example returns the system date and time:

```
RETURN CURRENT_TIMESTAMP;
```

3) The following example returns a cursor derived from the students table:

```
RETURN SELECT * FROM students;
```

Conformance

SQL:2003 standard - Core SQL

NexusDB extension- Cursor specification as return value

22.8.3 Compound statement

Compound statement

Specify a statement block.

Syntax

```
<compound statement> ::=  
  BEGIN [ [ NOT ] ATOMIC ]  
  [ <local declaration list> ]  
  [ <SQL statement list> ]  
  END
```

```
<local declaration list> ::= { <SQL variable declaration> ; }...
```

```
<SQL statement list> ::= { <SQL procedure statement> ; }...
```

Usage

The Compound statement is used to specify a sequence of SQL procedure statements inside a BEGIN..END block.

Notes

- NOT ATOMIC is implicit if neither ATOMIC nor NOT ATOMIC is specified.
- If ATOMIC is specified, then an internal transaction is started before the statement block is executed. The transaction is committed only if all the statements in the block executed successfully, otherwise the transaction is rolled back.
- All SQL variable declarations shall be specified before other SQL statements.

Examples

- 1) The following example shows a Compound statement with parameter declarations:

```
DROP FUNCTION IF EXISTS dummyFunc;
CREATE FUNCTION dummyFunc ()
RETURNS DOUBLE PRECISION
BEGIN

DECLARE d1, d2 DOUBLE PRECISION;
SET d1 = 25;
SET d2 = 4;
RETURN d1 * d2;

END;
SELECT dummyFunc FROM #dummy;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-01 "Compound statement"
- Beginning and ending statement labels are not supported

22.8.4 DECLARE variable statement

DECLARE variable statement

Declare one or more SQL variables.

Syntax

<SQL variable declaration> ::=

```
DECLARE <SQL variable name list> <data type> [ <default clause> ]  
<SQL variable name list> ::= variable-name [ { , variable-name }... ]
```

Usage

The DECLARE statement is used to declare SQL variables. SQL variables that are declared in the context of triggers, procedures and functions are local to the statement block where they are declared. SQL variables that are declared in the context of the main executing statement block are global session variables.

Notes

- SQL variable names shall conform to the rules for identifiers in NexusDB SQL.
- One or more SQL variables can be declared with the same data type.
- The DEFAULT clause is used to specify an initial value for the variable, and is similar to setting a default value for a column.

Examples

- 1) The following example declares two SQL variables with the same data type:

```
DECLARE min, max INTEGER;
```

- 2) The following example declares two SQL variables with different data types:

```
DECLARE count INTEGER DEFAULT 0;  
DECLARE total DOUBLE PRECISION DEFAULT  
0;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-04 "SQL variable declaration"

22.8.5 SET statement

Assignment statement

Assign a value to an assignment target.

Syntax

```

<assignment statement> ::=  
    SET <assignment target> = <assignment source>  
  

<assignment target> ::= <target specification>  
  

<assignment source> ::=  
    | <value expression>  
    | NULL  
    DEFAULT
  
```

Usage

The SET statement is used to assign values to assignment targets such as SQL variables, SQL parameters and columns of a transition table.

Notes

- The data types of the assignment target and the assignment source shall be compatible.

Examples

- The following example assigns a value to the name SQL variable:

```
SET name = firstName || ' ' || lastName;
```

- The following example assigns a value to a column of the transition table pointed to by the NEW variable in the context of a trigger definition:

```
SET NEW.lastUpdated =  
CURRENT_TIMESTAMP;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-05 "Assignment statement"

22.8.6 IF statement

IF statement

Specify a conditional execution.

Syntax

<if statement> ::=

```
  IF <search condition>
    <if statement then clause>
    [ <if statement elseif clause>... ]
    [ <if statement else clause> ]
  END IF
```

<if statement then clause> ::= THEN <SQL statement list>

<if statement elseif clause> ::= ELSEIF <search condition> THEN <SQL statement list>

<if statement else clause> ::= ELSE <SQL statement list>

Usage

The IF statement is used to specify a conditional execution of one or more SQL statements, and is similar in functionality to conditional IF statements found in traditional programming languages.

Notes

- The optional ELSEIF clause is used to specify additional conditions with separate statement blocks, and is a convenient shorthand for nesting multiple IF..THEN statements.
- Only the statements contained in the first IF or ELSEIF clause with a condition that evaluates to true are executed.
- An optional ELSE clause can be specified to execute one or more statements if none of the preceding conditions were true.

Examples

- 1) The following example shows the IF statement used in the body of a trigger definition:

```
DROP TRIGGER IF EXISTS enrolls_changes;
CREATE TRIGGER enrolls_changes
AFTER INSERT, DELETE, UPDATE ON enrolls
REFERENCING OLD AS o NEW AS n
BEGIN

IF INSERTING THEN

    INSERT INTO enrolls_log ( action, n1, n2, n3, n4,
    stamp )
    VALUES ( 'Insert', n.courseID, n.sectionID,
    n.studentID, n.grade, CURRENT_TIMESTAMP );

ELSEIF UPDATING THEN

    INSERT INTO enrolls_log ( action, o1, n1, o2, n2,
    o3, n3, o4, n4, stamp )
    VALUES ('Update', o.courseID, n.courseID,
    o.sectionID, n.sectionID,
    o.studentID, n.studentID, o.grade, n.grade,
    CURRENT_TIMESTAMP );

ELSEIF DELETING THEN

    INSERT INTO enrolls_log ( action, o1, o2, o3, o4,
    stamp )
    VALUES ( 'Delete', o.courseID, o.sectionID,
    o.studentID, o.grade, CURRENT_TIMESTAMP );

ELSE

    SIGNAL 'An unknown trigger event was fired.';

END IF;

END
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-07 "IF statement"

22.8.7 ITERATE statement

ITERATE statement

Start a new iteration.

Syntax

<iterate statement> ::= ITERATE

Usage

The ITERATE statement can only be specified in the body of the REPEAT and WHILE statements to start a new iteration from the first statement in the block.

See also: LEAVE statement

Examples

1) The following example shows the ITERATE statement used in SQL code:

```
DROP FUNCTION IF EXISTS dummyFunc;
CREATE FUNCTION dummyFunc( p1 INTEGER )
RETURNS INTEGER
BEGIN

DECLARE v1, v2 INTEGER;
SET v1 = 10;
SET v2 = 10;
WHILE v1 > 0 DO

    SET v1 = v1 - 1;
    IF v1 <= 5 THEN

        ITERATE;

    END IF;
    SET v2 = v2 - 1;

END WHILE;
RETURN v2 + p1;

END;
SELECT dummyFunc( 15 ) FROM #dummy;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-08 "ITERATE statement"

- Statement label is not supported

22.8.8 LEAVE statement

LEAVE statement

Exit from an iterated statement.

Syntax

<leave statement> ::= LEAVE

Usage

The LEAVE statement can only be specified in the body of the REPEAT and WHILE statements to stop the iteration and exit the statement.

See also: ITERATE statement

Examples

1) The following example shows the LEAVE statement used in SQL code:

```
DROP FUNCTION IF EXISTS dummyFunc;
CREATE FUNCTION dummyFunc( p1 INTEGER )
RETURNS INTEGER
BEGIN

DECLARE v1, v2 INTEGER;
SET v1 = 10;
SET v2 = 10;
WHILE v1 > 0 DO

SET v1 = v1 - 1;
IF v1 >= 5 THEN

LEAVE;

END IF;
SET v2 = v2 - 1;

END WHILE;
RETURN v2 + p1;

END;
SELECT dummyFunc( 15 ) FROM #dummy;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-09 "LEAVE statement"
- Statement label is not supported

22.8.9 REPEAT statement

REPEAT statement

Repeat the execution until the condition is true.

Syntax

```
<repeat statement> ::=  
    REPEAT  
        <SQL statement list>  
        UNTIL <search condition>  
    END REPEAT
```

Usage

The REPEAT statement is used to specify a repeated iteration of the statement block **until** the search condition is true.

See also: WHILE statement

Examples

1) The following example shows the REPEAT statement used in SQL code:

```
DROP FUNCTION IF EXISTS dummyFunc;  
CREATE FUNCTION dummyFunc( p1 INTEGER )  
RETURNS INTEGER  
BEGIN  
  
    DECLARE v1, v2 INTEGER;  
    SET v1 = p1;  
    SET v2 = p1 * 10;  
    REPEAT  
  
        SET v1 = v1 + p1;
```

```
UNTIL v1 = v2  
END REPEAT;  
RETURN v1;  
  
END;  
SELECT dummyFunc( 10 ) FROM #dummy;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-11 "REPEAT statement"
- Beginning and ending statement labels are not supported

22.8.10 WHILE statement

WHILE statement

Repeat the execution while the condition is true.

Syntax

```
<while statement> ::=  
    WHILE <search condition> DO  
        <SQL statement list>  
    END WHILE
```

Usage

The WHILE statement is used to specify a repeated iteration of the statement block **while** the search condition is true.

See also: REPEAT statement

Examples

1) The following example shows the WHILE statement used in SQL code:

```
DROP FUNCTION IF EXISTS dummyFunc;  
CREATE FUNCTION dummyFunc( p1 INTEGER )  
RETURNS INTEGER  
BEGIN
```

```
DECLARE v1, v2 INTEGER;
SET v1 = 10;
SET v2 = 10;
WHILE v1 > 0 DO

    SET v1 = v1 - 1;
    IF v1 <= 5 THEN

        ITERATE;

    END IF;
    SET v2 = v2 - 1;

    END WHILE;
    RETURN v2 + p1;

END;
SELECT dummyFunc( 15 ) FROM #dummy;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-12 "WHILE statement"
- Beginning and ending statement labels are not supported

22.8.11 SIGNAL statement

SIGNAL statement

Signal an exception condition.

Syntax

<signal statement> ::= SIGNAL <character string literal>

Usage

The SIGNAL statement is used to throw an exception with the message specified by the character string literal.

Examples

- 1) The following example throws an exception:

```
SIGNAL 'Dummy Exception.';
```

- 2) The following example shows the SIGNAL statement used in a conditional computation:

```
DECLARE v1, v2 DOUBLE PRECISION;
SET v1 = RAND;
SET v2 = RAND;
IF ( v1 = v2 ) THEN
    SIGNAL 'The values are equal.';
ELSE
    SIGNAL 'The values are different.';
END IF;
```

- 3) The following example shows the SIGNAL statement used in the CATCH clause of a TRY statement:

```
START TRANSACTION;
TRY
    DROP TABLE garbage;
    COMMIT;
    CATCH POSITION( 'Unable to open table' IN
        ERROR_MESSAGE ) <> 0
        ROLLBACK;
        SIGNAL 'The table couldn''t be dropped.' || CHR(13)
        || 'The transaction was rolled back.';
    END;
```

Conformance

SQL:2003 standard - SQL/PSM Feature P002-14 "SIGNAL statement"

22.8.12 TRY statement

TRY statement

Specify a TRY..CATCH block.

Syntax

```
<try statement> ::=  
    TRY <protected statement block> <catch clause>... END  
  
<protected statement block> ::= <SQL statement list>  
  
<catch clause> ::= CATCH error-condition [ <catch statement block> ]  
  
<catch statement block> ::= <SQL statement list>
```

Usage

The TRY statement is used to protect a block of SQL statements, and handle some or all of the possible errors that might occur during the execution, gracefully in SQL code. The concept is similar to TRY statements found in many programming languages.

Notes

- Several CATCH clauses may be specified, but at least one is required.
- A specific error-condition can be determined by evaluating the current error message using the ERROR_MESSAGE system function.
- The statement block of the first CATCH clause that evaluates to true is executed.
- If none of the CATCH clauses evaluates to true, then the current exception is raised.

Examples

- 1) The following example shows the TRY statement used to control an explicit transaction:

```
START TRANSACTION;  
TRY  
  
// Execute some data-change statements here.  
COMMIT;  
  
CATCH POSITION( 'Unable to open table' IN  
ERROR_MESSAGE ) <> 0
```

```
ROLLBACK;
SIGNAL ERROR_MESSAGE;

END;
```

Conformance

NexusDB - TRY statement
extensions

22.9 Special Statements

Special Statements

Special NexusDB SQL statements are classified under this category.

ASSERT TABLE statement

22.9.1 ASSERT TABLE statement

ASSERT TABLE statement

Syntax

<assert table statement> ::= ASSERT TABLE table-name = TABLE table-name

Usage

The assert table statement checks if the two tables have identical column definitions and row data, and throws an exception if any difference is detected.

Examples

1) ASSERT TABLE students = TABLE students_backup_20040810;

Conformance

NexusDB - ASSERT TABLE statement
extensions

23 System Tables

System Tables

SQL system tables

Table name	Description
\$SQL\$FUNCTIONS.nx1	Metadata for user-defined functions
\$SQL\$PROCEDURES.nx1	Metadata for user-defined procedures
\$SQL\$TRIGGERS.nx1	Metadata for triggers
\$SQL\$VIEWS.nx1	Metadata for views

Metadata views

Table name	Description
#META	This table
#TABLES	Base tables
#FIELDS	Columns
#INDICES	Indices
#INDEXES	Same as #INDICES
#INDEXFIELDS	Index segments
#FILES	Physical files
#DUMMY	Virtual test table with NULL as only row/column
#FUNCTIONS	User-defined functions
#PROCEDURES	Stored procedures
#PROCEDURE_ARGUMENTS	Stored procedure arguments
#TRIGGERS	Triggers
#VIEWS	Views
#COLLATIONS	Supported collations
#CODEPAGES	Supported code pages
#CHECK_CONSTRAINTS	Check constraints
#REFERENCES	Referential constraints
#FUNCTION_ARGUMENTS	Stored function arguments
#SCHEMAS	Database aliases
#FOREIGNKEY_CONSTRAINTS	Foreign key constraints

#FOREIGNKEY_CONSTRAINTS_REFERENCING_COLUMNS	Foreign key constraints referencing columns
#FOREIGNKEY_CONSTRAINTS_REFERENCED_COLUMNS	Foreign key constraints referenced columns
#ASSEMBLIES	Known assemblies
#ASSEMBLYFUNCTIONS	Functions of an Assembly
#SERVER_SETTINGS	Server Settings
#SESSIONS_STATS	Server Engine Statistics - all sessions, including closed ones
#CURRENT_SESSION_STATS	Server Engine Statistics - the current session (always only 1 row)

24 Appendices

24.1 Core SQL:2003 Features

Core SQL:2003 Features

Overview of Core SQL:2003 Features

Feature ID	Feature description	NexusDB
E011	Numeric data types	
E011-01	INTEGER and SMALLINT data types (including all spellings)	✓
E011-02	REAL, DOUBLE PRECISION, and FLOAT data types	✓
E011-03	DECIMAL and NUMERIC data types	✓
E011-04	Arithmetic operators	✓
E011-05	Numeric comparison	✓
E011-06	Implicit casting among the numeric data types	✓
E021	Character string types	
E021-01	CHARACTER data type (including all its spellings)	✓
E021-02	CHARACTER VARYING data type (including all its spellings)	✓
E021-03	Character literals	✓
E021-04	CHARACTER_LENGTH function	✓
E021-05	OCTET_LENGTH function	✓
E021-06	SUBSTRING function	✓
E021-07	Character concatenation	✓
E021-08	UPPER and LOWER functions	✓
E021-09	TRIM function	✓
E021-10	Implicit casting among the fixed and variable length character string types	✓
E021-11	POSITION function	✓
E021-12	Character comparison	✓
E031	Identifiers	
E031-01	Delimited identifiers	✓
E031-02	Lower case identifiers	✓
E031-03	Trailing underscore	✓
E051	Basic query specification	
E051-01	SELECT DISTINCT	✓
E051-02	GROUP BY clause	✓
E051-04	GROUP BY can contain columns not in <select list>	✓
E051-05	Select list items can be renamed	✓

E051-06	HAVING clause	✓
E051-07	Qualified * in select list	✓
E051-08	Correlation names in the FROM clause	✓
E051-09	Rename columns in the FROM clause	✓
E061	Basic predicates and search conditions	
E061-01	Comparison predicate	✓
E061-02	BETWEEN predicate	✓
E061-03	IN predicate with list of values	✓
E061-04	LIKE predicate	✓
E061-05	LIKE predicate: ESCAPE clause	✓
E061-06	NULL predicate	✓
E061-07	Quantified comparison predicate	✓
E061-08	EXISTS predicate	✓
E061-09	Subqueries in comparison predicate	✓
E061-11	Subqueries in IN predicate	✓
E061-12	Subqueries in quantified comparison predicate	✓
E061-13	Correlated subqueries	✓
E061-14	Search condition	✓
E071	Basic query expressions	
E071-01	UNION DISTINCT table operator	✓
E071-02	UNION ALL table operator	✓
E071-03	EXCEPT DISTINCT table operator	
E071-05	Columns combined via table operators need not have exactly the same data type.	✓
E071-06	Table operators in subqueries	✓
E081	Basic Privileges	
E081-01	SELECT privilege at the table level	
E081-02	DELETE privilege	
E081-03	INSERT privilege at the table level	
E081-04	UPDATE privilege at the table level	
E081-05	UPDATE privilege at the column level	
E081-06	REFERENCES privilege at the table level	
E081-07	REFERENCES privilege at the column level	
E081-08	WITH GRANT OPTION	
E081-09	USAGE privilege	
E081-10	EXECUTE privilege	
E091	Set functions	

E091-01	AVG	✓
E091-02	COUNT	✓
E091-03	MAX	✓
E091-04	MIN	✓
E091-05	SUM	✓
E091-06	ALL quantifier	✓
E091-07	DISTINCT quantifier	✓
E101	Basic data manipulation	
E101-01	INSERT statement	✓
E101-03	Searched UPDATE statement	✓
E101-04	Searched DELETE statement	✓
E111	Single row SELECT statement	
E121	Basic cursor support	
E121-01	DECLARE CURSOR	
E121-02	ORDER BY columns need not be in select list	✓
E121-03	Value expressions in ORDER BY clause	
E121-04	OPEN statement	
E121-06	Positioned UPDATE statement	
E121-07	Positioned DELETE statement	
E121-08	CLOSE statement	
E121-10	FETCH statement: implicit NEXT	
E121-17	WITH HOLD cursors	
E131	Null value support (nulls in lieu of values)	✓
E141	Basic integrity constraints	
E141-01	NOT NULL constraints	✓
E141-02	UNIQUE constraints of NOT NULL columns	✓
E141-03	PRIMARY KEY constraints	✓
E141-04	Basic FOREIGN KEY constraint with the NO ACTION default for both referential delete action and referential update action.	✓
E141-06	CHECK constraints	✓
E141-07	Column defaults	✓
E141-08	NOT NULL inferred on PRIMARY KEY	✓
E141-10	Names in a foreign key can be specified in any order	✓
E151	Transaction support	
E151-01	COMMIT statement	✓
E151-02	ROLLBACK statement	✓
E152	Basic SET TRANSACTION statement	

E152-01	SET TRANSACTION statement: ISOLATION LEVEL SERIALIZABLE clause	
E152-02	SET TRANSACTION statement: READ ONLY and READ WRITE clauses	
E153	Updatable queries with subqueries	✓
E161	SQL comments using leading double minus	✓
E171	SQLSTATE support	
E182	Module language	
F021	Basic information schema	
F021-01	COLUMNS view	
F021-02	TABLES view	
F021-03	VIEWS view	
F021-04	TABLE_CONSTRAINTS view	
F021-05	REFERENTIAL_CONSTRAINTS view	
F021-06	CHECK_CONSTRAINTS view	
F031	Basic schema manipulation	
F031-01	CREATE TABLE statement to create persistent base tables	✓
F031-02	CREATE VIEW statement	✓
F031-03	GRANT statement	
F031-04	ALTER TABLE statement: ADD COLUMN clause	✓
F031-13	DROP TABLE statement: RESTRICT clause	✓
F031-16	DROP VIEW statement: RESTRICT clause	✓
F031-19	REVOKE statement: RESTRICT clause	
F041	Basic joined table	
F041-01	Inner join (but not necessarily the INNER keyword)	✓
F041-02	INNER keyword	✓
F041-03	LEFT OUTER JOIN	✓
F041-04	RIGHT OUTER JOIN	✓
F041-05	Outer joins can be nested	✓
F041-07	The inner table in a left or right outer join can also be used in an inner join	✓
F041-08	All comparison operators are supported (rather than just =)	✓
F051	Basic date and time	
F051-01	DATE data type (including support of DATE literal)	✓
F051-02	TIME data type (including support of TIME literal)	✓
F051-03	TIMESTAMP data type (including support of TIMESTAMP literal)	✓
F051-04	Comparison predicate on DATE, TIME, and TIMESTAMP data types	✓
F051-05	Explicit CAST between datetime types and character string types	✓

F051-06	CURRENT_DATE	✓
F051-07	LOCALTIME	✓
F051-08	LOCALTIMESTAMP	✓
F081	UNION and EXCEPT in views	✓
F131	Grouped operations	
F131-01	WHERE, GROUP BY, and HAVING clauses supported in queries with grouped views	✓
F131-02	Multiple tables supported in queries with grouped views	✓
F131-03	Set functions supported in queries with grouped views	✓
F131-04	Subqueries with GROUP BY and HAVING clauses and grouped views	✓
F131-05	Single row SELECT with GROUP BY and HAVING clauses and grouped views	
F181	Multiple module support	
F201	CAST function	✓
F221	Explicit defaults	✓
F261	CASE expression	
F261-01	Simple CASE	✓
F261-02	Searched CASE	✓
F261-03	NULLIF	✓
F261-04	COALESCE	✓
F311	Schema definition statement	
F471	Scalar subquery values	✓
F481	Expanded NULL predicate	✓
F501	Features and conformance views	
F501-01	SQL_FEATURES view	
F501-02	SQL_SIZING view	
F501-03	SQL_LANGUAGES view	
F812	Basic flagging	
S011	Distinct data types	
S011-01	USER_DEFINED_TYPES view	
T321	Basic SQL-invoked routines	
T321-01	User-defined functions with no overloading	✓
T321-02	User-defined stored procedures with no overloading	✓
T321-03	Function invocation	✓
T321-04	CALL statement	✓
T321-05	RETURN statement	✓
T321-06	ROUTINES view	

T321-07	PARAMETERS view	
T631	IN predicate with one list element	✓

24.2 Additional SQL:2003 Features

Additional SQL:2003 Features

Additional SQL:2003 features supported in NexusDB SQL

Feature ID	Feature description
B128	Routine language SQL
F033	ALTER TABLE statement: DROP COLUMN clause
F052	Intervals and datetime arithmetic
F171	Multiple schemas per user
F191	Referential delete actions
F222	INSERT statement: DEFAULT VALUES clause
F271	Compound character literals
F281	LIKE enhancements
F291	UNIQUE predicate
F302	INTERSECT table operator
F302-01	INTERSECT DISTINCT table operator
F302-02	INTERSECT ALL table operator
F304	EXCEPT ALL table operator
F381	Extended schema manipulation
F381-01	ALTER TABLE statement: ALTER COLUMN clause
F381-02	ALTER TABLE statement: ADD CONSTRAINT clause
F381-03	ALTER TABLE statement: DROP CONSTRAINT clause
F391	Long identifiers
F401	Extended joined table
F401-01	NATURAL JOIN
F401-02	FULL OUTER JOIN
F401-04	CROSS JOIN
F421	National character
F491	Constraint management
F531	Temporary tables
F561	Full value expressions

F571	Truth value tests
F591	Derived tables
F671	Subqueries in CHECK constraints
F672	Retrospective check constraints
F692	Enhanced collation support
F701	Referential update actions
F741	Referential MATCH types
T031	BOOLEAN data type
T041	Basic LOB data type support
T041-01	BLOB data type
T041-02	CLOB data type
T041-03	POSITION, LENGTH, LOWER, TRIM, UPPER, and SUBSTRING functions for LOB data types
T041-04	Concatenation of LOB data types
T071	BIGINT data type
T171	LIKE clause in table definition
T172	AS subquery clause in table definition
T191	Referential action RESTRICT
T201	Comparable data types for referential constraints
T211	Basic trigger capability
T211-01	Triggers activated on UPDATE, INSERT, or DELETE of one base table.
T211-02	BEFORE triggers
T211-03	AFTER triggers
T211-04	FOR EACH ROW triggers
T211-05	Ability to specify a search condition that shall be True before the trigger is invoked.
T241	START TRANSACTION statement
T326	Table functions
T351	Bracketed SQL comments /*...*/ comments)
T441	ABS and MOD functions
T501	Enhanced EXISTS predicate
T551	Optional key words for default syntax
T591	UNIQUE constraints of possibly null columns
T621	Enhanced numeric functions
T651	SQL-schema statements in SQL routines
T653	SQL-schema statements in external routines
P002	Computational completeness
P002-01	<compound statement>

P002-04	<SQL variable declaration>
P002-05	<assignment statement>
P002-07	<if statement>
P002-08	<iterate statement>
P002-09	<leave statement>
P002-11	<repeat statement>
P002-12	<while statement>
P002-14	<signal statement>

25 Symbol Reference

25.1 Classes

25.1.1 EnxAbortStateChange Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxAbortStateChange Class

nxllComponent

Pascal

```
public EnxAbortStateChange = class(EAbort);
```

File

nxllComponent

Remarks

Exception type used for critical state errors.

See Also

nxllComponent

You are here: Symbol Reference > Classes > EnxAbortStateChange Class

25.1.2 EnxAdvAutoIncDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxAdvAutoIncDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxAdvAutoIncDescriptorException = class(EnxBaseAutoIncDescriptorException);
```

File

nxsdDataDictionary

Remarks

This is the Exception class for the Advanced AutoInc Descriptor. All errors raised because of a problem in a Advanced AutoInc Descriptor class use this or a descendent exception class.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Classes > EnxAdvAutoIncDescriptorException Class

25.1.3 EnxBaseAutoIncDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxBaseAutoIncDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxBaseAutoIncDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the Base AutoInc Descriptor. All errors raised because of a problem in a Base AutoInc Descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxBaseAutoIncDescriptorException Class

25.1.4 EnxBaseBlobDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseBlobDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxBaseBlobDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the Blob Descriptor. All errors raised because of a problem in a Blob Descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxBaseBlobDescriptorException Class

25.1.5 EnxBaseBlockHeapDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseBlockHeapDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxBaseBlockHeapDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the Block Heap Descriptor. All errors raised because of a problem in a Block Heap Descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxBaseBlockHeapDescriptorException Class

25.1.6 EnxBaseCustomDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseCustomDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxBaseCustomDescriptorException = class(EnxDictionaryItemException);
```

File

nxsdDataDictionary

Remarks

This is the Exception class for Base Custom Descriptors. All errors raised because of a problem in a Base Custom Descriptor class use this or a descendent exception class.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Classes > EnxBaseCustomDescriptorException Class

25.1.7 EnxBaseDefaultValueDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseDefaultValueDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxBaseDefaultValueDescriptorException = class(EnxDictionaryItemException);
```

File

nxsdDataDictionary

Remarks

This is the Exception class for the Default Value Descriptor. All errors raised because of a problem in a Default Value Descriptor class use this or a descendent exception class.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Classes > EnxBaseDefaultValueDescriptorException Class

25.1.8 EnxBaseDictionaryException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseDictionaryException Class

nxsdDataDictionary

Pascal

```
public EnxBaseDictionaryException = class(EnxBaseException);
```

File

nxsdDataDictionary

Remarks

This is the base Exception class for the various data dictionary exceptions.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxBaseDictionaryException Class

25.1.9 EnxBaseFieldValidationDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxBaseFieldValidationDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxBaseFieldValidationDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Validation Descriptor. All errors raised because of a problem in a Validation Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxBaseFieldValidationDescriptorException Class

25.1.10 EnxBaseHeapDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxBaseHeapDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxBaseHeapDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Heap Descriptor. All errors raised because of a problem in a Heap Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxBaseHeapDescriptorException Class

25.1.11 EnxBaseIndicesDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxBaseIndicesDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxBaseIndicesDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Indices descriptor. All errors raised because of a problem in a Indices descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxBaseIndicesDescriptorException Class

25.1.12 EnxBaseKeyDescriptorException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxBaseKeyDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxBaseKeyDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Key Descriptor. All errors raised because of a problem in a Key Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxBaseKeyDescriptorException Class

25.1.13 EnxBaseLogException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxBaseLogException Class

[nxllComponent](#)**Pascal**

```
public EnxBaseLogException = class(EnxRegisterableComponentException);
```

File[nxllComponent](#)**Remarks**

Special exception type for all logging classes.

See Also[nxllComponent](#)

You are here: Symbol Reference > Classes > EnxBaseLogException Class

25.1.14 EnxBaseRecordCompressionDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseRecordCompressionDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxBaseRecordCompressionDescriptorException = class(EnxDictionaryItemException);
```

File

nxsdDataDictionary

Remarks

This is the Exception class for the Record Compression Descriptor. All errors raised because of a problem in a Record Compression Descriptor class use this or a descendent exception class.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Classes > EnxBaseRecordCompressionDescriptorException Class

25.1.15 EnxBaseRecordDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseRecordDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxBaseRecordDescriptorException = class(EnxDictionaryItemException);
```

File

nxsdDataDictionary

Remarks

This is the Exception class for the Record Descriptor. All errors raised because of a problem in a Record Descriptor class use this or a descendent exception class.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Classes > EnxBaseRecordDescriptorException Class

25.1.16 EnxBaseStreamDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseStreamDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxBaseStreamDescriptorException = class(EnxDictionaryItemException);
```

File

nxsdDataDictionary

Remarks

This is the Exception class for the Stream Descriptor. All errors raised because of a problem in a Stream Descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxBaseStreamDescriptorException Class

25.1.17 EnxBaseTableDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxBaseTableDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxBaseTableDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for Table Descriptors. All errors raised because of a problem in a Table Descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxBaseTableDescriptorException Class

25.1.18 EnxCommandHandlerException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxCommandHandlerException Class

[nxllTransport](#)

Pascal

```
public EnxCommandHandlerException = class(EnxStateComponentException);
```

File

[nxllTransport](#)

Remarks

The base exception type used by the command handler.

See Also

[nxllTransport](#)

You are here: Symbol Reference > Classes > EnxCommandHandlerException Class

25.1.19 EnxCompKeyDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxCompKeyDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxCompKeyDescriptorException = class(EnxBaseKeyDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the exception class used for Compound Key descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxCompKeyDescriptorException Class

25.1.20 EnxComponentException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxComponentException Class

[nxllComponent | Members](#)

Pascal

```
public EnxComponentException = class(EnxBaseException);
```

File

[nxllComponent](#)

Remarks

A special Exception class that can have additional information about TnxComponent.

See Also

[nxllComponent](#)
[Members](#)

You are here: Symbol Reference > Classes > EnxComponentException Class

25.1.20.1 Constructors

25.1.20.1.1 nxcCreate

25.1.20.1.1.1 nxcCreate Constructor (TComponent, PResStringRec)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, PResStringRec)

[EnxComponentException Class](#)

Pascal

```
public constructor nxcCreate(
    aComponent: TComponent;
    aMsgRes: PResStringRec
); overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, PResStringRec)

25.1.20.1.1.2 nxcCreate Constructor (TComponent, PResStringRec, array of const)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, PResStringRec, array of const)

EnxComponentException Class

Pascal

```
public constructor nxcCreate(
    aComponent: TComponent;
    aMsgRes: PResStringRec;
    const aArgs: array of const
); overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, PResStringRec, array of const)

25.1.20.1.1.3 nxcCreate Constructor (TComponent, TnxResult)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, TnxResult)

EnxComponentException Class

Pascal

```
public constructor nxcCreate(
    aComponent: TComponent;
    aErrorCode: TnxResult
); overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, TnxResult)

25.1.20.1.1.4 nxcCreate Constructor (TComponent, TnxResult, PResStringRec)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, TnxResult, PResStringRec)

EnxComponentException Class

Pascal

```
public constructor nxcCreate(
```

```
aComponent: TComponent;
aErrorCode: TnxResult;
aAltMsgRes: PResStringRec
) ; overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, TnxResult, PResStringRec)

25.1.20.1.1.5 nxcCreate Constructor (TComponent, TnxResult, PResStringRec, array of const)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, TnxResult, PResStringRec, array of const)

[EnxComponentException Class](#)

Pascal

```
public constructor nxcCreate(
  aComponent: TComponent;
  aErrorCode: TnxResult;
  aAltMsgRes: PResStringRec;
  const aArgs: array of const
) ; overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, TnxResult, PResStringRec, array of const)

25.1.20.1.1.6 nxcCreate Constructor (TComponent, TnxResult, string)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, TnxResult, string)

[EnxComponentException Class](#)

Pascal

```
public constructor nxcCreate(
  aComponent: TComponent;
  aErrorCode: TnxResult;
  const aAltMsg: string
) ; overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, TnxResult, string)

25.1.20.1.1.7 nxcCreate Constructor (TComponent, TnxResult, string, array of const)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, TnxResult, string, array of const)

EnxComponentException Class

Pascal

```
public constructor nxcCreate(
    aComponent: TComponent;
    aErrorCode: TnxResult;
    const aAltMsg: string;
    const aArgs: array of const
); overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, TnxResult, string, array of const)

25.1.20.1.1.8 nxcCreate Constructor (TComponent, string)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, string)

EnxComponentException Class

Pascal

```
public constructor nxcCreate(
    aComponent: TComponent;
    const aMsg: string
); overload;
```

Remarks

constructor

See Also

[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, string)

25.1.20.1.1.9 nxcCreate Constructor (TComponent, string, array of const)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxComponentException.nxcCreate Constructor (TComponent, string, array of const)

EnxComponentException Class

Pascal

```
public constructor nxcCreate(
    aComponent: TComponent;
    const aMsg: string;
    const aArgs: array of const
); overload;
```

Remarks

constructor

See Also[EnxComponentException Class](#)

You are here: Symbol Reference > Classes > EnxComponentException Class > Constructors > nxcCreate > nxcCreate Constructor (TComponent, string, array of const)

25.1.21 EnxConstDefaultValueDescriptorException Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**EnxConstDefaultValueDescriptorException Class**[nxsdDataDictionary](#)**Pascal**

```
public EnxConstDefaultValueDescriptorException = class(EnxBaseDefaultValueDescriptorException)
```

File[nxsdDataDictionary](#)**Remarks**

The Exception class for the Constant Default Value Descriptor.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxConstDefaultValueDescriptorException Class

25.1.22 EnxCustomDescsDescriptorException Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**EnxCustomDescsDescriptorException Class**[nxsdDataDictionary](#)**Pascal**

```
public EnxCustomDescsDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for Custom Descriptor Descriptors. All errors raised because of a problem in a Custom Descriptor Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxCustomDescsDescriptorException Class

25.1.23 EnxDataDictionaryException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxDataDictionaryException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxDataDictionaryException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for general, not descriptor based, exceptions of the Data Dictionary.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxDataDictionaryException Class

25.1.24 EnxDictionaryItemException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxDictionaryItemException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxDictionaryItemException = class(EnxBaseDictionaryException);
```

File

[nxsdDataDictionary](#)

Remarks

All exceptions for descriptors are inherited from this.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxDictionaryItemException Class

25.1.25 EnxExtTextKeyFieldDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxExtTextKeyFieldDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxExtTextKeyFieldDescriptorException = class(EnxTextKeyFieldDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the exception class used for Extended Text Key Field descriptors.

See Also

[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxExtTextKeyFieldDescriptorException Class

25.1.26 EnxFieldDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxFieldDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxFieldDescriptorException = class(EnxLocaleizedDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Field Descriptor. All errors raised because of a problem in a Field Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxFieldDescriptorException Class

25.1.27 EnxFIELDSDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxFIELDSDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxFIELDSDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Fields descriptor. All errors raised because of a problem in a Fields descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxFIELDSDescriptorException Class

25.1.28 EnxFieldValidationsDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxFieldValidationsDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxFieldValidationsDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the Validations Descriptor. All errors raised because of a problem in a Validations Descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxFieldValidationsDescriptorException Class

25.1.29 EnxFilDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxFilDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxFilDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the File descriptor. All errors raised because of a problem in a File descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxFilDescriptorException Class

25.1.30 EnxFilesDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxFilesDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxFilesDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the Files descriptor. All errors raised because of a problem in a Files descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxFilesDescriptorException Class

25.1.31 EnxHeapBlobDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxHeapBlobDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxHeapBlobDescriptorException = class(EnxBaseBlobDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the exception class used for the Heap Blob Descriptor.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxHeapBlobDescriptorException Class

25.1.32 EnxHeapRecordDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxHeapRecordDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxHeapRecordDescriptorException = class(EnxBaseRecordDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the exception class used for the Heap Record Descriptor.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxHeapRecordDescriptorException Class

25.1.33 EnxIndexDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxIndexDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxIndexDescriptorException = class(EnxDictionaryItemException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the Index Descriptor. All errors raised because of a problem in a Key Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxIndexDescriptorException Class

25.1.34 EnxKeyFieldDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**EnxKeyFieldDescriptorException Class**[nxsdDataDictionary](#)**Pascal**

```
public EnxKeyFieldDescriptorException = class(EnxLocaleizedDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Key Field Descriptor. All errors raised because of a problem in a Key Field Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxKeyFieldDescriptorException Class

25.1.35 EnxLocaleDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**EnxLocaleDescriptorException Class**[nxsdDataDictionary](#)**Pascal**

```
public EnxLocaleDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Locale descriptor. All errors raised because of a problem in a Locale descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxLocaleDescriptorException Class

25.1.36 EnxLocaleizedDictionaryItemException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**EnxLocaleizedDictionaryItemException Class**[nxsdDataDictionary](#)**Pascal**

```
public EnxLocaleizedDictionaryItemException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for a localized dictionary item. All errors raised because of a problem in a localized dictionary item class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxLocalizedDictionaryItemException Class

25.1.37 EnxLoggableComponentException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxLoggableComponentException Class

[nxlComponent](#)**Pascal**

```
public EnxLoggableComponentException = class(EnxRegisterableComponentException);
```

File[nxlComponent](#)**Remarks**

Special exception type for logable components.

See Also[nxlComponent](#)

You are here: Symbol Reference > Classes > EnxLoggableComponentException Class

25.1.38 EnxMainIndicesDescriptorException Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

EnxMainIndicesDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxMainIndicesDescriptorException = class(EnxBaseIndicesDescriptorException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Main Indices descriptor. All errors raised because of a problem in a Main Indices descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxMainIndicesDescriptorException Class

25.1.39 EnxMappedKeyDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxMappedKeyDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxMappedKeyDescriptorException = class(EnxBaseKeyDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the exception class used for Mapped Key descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxMappedKeyDescriptorException Class

25.1.40 EnxMinMaxValidationDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxMinMaxValidationDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxMinMaxValidationDescriptorException = class(EnxBaseFieldValidationDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for the Min/Max Validation Descriptor. All errors raised because of a problem in a Min/Max Validation Descriptor class use this or a descendent exception class.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxMinMaxValidationDescriptorException Class

25.1.41 EnxNestedTableDescriptorException Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

EnxNestedTableDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxNestedTableDescriptorException = class(EnxBaseTableDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Exception class for Nested Table Descriptors. All errors raised because of a problem in a Nested Table Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxNestedTableDescriptorException Class

25.1.42 EnxNoChangeValidationDescriptorException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxNoChangeValidationDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxNoChangeValidationDescriptorException = class(EnxBaseFieldValidationDescriptorException)
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the No Change Validation Descriptor. All errors raised because of a problem in a No Change Validation Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxNoChangeValidationDescriptorException Class

25.1.43 EnxNotNullCompKeyDescriptorException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxNotNullCompKeyDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxNotNullCompKeyDescriptorException = class(EnxCompKeyDescriptorException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the exception class used for Compound Key descriptors which don't allow null fields.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Classes > EnxNotNullCompKeyDescriptorException Class

25.1.44 EnxPluginCommandHandlerException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxPluginCommandHandlerException Class

[nxllTransport](#)**Pascal**

```
public EnxPluginCommandHandlerException = class(EnxPluginException);
```

File

nxllTransport

Remarks

The base Exception class for Plug in Command handlers.

See Also

nxllTransport

You are here: Symbol Reference > Classes > EnxPluginCommandHandlerException Class

25.1.45 EnxRefKeyDescriptorException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxRefKeyDescriptorException Class

nxsdDataDictionary

Pascal

```
public EnxRefKeyDescriptorException = class(EnxBaseKeyDescriptorException);
```

File

nxsdDataDictionary

Remarks

This is the exception class used for Ref Key descriptors.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Classes > EnxRefKeyDescriptorException Class

25.1.46 EnxRegisterableComponentException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxRegisterableComponentException Class

nxllComponent

Pascal

```
public EnxRegisterableComponentException = class(EnxComponentException);
```

File

nxllComponent

Remarks

Special exception type for TnxRegisterableComponent.

See Also

nxllComponent

You are here: Symbol Reference > Classes > EnxRegisterableComponentException Class

25.1.47 EnxSqlCheckValidationDescriptorException Class

NexusDB V2 VCL Reference[Contents | Index](#)

EnxSqlCheckValidationDescriptorException Class

[nxsdDataDictionary](#)**Pascal**

```
public EnxSqlCheckValidationDescriptorException = class(EnxBaseFieldValidationDescriptorException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for the Sql Check Validation Descriptor. All errors raised because of a problem in a Sql Check Validation Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxSqlCheckValidationDescriptorException Class

25.1.48 EnxStateComponentException Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**EnxStateComponentException Class**[nxllComponent](#)**Pascal**

```
public EnxStateComponentException = class(EnxLoggableComponentException);
```

File[nxllComponent](#)**Remarks**

Exception type used for State components.

See Also[nxllComponent](#)

You are here: Symbol Reference > Classes > EnxStateComponentException Class

25.1.49 EnxTablesDescriptorException Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**EnxTablesDescriptorException Class**[nxsdDataDictionary](#)**Pascal**

```
public EnxTablesDescriptorException = class(EnxDictionaryItemException);
```

File[nxsdDataDictionary](#)**Remarks**

This is the Exception class for Tables Descriptors. All errors raised because of a problem in a Tables Descriptor class use this or a descendent exception class.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxTablesDescriptorException Class

25.1.50 EnxTextKeyFieldDescriptorException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxTextKeyFieldDescriptorException Class

[nxsdDataDictionary](#)

Pascal

```
public EnxTextKeyFieldDescriptorException = class(EnxKeyFieldDescriptorException);
```

File

[nxsdDataDictionary](#)

Remarks

This is the exception class used for Text Key Field descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > EnxTextKeyFieldDescriptorException Class

25.1.51 EnxTransportException Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxTransportException Class

[nxllTransport](#)

Pascal

```
public EnxTransportException = class(EnxStateComponentException);
```

File

[nxllTransport](#)

Remarks

This is the exception type for transport based exceptions.

See Also

[nxllTransport](#)

You are here: Symbol Reference > Classes > EnxTransportException Class

25.1.52 TnxAbstractCursor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAbstractCursor = class(TnxAbstractTimeoutObject, InxFieldsSource);
```

File

[nxsdServerEngine](#)

Remarks

The abstract declaration of a Server cursor. All actual cursor implementations must be descendent from this class. This is the server side version of TnxCursor.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class

25.1.52.1 Constructors

25.1.52.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractCursor.Create Constructor

TnxAbstractCursor Class

Pascal

```
public constructor Create(
    aDatabase: TnxAbstractDatabase;
    aTimeout: TnxWord32
);
```

Parameters

Parameters	Description
aDatabase	the database object the cursor belongs to.
aTimeout	the timeout for this cursor in msec.

Remarks

Constructor

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Constructors > Create Constructor

25.1.52.2 Destructors

25.1.52.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractCursor.Destroy Destructor

TnxAbstractCursor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Destructors > Destroy Destructor

25.1.52.3 Methods

25.1.52.3.1 AfterConstruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractCursor.AfterConstruction Method

TnxAbstractCursor Class

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

see TObject.AfterConstruction

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > AfterConstruction Method

25.1.52.3.2 AutoIncGet Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractCursor.AutoIncGet Method

TnxAbstractCursor Class

Pascal

```
public function AutoIncGet(
    out aValue: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

- autoinc -

This function returns the next AutoInc value for the current table.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > AutoIncGet Method

25.1.52.3.3 AutoIncSet Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractCursor.AutoIncSet Method

TnxAbstractCursor Class

Pascal

```
public function AutoIncSet(
    aValue: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

This function sets the next AutoInc value for the current table.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > AutoIncSet Method

25.1.52.3.4 BeforeDestruction Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BeforeDestruction Method

[TnxAbstractCursor Class](#)

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeConstruction

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BeforeDestruction Method

25.1.52.3.5 BlobCreate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BlobCreate Method

[TnxAbstractCursor Class](#)

Pascal

```
public function BlobCreate(
    aFieldNo: TnxInt32;
    out aBlobNr: TnxInt64
): TnxResult; virtual; abstract;
```

Remarks

This function creates a new blob and returns the new blob reference number in aBlobNr. This number is unique for any blob and is needed to access the blob.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobCreate Method

25.1.52.3.6 BlobCreateFile Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BlobCreateFile Method

[TnxAbstractCursor Class](#)

Pascal

```
public function BlobCreateFile(
    aFieldNo: TnxInt32;
```

```
    const aFileName: string;
    out aBlobNr: TnxInt64
  ): TnxResult; virtual; abstract;
```

Remarks

This creates a new FileBlob and returns the new blob reference number in aBlobNr. This number is unique for any blob and is needed to access the blob.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobCreateFile Method

25.1.52.3.7 BlobDelete Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BlobDelete Method

[TnxAbstractCursor Class](#)

Pascal

```
public function BlobDelete(
  aFieldNo: TnxInt32;
  const aBlobNr: TnxInt64
): TnxResult; virtual; abstract;
```

Remarks

This function deletes the blob with the given aBlobNr.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobDelete Method

25.1.52.3.8 BlobFree Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BlobFree Method

[TnxAbstractCursor Class](#)

Pascal

```
public function BlobFree(
  aFieldNo: TnxInt32;
  const aBlobNr: TnxInt64
): TnxResult; virtual; abstract;
```

Remarks

This function frees all temporary buffers for this blob.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobFree Method

25.1.52.3.9 BlobGetLength Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.BlobGetLength Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function BlobGetLength(
    aFieldNo: TnxInt32;
    const aBlobNr: TnxInt64;
    out aLength: TnxWord32;
    aReadOrg: Boolean
): TnxResult; virtual; abstract;
```

Remarks

This function returns the length of the blob in aLength as bytes.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobGetLength Method

25.1.52.3.10 BlobModified Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.BlobModified Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function BlobModified(
    aFieldNo: TnxInt32;
    const aBlobNr: TnxInt64;
    out aModified: Boolean
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aFieldNo	of the field in FieldsDescriptor
aBlobNr	the blob reference number
aModified	will be set to true if the blob has changed

Remarks

Checks if the given Blob Buffer was modified

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobModified Method

25.1.52.3.11 BlobRead Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.BlobRead Method**[TnxAbstractCursor Class](#)

```
Pascal
public function BlobRead(
    aFieldNo: TnxInt32;
    const aBlobNr: TnxInt64;
    aOffset: TnxWord32;
    aLen: TnxWord32;
    aStream: TStream;
    aReadOrg: Boolean
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aBlobNr	the blob reference number
aOffset	a offset in bytes where to start to read
aLen	the number of bytes to read
aBlob	the buffer to receive the data
aBytesRead	the number of bytes returned

Remarks

This function reads data from a blob.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobRead Method

25.1.52.3.12 BlobTruncate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BlobTruncate Method

[TnxAbstractCursor Class](#)

```
Pascal
public function BlobTruncate(
    aFieldNo: TnxInt32;
    const aBlobNr: TnxInt64;
    aBlobLength: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

This function truncates the given blob at aBlobLength bytes.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > BlobTruncate Method

25.1.52.3.13 BlobWrite Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BlobWrite Method

[TnxAbstractCursor Class](#)

```
Pascal
```

```
public function BlobWrite(
    aFieldNo: TnxInt32;
    const aBlobNr: TnxInt64;
    aOffset: TnxWord32;
    aLen: TnxWord32;
    const aBlob
) : TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aBlobNr	the blob reference number
aOffset	a offset in bytes where to start to write
aLen	the number of bytes to write
aBlob	the data to write

Remarks

This function writes data to a blob.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > [TnxAbstractCursor Class](#) > Methods > [BlobWrite Method](#)

25.1.52.3.14 BookmarkAsVariant Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.BookmarkAsVariant Method

[TnxAbstractCursor Class](#)

Pascal

```
public function BookmarkAsVariant(
    aBookmark: PnxByteArray;
    aSize: TnxWord32;
    out aVariant: Variant
) : TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aBookmark	the data of the bookmark
aSize	size of the bookmark data
aVariant	the new variant value returned

Remarks

Return a given bookmark as Variant

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > [TnxAbstractCursor Class](#) > Methods > [BookmarkAsVariant Method](#)

25.1.52.3.15 ChangePassword Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.ChangePassword Method

TnxAbstractCursor Class

```
Pascal
public function ChangePassword(
    const aNewKey: string
): TnxResult; virtual; abstract;
```

Remarks

This function changes the password of the table. It fails if the table doesn't us an encryption engine.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > ChangePassword Method

25.1.52.3.16 CompareBookmarks Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.CompareBookmarks Method

TnxAbstractCursor Class

```
Pascal
public function CompareBookmarks(
    aBookmark1: PnxByteArray;
    aBookmark2: PnxByteArray;
    aSize: TnxWord32;
    out aCompResult: TnxValueRelationship
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aBookmark1	first Bookmark
aBookmark2	second Bookmark
aSize	the size of the bookmarks
aCompResult	the var that will receive the result

Remarks

This function compares two bookmarks.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > CompareBookmarks Method

25.1.52.3.17 CompareKeys Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.CompareKeys Method

TnxAbstractCursor Class

```
Pascal
public function CompareKeys(
    aFieldCount: Integer;
    aPartialLen: Integer;
```

```

    aDirectKey1: Boolean;
    aKeyData1: PnxByteArray;
    aDirectKey2: Boolean;
    aKeyData2: PnxByteArray;
    out aResult: TnxValueRelationship
  ): TnxResult; virtual; abstract;

```

Parameters

Parameters	Description
aFieldCount	number of fields to be compared. 0 = all fields
aPartialLen	number of characters yo compare for the last field. 0 = no partial compare
aDirectKey1	True = passed in data is directly a key, False = passed in data is a record and the key is extracted.
aKeyData1	The data according to the format defined by aDirectKey1.
aDirectKey2	True = passed in data is directly a key, False = passed in data is a record and the key is extracted.
aKeyData2	The data according to the format defined by aDirectKey2.
aResult	returns the result of the comparison.

Remarks

CompareKeys compares 2 keys.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > CompareKeys Method

25.1.52.3.18 CopyRecords Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.CopyRecords Method

[TnxAbstractCursor Class](#)

Pascal

```

public function CopyRecords(
  aDestCursor: TnxAbstractCursor;
  aBlobCopyMode: TnxBlobCopyMode;
  aMaxTransSize: Integer
): TnxResult; virtual; abstract;

```

Remarks

This function copies *all* visible (in regards of ranges and filters) records to the cursor given in aDestCursor. aMaxTransSize specifies the maximum number of dirty transaction bytes before the transaction is committed.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > CopyRecords Method

25.1.52.3.19 CopyRecordsEx Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.CopyRecordsEx Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function CopyRecordsEx(
    aDestCursor: TnxAbstractCursor;
    aBlobCopyMode: TnxBlobCopyMode;
    aMaxTransSize: Integer;
    aCheckValFields: Boolean
): TnxResult; virtual; abstract;
```

Remarks

This function copies *all* visible (in regards of ranges and filters) records to the cursor given in aDestCursor. aMaxTransSize specifies the maximum number of dirty transaction bytes before the transaction is committed. aCheckValFields determines whether a check is done for required fields having a value.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > CopyRecordsEx Method

25.1.52.3.20 DeleteRecords Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.DeleteRecords Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function DeleteRecords: TnxResult; virtual; abstract;
```

Remarks

This function deletes *all* visible (in regards of ranges and filters) records.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > DeleteRecords Method

25.1.52.3.21 Duplicate Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.Duplicate Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function Duplicate(
    aOpenMode: TnxOpenMode;
    out aNewCursor: TnxAbstractCursor
): TnxResult; virtual; abstract;
```

Remarks

This function clones the current cursor and returns a new instance in aNewCursor. The new cursor can be opened with a different mode by specifying the aOpenMode accordingly.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > Duplicate Method

25.1.52.3.22 FilterActivate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.FilterActivate Method

[TnxAbstractCursor Class](#)

Pascal

```
public function FilterActivate(
    aFilterID: TnxFilterID
): TnxResult; virtual; abstract;
```

Remarks

This function sets the given Filter to active.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > FilterActivate Method

25.1.52.3.23 FilterAddCallback Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.FilterAddCallback Method

[TnxAbstractCursor Class](#)

Pascal

```
public function FilterAddCallback(
    aCookie: Integer;
    aCallback: TnxFilterCallback;
    out aFilterID: TnxFilterID
): TnxResult; virtual;
```

Remarks

Adds a new callback function to the filter valuation list.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > FilterAddCallback Method

25.1.52.3.24 FilterAddCustom Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.FilterAddCustom Method

[TnxAbstractCursor Class](#)

Pascal

```
public function FilterAddCustom(
    aFilter: TnxBaseFilterDescriptor;
    out aFilterID: TnxFilterID
): TnxResult; virtual; abstract;
```

Remarks

Adds a new filter to the filter evaluation list.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > FilterAddCustom Method

25.1.52.3.25 FilterAddExpression Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.FilterAddExpression Method

[TnxAbstractCursor Class](#)

Pascal

```
public function FilterAddExpression(
    aExpression: pCANExpr;
    out aFilterID: TnxFilterID
): TnxResult; virtual;
```

Remarks

FilterAddExpression creates a new Filter instance for the given aExpression and returns its in aFilterID.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > FilterAddExpression Method

25.1.52.3.26 FilterDeactivate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.FilterDeactivate Method

[TnxAbstractCursor Class](#)

Pascal

```
public function FilterDeactivate(
    aFilterID: TnxFilterID
): TnxResult; virtual; abstract;
```

Remarks

This function sets the given Filter to inactive.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > FilterDeactivate Method

25.1.52.3.27 FilterRemove Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.FilterRemove Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function FilterRemove(
    aFilterID: TnxFilterID
): TnxResult; virtual; abstract;
```

Remarks

This function removes the given Filter from the internal list.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > FilterRemove Method

25.1.52.3.28 FilterSetTimeout Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.FilterSetTimeout Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function FilterSetTimeout(
    aTimeout: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

This function sets the timeout for the filter evaluation in msec.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > FilterSetTimeout Method

25.1.52.3.29 GetBookmark Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.GetBookmark Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function GetBookmark(
    aBookmark: PnxByteArray;
    aSize: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks**- bookmarks -**

This function creates a new bookmark Parameters: aBookmark - a buffer to receive the Bookmark data aSize - the size of the given buffer

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > GetBookmark Method

25.1.52.3.30 GetIndexID Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractCursor.GetIndexID Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function GetIndexID(
    out aIndexID: Integer
): TnxResult; virtual; abstract;
```

Remarks

This function returns the id of the current index.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > GetIndexID Method

25.1.52.3.31 GetRecordCount Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractCursor.GetRecordCount Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function GetRecordCount(
    out aRecCount: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

GetRecordCount returns the number of currently "visible" records. It obeys set ranges and filters.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > GetRecordCount Method

25.1.52.3.32 GetRecordCountAsync Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractCursor.GetRecordCountAsync Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function GetRecordCountAsync(
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; virtual; abstract;
```

Remarks

This function is an asynchronous version of GetRecordCount.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > [GetRecordCountAsync Method](#)

25.1.52.3.33 GetRecordCountEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.GetRecordCountEx Method

[TnxAbstractCursor Class](#)

Pascal

```
public function GetRecordCountEx(
    aOption: TnxRecordCountOption;
    out aRecCount: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

GetRecordCountEx returns the number of records or keys based on the value of aOption.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > [GetRecordCountEx Method](#)

25.1.52.3.34 GetRecordCountExAsync Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.GetRecordCountExAsync Method

[TnxAbstractCursor Class](#)

Pascal

```
public function GetRecordCountExAsync(
    aOption: TnxRecordCountOption;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; virtual; abstract;
```

Remarks

This function is an asynchronous version of GetRecordCountEx.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > [GetRecordCountExAsync Method](#)

25.1.52.3.35 KeyFilterActivate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.KeyFilterActivate Method

[TnxAbstractCursor Class](#)

```
Pascal
public function KeyFilterActivate(
    aKeyFilterID: TnxKeyFilterID
) : TnxResult; virtual; abstract;
```

Remarks

This function sets the given KeyFilter to active.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > KeyFilterActivate Method

25.1.52.3.36 KeyFilterAddCallback Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.KeyFilterAddCallback Method

[TnxAbstractCursor Class](#)

```
Pascal
public function KeyFilterAddCallback(
    aCookie: Integer;
    aCallback: TnxFilterCallback;
    out aKeyFilterID: TnxKeyFilterID
) : TnxResult; virtual;
```

Remarks

Adds a new callback function to the filter valuation list.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > KeyFilterAddCallback Method

25.1.52.3.37 KeyFilterAddCustom Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.KeyFilterAddCustom Method

[TnxAbstractCursor Class](#)

```
Pascal
public function KeyFilterAddCustom(
    aKeyFilter: TnxBaseFilterDescriptor;
    out aKeyFilterID: TnxKeyFilterID
) : TnxResult; virtual; abstract;
```

Remarks

Adds a new filter to the filter evaluation list.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > KeyFilterAddCustom Method

25.1.52.3.38 KeyFilterAddExpression Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.KeyFilterAddExpression Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function KeyFilterAddExpression(
    aExpression: pCANExpr;
    out aKeyFilterID: TnxKeyFilterID
): TnxResult; virtual;
```

Remarks

KeyFilterAddExpression creates a new KeyFilter instance for the given aExpression and returns its in aKeyFilterID.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > [KeyFilterAddExpression Method](#)

25.1.52.3.39 KeyFilterDeactivate Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.KeyFilterDeactivate Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function KeyFilterDeactivate(
    aKeyFilterID: TnxKeyFilterID
): TnxResult; virtual; abstract;
```

Remarks

This function sets the given KeyFilter to inactive.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > [KeyFilterDeactivate Method](#)

25.1.52.3.40 KeyFilterRemove Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractCursor.KeyFilterRemove Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function KeyFilterRemove(
    aKeyFilterID: TnxKeyFilterID
): TnxResult; virtual; abstract;
```

Remarks

This function removes the given KeyFilter from the internal list.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > KeyFilterRemove Method

25.1.52.3.41 LookupByID Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractCursor.LookupByID Method

TnxAbstractCursor Class

Pascal

```
public class function LookupByID(
  aSessionID: TnxSessionID;
  aCursorID: TnxCursorID;
  out aCursor
): TnxResult;
```

Parameters

Parameters	Description
aDatabaseID	the ID to look for
aDatabase	a var that will hold the class if found

Remarks

This is the class function that returns the object with a certain ID.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > LookupByID Method

25.1.52.3.42 RangeReset Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractCursor.RangeReset Method

TnxAbstractCursor Class

Pascal

```
public function RangeReset: TnxResult; virtual; abstract;
```

Remarks

This function cancels the active range.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RangeReset Method

25.1.52.3.43 RangeSet Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractCursor.RangeSet Method

TnxAbstractCursor Class

Pascal

```
public function RangeSet(
    aFieldCount1: Integer;
    aPartialLen1: Integer;
    aDirectKey1: Boolean;
    aKeyData1: PnxByteArray;
    aKeyIncl1: Boolean;
    aFieldCount2: Integer;
    aPartialLen2: Integer;
    aDirectKey2: Boolean;
    aKeyData2: PnxByteArray;
    aKeyIncl2: Boolean
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aFieldCount1	For start of the range, number of fields to be compared. 0 = all fields
aPartialLen1	For start of the range, number of characters yo compare for the last field. 0 = no partial compare
aDirectKey1	For start of the range, True = passed in data is directly a key, False = passed in data is a record and the key is extracted.
aKeyData1	For start of the range, The data according to the format defined by aDirectKey1.
aKeyIncl1	For start of the range, does the range include the key or starts after the key?
aFieldCount2	For end of the range, number of fields to be compared. 0 = all fields
aPartialLen2	For end of the range, number of characters yo compare for the last field. 0 = no partial compare
aDirectKey2	For end of the range, True = passed in data is directly a key, False = passed in data is a record and the key is extracted.
aKeyData2	For end of the range, The data according to the format defined by aDirectKey2.
aKeyIncl2	For end of the range, does the range include the key or starts after the key?

Remarks

Sets a range on the current cursor

See Also

[TnxAbstractCursor Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxAbstractCursor Class](#) > [Methods](#) > [RangeSet Method](#)

25.1.52.3.44 RecNoGet Method

[NexusDB V2 VCL Reference](#)

TnxAbstractCursor.RecNoGet Method

[TnxAbstractCursor Class](#)

[Contents | Index](#)

```
Pascal
public function RecNoGet(
    out aRecNo: TnxWord32;
    aFlipOrder: Boolean
): TnxResult; virtual; abstract;
```

■ Remarks

Returns the RecNo (Position) of the current record according to the current Index, Range and Filter(s). (using filters/indices is not recommended if that table (subset) contains more than a few data pages of records!).

■ See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecNoGet Method

25.1.52.3.45 RecNoSet Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.RecNoSet Method

[TnxAbstractCursor Class](#)

```
Pascal
public function RecNoSet(
    aRecNo: TnxWord32;
    aFlipOrder: Boolean
): TnxResult; virtual; abstract;
```

■ Remarks

Sets the current position according to the RecNo

■ See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecNoSet Method

25.1.52.3.46 RecNoSupported Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.RecNoSupported Method

[TnxAbstractCursor Class](#)

```
Pascal
public function RecNoSupported: TnxResult; virtual; abstract;
```

■ Remarks

Returns DBIERR_NONE if RecNo is supported.

■ See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecNoSupported Method

25.1.52.3.47 RecordDelete Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordDelete Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordDelete(
    aData: PnxByteArray
): TnxResult; virtual; abstract;
```

Remarks

RecordDelete deletes the current record and returns a pointer to the then current record buffer. It needs to be able to acquire the given lock type for this function to be successful.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordDelete Method

25.1.52.3.48 RecordGet Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordGet Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordGet(
    aLockType: TnxLockType;
    aData: PnxByteArray
): TnxResult; virtual; abstract;
```

Remarks

RecordGet returns a pointer to the current record buffer. It needs to be able to acquire the given lock type for this function to be successful.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordGet Method

25.1.52.3.49 RecordGetBatch Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordGetBatch Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordGetBatch(
    aCutoffSize: Integer;
    aOptions: TnxRecordGetBatchExOptions;
    aStream: TnxMemoryStream;
    aMaxRecords: Integer = -1;
    aForward: Boolean = True
): TnxResult; virtual; abstract;
```

Remarks

This function returns a batch of record data in a Stream. aCutoffSize - the maximum length to return in bytes. aOptions - the (blob/bookmark) options to use for filling the stream aStream - the stream that receives the data aMaxRecords - the maximum number of records to return.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordGetBatch Method

25.1.52.3.50 RecordGetForKey Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.RecordGetForKey Method

[TnxAbstractCursor Class](#)

Pascal

```
public function RecordGetForKey(
  aFieldCount: Integer;
  aPartialLen: Integer;
  aDirectKey: Boolean;
  aKeyData: PnxByteArray;
  aData: PnxByteArray;
  aMatchFirst: Boolean;
  aFirstCall: Boolean
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aFieldCount	number of fields to be compared. 0 = all fields
aPartialLen	number of characters yo compare for the last field. 0 = no partial compare
aDirectKey	True = passed in data is directly a key, False = passed in data is a record and the key is extracted.
aKeyData	The data according to the format defined by aDirectKey.
aData	used to return the record once it has been found. can be nil.
aMatchFirst	matches the first (True) or lsat (False) record that matches the key.
aFirstCall	must be true on the first call. If the function returns DBIERR_NX_FILTERTIMEOUT keep calling the function with the same parameters and aFirstCall set to false till another result is returned.

Remarks

RecordGetForKey positions the cursor on the first/last record that matches the passed in key and returns the record.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordGetForKey Method

25.1.52.3.51 RecordGetNext Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordGetNext Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordGetNext(
    aLockType: TnxLockType;
    aData: PnxByteArray
): TnxResult; virtual; abstract;
```

Remarks

RecordGet moves to the next valid (in regards of ranges and filters) record returns a pointer to the then current record buffer. It needs to be able to acquire the given lock type for this function to be successful.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordGetNext Method

25.1.52.3.52 RecordGetPrior Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordGetPrior Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordGetPrior(
    aLockType: TnxLockType;
    aData: PnxByteArray
): TnxResult; virtual; abstract;
```

Remarks

RecordGet moves to the prior valid (in regards of ranges and filters) record returns a pointer to the then current record buffer. It needs to be able to acquire the given lock type for this function to be successful.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordGetPrior Method

25.1.52.3.53 RecordInsert Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordInsert Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordInsert(
    aLockType: TnxLockType;
    aData: PnxByteArray
): TnxResult; virtual; abstract;
```

■ Remarks

RecordInsert creates a new record and returns a pointer to the then current record buffer. It needs to be able to acquire the given lock type for this function to be successful.

■ See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordInsert Method

25.1.52.3.54 RecordInsertBatch Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.RecordInsertBatch Method

[TnxAbstractCursor Class](#)

Pascal

```
public function RecordInsertBatch(
    aStream: TnxCustomMemoryStream
): TnxResult; virtual; abstract;
```

■ Remarks

This function adds the data in a aStream as new records. Please note that the size of aStream must be a multiple of the record size.

■ See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordInsertBatch Method

25.1.52.3.55 RecordIsLocked Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.RecordIsLocked Method

[TnxAbstractCursor Class](#)

Pascal

```
public function RecordIsLocked(
    aLockType: TnxLockType;
    out aLockPresent: TnxLockPresent
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aLockType	the type of lock you want to check.
aLockPresent	the state of the given lock type

■ Remarks

RecordIsLocked returns the current record lock state for a certain lock type.

■ See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordIsLocked Method

25.1.52.3.56 RecordLockRelease Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordLockRelease Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordLockRelease(
    aAllLocks: Boolean
): TnxResult; virtual; abstract;
```

Remarks

This function releases the last lock or alternatively all locks in the record.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordLockRelease Method

25.1.52.3.57 RecordModify Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RecordModify Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function RecordModify(
    aData: PnxByteArray;
    aReleaseLock: boolean
): TnxResult; virtual; abstract;
```

Remarks

RecordInsert sets the record buffer to the give data. The record must have a proper lock acquired for this function to succeed. If aReleaseLock is true all locks on the current record are released.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RecordModify Method

25.1.52.3.58 RequestNestedTransactions Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.RequestNestedTransactions Method**[TnxAbstractCursor Class](#)**Pascal**

```
public procedure RequestNestedTransactions; virtual;
```

Remarks

This function can be called from an extender to request that the cursor uses nested transactions for all operations. By default this is not supported.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > RequestNestedTransactions Method

25.1.52.3.59 SetToBegin Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractCursor.SetToBegin Method

TnxAbstractCursor Class

Pascal

```
public function SetToBegin: TnxResult; virtual; abstract;
```

Remarks

- navigation -

This function sets the cursor position to the begin of the dataset. This is not the first record but before it!

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > SetToBegin Method

25.1.52.3.60 SetToBookmark Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractCursor.SetToBookmark Method

TnxAbstractCursor Class

Pascal

```
public function SetToBookmark(
    aBookmark: PnxByteArray;
    aSize: TnxWord32
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aBookmark	the data of the bookmark
aSize	size of the bookmark data

Remarks

This function will position the cursor on the given bookmark

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > SetToBookmark Method

25.1.52.3.61 SetToCursor Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractCursor.SetToCursor Method

TnxAbstractCursor Class

Pascal

```
public function SetToCursor(
    aCursor: TnxAbstractCursor
): TnxResult; virtual; abstract;
```

Remarks

This function synchronizes the position with the given aCursor. Please note that the two cursor have to point to the same dataset.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > SetToCursor Method

25.1.52.3.62 SetToEnd Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.SetToEnd Method

[TnxAbstractCursor Class](#)

Pascal

```
public function SetToEnd: TnxResult; virtual; abstract;
```

Remarks

This function sets the cursor position to the end of the dataset. This is NOT the last record but after it!

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > SetToEnd Method

25.1.52.3.63 SetToFilter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.SetToFilter Method

[TnxAbstractCursor Class](#)

Pascal

```
public function SetToFilter(
    aFilter: TnxBaseFilterDescriptor;
    aKeyFilter: TnxBaseFilterDescriptor;
    aForward: TnxBoolean
): TnxResult; virtual; abstract;
```

Remarks

This function sets the cursor position before the first record that meets the filter criteria in aFilter.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > SetToFilter Method

25.1.52.3.64 SetToKey Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.SetToKey Method

TnxAbstractCursor Class

```
Pascal
public function SetToKey(
    aSearchAction: TnxSearchKeyAction;
    aFieldCount: Integer;
    aPartialLen: Integer;
    aDirectKey: Boolean;
    aKeyData: PnxByteArray
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aSearchAction	defines the relationship between the passed in key and the key the cursor is positioned on.
aFieldCount	number of fields to be compared. 0 = all fields
aPartialLen	number of characters yo compare for the last field. 0 = no partial compare
aDirectKey	True = passed in data is directly a key, False = passed in data is a record and the key is extracted.
aKeyData	The data according to the format defined by aDirectKey.

Remarks

SetToKey positions the cursor relative to a specific key, The cursor is always positioned on a crack. For skaEqualLast, skaSmallerEqual and skaSmaller it's the crack after the matching key, otherwise the crack before.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > SetToKey Method

25.1.52.3.65 SwitchToIndex Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractCursor.SwitchToIndex Method

TnxAbstractCursor Class

```
Pascal
public function SwitchToIndex(
    const aIndexName: string;
    aIndexID: Integer;
    aPosnOnRec: Boolean
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aIndexName	the name of the index
aIndexID	the id of the index in the dictionary
aPosnOnRec	if true it keeps the current record active

Remarks

SwitchToIndex activates the given index. It aPosOnRec it repositions the cursor to the record active before changing the index, otherwise it jumps to the first record.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > SwitchToIndex Method

25.1.52.3.66 TableIsLocked Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.TableIsLocked Method

[TnxAbstractCursor Class](#)

Pascal

```
public function TableIsLocked(
  aLockType: TnxLockType;
  out aLockPresent: TnxLockPresent
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aLockType	the type of lock you want to check.
aLockPresent	the state of the given lock type

Remarks

TableIsLocked returns the current table lock state for a certain lock type.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > TableIsLocked Method

25.1.52.3.67 TableLockAcquire Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.TableLockAcquire Method

[TnxAbstractCursor Class](#)

Pascal

```
public function TableLockAcquire(
  aLockType: TnxLockType
): TnxResult; virtual; abstract;
```

Remarks

This function will try to acquire a table lock with the given aLockType.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > TableLockAcquire Method

25.1.52.3.68 TableLockRelease Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.TableLockRelease Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function TableLockRelease(
    aLockType: TnxLockType;
    aAllLocks: Boolean
): TnxResult; virtual; abstract;
```

Remarks

This function will release and acquired lock with the given aLockType or alternatively (if aAllLocks is true) all locks on the table.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > TableLockRelease Method

25.1.52.3.69 TableStreamDelete Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.TableStreamDelete Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function TableStreamDelete(
    const aName: string
): TnxResult; virtual; abstract;
```

Remarks

This function deletes the table stream with the given aName.

See Also[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > TableStreamDelete Method

25.1.52.3.70 TableStreamGetList Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractCursor.TableStreamGetList Method**[TnxAbstractCursor Class](#)**Pascal**

```
public function TableStreamGetList(
    aStreams: TStrings
): TnxResult; virtual; abstract;
```

Remarks

This function returns the names of all streams stored with the table.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > TableStreamGetList Method

25.1.52.3.71 TableStreamRead Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.TableStreamRead Method

TnxAbstractCursor Class

Pascal

```
public function TableStreamRead(
  const aName: string;
  aStream: TStream
): TnxResult; virtual; abstract;
```

Remarks

This function reads the WHOLE stream with the given name into aStream.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > TableStreamRead Method

25.1.52.3.72 TableStreamWrite Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.TableStreamWrite Method

TnxAbstractCursor Class

Pascal

```
public function TableStreamWrite(
  const aName: string;
  aStream: TStream
): TnxResult; virtual; abstract;
```

Remarks

This function rewrites the WHOLE stream with the given name.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Methods > TableStreamWrite Method

25.1.52.4 Properties

25.1.52.4.1 _Dictionary Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor._Dictionary Property

TnxAbstractCursor Class

Pascal

```
public property _Dictionary: TnxDataDictionary;
```

Remarks

This property returns the data dictionary for this cursor.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Properties > _Dictionary Property

25.1.52.4.2 Database Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.Database Property

[TnxAbstractCursor Class](#)

Pascal

```
public property Database: TnxAbstractDatabase;
```

Remarks

This property returns the database the cursor is linked to.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Properties > Database Property

25.1.52.4.3 DatabaseNext Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.DatabaseNext Property

[TnxAbstractCursor Class](#)

Pascal

```
public property DatabaseNext: TnxAbstractCursor;
```

Remarks

This is DatabaseNext, a member of class TnxAbstractCursor.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Properties > DatabaseNext Property

25.1.52.4.4 DatabasePrev Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.DatabasePrev Property

[TnxAbstractCursor Class](#)

Pascal

```
public property DatabasePrev: TnxAbstractCursor;
```

Remarks

This is DatabasePrev, a member of class TnxAbstractCursor.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Properties > DatabasePrev Property

25.1.52.4.5 FullName Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.FullName Property

[TnxAbstractCursor Class](#)

Pascal

```
public property FullName: string;
```

Remarks

Returns the full table name in relation to master and sub tables.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Properties > FullName Property

25.1.52.4.6 TableDescriptor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractCursor.TableDescriptor Property

[TnxAbstractCursor Class](#)

Pascal

```
public property TableDescriptor: TnxBaseTableDescriptor;
```

Remarks

Returns the TableDescriptor instance in the data dictionary associated with the cursor.

See Also

[TnxAbstractCursor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractCursor Class > Properties > TableDescriptor Property

25.1.53 TnxAbstractDatabase Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAbstractDatabase = class(TnxAbstractTimeoutObject);
```

File

[nxsdServerEngine](#)

Remarks

The abstract declaration of a Server Database. All actual database implementations MUST be descendent from this class. This is the server side version of TnxDatabase.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class

25.1.53.1 Constructors

25.1.53.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.Create Constructor

TnxAbstractDatabase Class

Pascal

```
public constructor Create(
    aSession: TnxAbstractSession;
    aTransContext: TnxAbstractTransContext;
    aTimeout: TnxWord32
);
```

Parameters

Parameters	Description
aSession	the session object the database belongs to
aTransContext	the transaction context this database belongs to. If no transaction context is passed in a new one will be created.
aTimeout	the timeout for this database in msec

Remarks

constructor

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Constructors > Create Constructor

25.1.53.2 Destructors

25.1.53.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.Destroy Destructor

TnxAbstractDatabase Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Destructors > Destroy Destructor

25.1.53.3 Methods

25.1.53.3.1 AfterConstruction Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractDatabase.AfterConstruction Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public procedure AfterConstruction; override;
```

Remarks

see TObject.AfterConstruction

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > AfterConstruction Method

25.1.53.3.2 BeforeDestruction Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractDatabase.BeforeDestruction Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeConstruction

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > BeforeDestruction Method

25.1.53.3.3 CursorOpen Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractDatabase.CursorOpen Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function CursorOpen(
  out aCursor: TnxAbstractCursor;
  const aTableName: string;
  const aKey: string;
  aOpenMode: TnxOpenMode;
```

```
aShareMode: TnxShareMode;
aTimeout: TnxWord32;
aAllowSystem: Boolean = False
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aCursor	the returned instance
aTableName	the name of the table the cursor should be created for
aOpenMode	the mode you want to open the cursor for
aShareMode	defines how the Cursor should handle sharing
aTimeout	the timeout for the cursor in msec
aDictionary	the dictionary of the table the cursor is created for

Remarks

This function opens a new cursor for a table and returns an instance of a descendent of TnxAbstractCursor. Each table can have as many Cursors as wanted active at the same time

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > CursorOpen Method

25.1.53.3.4 GetChangedTables Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.GetChangedTables Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function GetChangedTables(
  aTables: TStrings
): TnxResult; virtual; abstract;
```

Remarks

Returns the names of tables that have been changed since this function was last called.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > GetChangedTables Method

25.1.53.3.5 GetFreespace Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.GetFreespace Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function GetFreespace(
  out aFreeSpace: TnxInt64
```

```
    ): TnxResult; virtual; abstract;
```

Remarks

This function returns the free space of the device the database is stored on. Result is in bytes.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > GetFreespace Method

25.1.53.6 GetPath Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.GetPath Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function GetPath(
    out aPath: string
): TnxResult; virtual; abstract;
```

Remarks

Returns the (file) path of the current database.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > GetPath Method

25.1.53.7 LookupByID Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.LookupByID Method

[TnxAbstractDatabase Class](#)

Pascal

```
public class function LookupByID(
    aSessionID: TnxSessionID;
    aDatabaseID: TnxDatabaseID;
    out aDatabase
): TnxResult; virtual;
```

Parameters

Parameters	Description
aDatabaseID	the ID to look for
aDatabase	a var that will hold the class if found

Remarks

This is the class function that returns the object with a certain ID.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > LookupByID Method

25.1.53.3.8 StatementAlloc Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.StatementAlloc Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function StatementAlloc(
    out aStatement: TnxAbstractStatement;
    aTimeout: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

This function creates a new instance of a SQL statement. It does NOT execute the query! Parameters:
 aStatement: the returned statement instance aTimeout: the timeout for the new statement in msec

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > StatementAlloc Method

25.1.53.3.9 StatementExecDirect Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.StatementExecDirect Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function StatementExecDirect(
    aTimeout: TnxWord32;
    aStatementType: TnxStatementType;
    const aQueryText: WideString;
    aPrepareStream: TStream;
    const aParams: TnxSqlParamList;
    aSetParamsStream: TStream;
    out aCursor: TnxAbstractCursor;
    aOpenMode: TnxOpenMode;
    aExecStream: TStream;
    aGetParamsStream: TStream;
    out aPhase: TnxStatementExecDirectPhase
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTimeout	Timeout in ms
aStatementType	indicates if this is a stored procedure or standard query
aPrepareStream	data returned by the prepare phase
aParams	the list of parameter definitions used for preparing and executing
aSetParamsStream	data returned by the setparams phase
aCursor	the new cursor returned
aOpenMode	the open mode for the cursor

aExecStream	data returned by the exec phase
aGetParamsStream	out-parameter stream
aPhase	returns the phase the query is in

Remarks

This function creates and executes a given SQL query or stored procedure and returns a cursor to the result set.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > StatementExecDirect Method

25.1.53.3.10 TableAddIndex Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.TableAddIndex Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function TableAddIndex(
    const aTableName: string;
    const aKey: string;
    aIndexDesc: TnxIndexDescriptor;
    aNewDefault: Boolean
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTableName	table name
aIndexDesc	a full index descriptor for the new index

Remarks

With TableAddIndex you can create a new index.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableAddIndex Method

25.1.53.3.11 TableAutoIncGet Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.TableAutoIncGet Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function TableAutoIncGet(
    const aTableName: string;
    const aKey: string;
    out aValue: TnxWord32
): TnxResult; virtual; abstract;
```

Remarks

This function returns the next AutoInc value for the specified table.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableAutoIncGet Method

25.1.53.3.12 TableBackup Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.TableBackup Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function TableBackup(
    const aTableName: string;
    const aKey: string;
    aTargetDatabase: TnxAbstractDataBase;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTargetDatabase	The database where the new table with identical name is created
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aName	table name

Remarks

This method starts an asynchronous backup of the table. For this purpose a new table with identical fields is created and the records are copied over from the old one. The original data dictionary is added as a tablestream.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableBackup Method

25.1.53.3.13 TableBuild Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.TableBuild Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function TableBuild(
    aOverWrite: Boolean;
    const aTableName: string;
    const aKey: string;
    aDictionary: TnxDataDictionary;
    aTableScope: TnxTableScope;
    aAllowSystem: Boolean = False
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aOverWrite	if true, an existing table is overwritten if it is not in use
aTableName	the name of the new table
aDictionary	the dictionary that defines the table and index structure

Remarks

This function creates a new table.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableBuild Method

25.1.53.3.14 TableChangePassword Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.TableChangePassword Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function TableChangePassword(
    const aTableName: string;
    const aOldKey: string;
    const aNewKey: string
): TnxResult; virtual; abstract;
```

Remarks

This function changes the password of the table with the given aTableName.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableChangePassword Method

25.1.53.3.15 TableDelete Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.TableDelete Method

[TnxAbstractDatabase Class](#)

Pascal

```
public function TableDelete(
    const aTableName: string;
    const aKey: string
): TnxResult; virtual; abstract;
```

Remarks

This function deletes the table with the given aTableName.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableDelete Method

25.1.53.3.16 TableDropIndex Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractDatabase.TableDropIndex Method

TnxAbstractDatabase Class

Pascal

```
public function TableDropIndex(
  const aTableName: string;
  const aKey: string;
  const aIndexName: string;
  aIndexID: Integer
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTableName	the name of the table that holds the index to be deleted.
aIndexName	the name of the index to be deleted
aIndexID	the id in the list of index descriptors of the index to be deleted.

Remarks

This function deletes an index.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableDropIndex Method

25.1.53.3.17 TableEmpty Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractDatabase.TableEmpty Method

TnxAbstractDatabase Class

Pascal

```
public function TableEmpty(
  const aTableName: string;
  const aKey: string
): TnxResult; virtual; abstract;
```

Remarks

This function deletes all records from the table with the given aTableName.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableEmpty Method

25.1.53.3.18 TableExists Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractDatabase.TableExists Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TableExists(
    const aTableName: string;
    const aKey: string;
    out aExists: Boolean;
    aAllowSystem: Boolean = False
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTableName	the table name to check
aExists	the var to be filled with the result.

Remarks**- table information -**

TableExists checks if a table exists.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableExists Method

25.1.53.3.19 TableGetDictionary Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxAbstractDatabase.TableGetDictionary Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TableGetDictionary(
    const aTableName: string;
    const aKey: string;
    aDictionary: TnxDataDictionary
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTableName	the name of the source table
aDictionary	the instance of TnxDataDictionary that should be filled with the information.

Remarks

With TableGetDictionary you can fill a TnxDataDictionary instance with the information about an existing table.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableGetDictionary Method

25.1.53.3.20 TableGetList Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractDatabase.TableGetList Method

TnxAbstractDatabase Class

Pascal

```
public function TableGetList(
    aList: TStrings
): TnxResult; virtual; abstract;
```

Remarks

TableGetList fills the given aList with all table names of the current database.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableGetList Method

25.1.53.3.21 TablePack Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractDatabase.TablePack Method

TnxAbstractDatabase Class

Pascal

```
public function TablePack(
    const aTableName: string;
    const aKey: string;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; virtual; abstract;
```

Parameters

Parameters

aTaskInfo

Description

Task information instance; use TaskInfo.GetStatus to find out the status of the operation.

aName

table name

Remarks

This method starts an asynchronous packing of the table. For this purpose a new identical table is created and the records are copied over from the old one.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TablePack Method

25.1.53.3.22 TableRebuildIndex Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.TableRebuildIndex Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TableRebuildIndex(
  const aTableName: string;
  const aKey: string;
  const aIndexName: string;
  aIndexID: Integer;
  out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aIndexName	the name of the index
aIndexID	the id of the index in the index descriptor list.
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aName	table name

Remarks

ReIndexTable starts an asynchronous re indexing of a certain index of a table.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableRebuildIndex Method

25.1.53.3.23 TableRecover Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.TableRecover Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TableRecover(
  const aTableName: string;
  const aKey: string;
  out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; virtual; abstract;
```

Remarks

This function walks all the blocks in the table on disk, and copies healthy records to aTableName_recovered, and copies bad records to aTableName_Failed.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableRecover Method

25.1.53.3.24 TableRename Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.TableRename Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TableRename(
    const aOldName: string;
    const aNewName: string;
    const aKey: string
): TnxResult; virtual; abstract;
```

Remarks

This function renames a table from aOldName to aNewName, if the table is not in use.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableRename Method

25.1.53.3.25 TableRestructure Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.TableRestructure Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TableRestructure(
    const aTableName: string;
    const aKey: string;
    aDictionary: TnxDataDictionary;
    aMapperDesc: TnxBaseTableMapperDescriptor;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTableName	table name
aDictionary	new dictionary of the table
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aFieldMap	defines how fields are mapped from the old to the new structure

Remarks

With this method you can start an asynchronous restructure of a given table.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TableRestructure Method

25.1.53.3.26 TransactionCommit Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractDatabase.TransactionCommit Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TransactionCommit: TnxResult;
```

Remarks

This method commits a pending transaction (level). If an error occurs (e.g. deadlock detected or transaction level has been marked as corrupted) the transaction is **not** automatically rolled back.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > [TransactionCommit Method](#)

25.1.53.3.27 TransactionCorrupted Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractDatabase.TransactionCorrupted Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TransactionCorrupted: TnxResult;
```

Remarks

This method marks the transaction (level) as corrupted and thus preventing it from ever being committed.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > [TransactionCorrupted Method](#)

25.1.53.3.28 TransactionGetLevel Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractDatabase.TransactionGetLevel Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TransactionGetLevel(
    out aTransLevel: Integer
): TnxResult;
```

Remarks

This method returns the nesting level of the current transaction.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > [TransactionGetLevel Method](#)

25.1.53.3.29 TransactionRollback Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.TransactionRollback Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TransactionRollback: TnxResult;
```

Remarks

Rollback rolls (as the name says) back a pending transaction (level). All changes since the last StartTransaction (or equivalent) call are lost.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > [TransactionRollback Method](#)

25.1.53.3.30 TransactionStart Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.TransactionStart Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TransactionStart(
    aFailSafe: Boolean;
    aSnapShot: Boolean
): TnxResult;
```

Parameters**Parameters****Description**

aFailSafe

if true it must be handled as fail safe.

aSnapShot

if true it is handled as a SnapshotTransaction

Remarks**- transaction management -**

StartTransaction tries to start a transaction. If it fails an exception is triggered.

See Also[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > [TransactionStart Method](#)

25.1.53.3.31 TransactionStartWith Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractDatabase.TransactionStartWith Method**[TnxAbstractDatabase Class](#)**Pascal**

```
public function TransactionStartWith(
    aFailSafe: Boolean;
    aSnapShot: Boolean;
```

```
const aCursors: array of TnxAbstractCursor
): TnxResult;
```

Parameters

Parameters	Description
aFailSafe	if true it must be handled as fail safe.
aSnapShot	if true it is handled as a SnapshotTransaction
aCursors	the cursors that should be part of the transaction

Remarks

Start a transaction if an exclusive lock can be granted on all specified cursors. If the grant fails one or more tables or an error occurs, an error code is returned.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Methods > TransactionStartWith Method

25.1.53.4 Properties

25.1.53.4.1 Session Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.Session Property

[TnxAbstractDatabase Class](#)

Pascal

```
public property Session: TnxAbstractSession;
```

Remarks

This property returns the session the database is linked to.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Properties > Session Property

25.1.53.4.2 TransContext Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractDatabase.TransContext Property

[TnxAbstractDatabase Class](#)

Pascal

```
public property TransContext: TnxAbstractTransContext;
```

Remarks

This property returns the transaction context the database is linked to.

See Also

[TnxAbstractDatabase Class](#)

You are here: Symbol Reference > Classes > TnxAbstractDatabase Class > Properties > TransContext Property

25.1.54 TnxAbstractSecurityMonitor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSecurityMonitor Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxAbstractSecurityMonitor = class(TnxBaseEngineMonitor);
```

File

[nxsdServerEngine](#)

Remarks

This is base for all a security monitors used in NexusDB.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxAbstractSecurityMonitor Class](#)

25.1.54.1 Methods

25.1.54.1.1 HasAdmins Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSecurityMonitor.HasAdmins Method

[TnxAbstractSecurityMonitor Class](#)

Pascal

```
public function HasAdmins: Boolean; virtual; abstract;
```

Remarks

HasAdmins returns true if there are any users defined.

See Also

[TnxAbstractSecurityMonitor Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxAbstractSecurityMonitor Class](#) > [Methods](#) > [HasAdmins Method](#)

25.1.54.1.2 HasUsers Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSecurityMonitor.HasUsers Method

[TnxAbstractSecurityMonitor Class](#)

Pascal

```
public function HasUsers: Boolean; virtual; abstract;
```

Remarks

HasUsers returns true if there are any users defined.

See Also

[TnxAbstractSecurityMonitor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSecurityMonitor Class > Methods > HasUsers Method

25.1.54.1.3 IsAdmin Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSecurityMonitor.IsAdmin Method

TnxAbstractSecurityMonitor Class

Pascal

```
public function IsAdmin(
  const aUserName: String;
  const aPassword: String
): boolean; virtual; abstract;
```

Remarks

This function returns true if there is an Admin user with the given aUserName and aPassword.

See Also

[TnxAbstractSecurityMonitor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSecurityMonitor Class > Methods > IsAdmin Method

25.1.54.1.4 UiStateVisible Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSecurityMonitor.UiStateVisible Method

TnxAbstractSecurityMonitor Class

Pascal

```
public function UiStateVisible: Boolean; override;
```

Remarks

returns always true, cause the User Interface for a Security Monitor should always be visible.

See Also

[TnxAbstractSecurityMonitor Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSecurityMonitor Class > Methods > UiStateVisible Method

25.1.55 TnxAbstractServerObject Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractServerObject Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAbstractServerObject = class(TnxPersistent);
```

File

[nxsdServerEngine](#)

Remarks

The base class for all server dependent objects.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractServerObject Class

25.1.55.1 Methods

25.1.55.1.1 BeforeDestruction Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractServerObject.BeforeDestruction Method

[TnxAbstractServerObject Class](#)

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeDestruction.

See Also

[TnxAbstractServerObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractServerObject Class > Methods > BeforeDestruction Method

25.1.55.1.2 Free Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractServerObject.Free Method

[TnxAbstractServerObject Class](#)

Pascal

```
public procedure Free;
```

Remarks

Reintroduction of the Free method

See Also

[TnxAbstractServerObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractServerObject Class > Methods > Free Method

25.1.55.2 Properties

25.1.55.2.1 RemoteThreadPriority Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractServerObject.RemoteThreadPriority Property

[TnxAbstractServerObject Class](#)

Pascal

```
public property RemoteThreadPriority: TnxThreadPriority;
```

Remarks

RemoteThreadPriority is the thread priority send to the server in case the object is living on a remote machine.

See Also

[TnxAbstractServerObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractServerObject Class > Properties > RemoteThreadPriority Property

25.1.56 TnxAbstractSession Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAbstractSession = class(TnxAbstractTimeoutObject);
```

File

[nxsdServerEngine](#)

Remarks

The abstract declaration of a Server Session. All actual session implementations MUST be descendent from this class. This is the server side version of TnxSession.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class

25.1.56.1 Constructors

25.1.56.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.Create Constructor

[TnxAbstractSession Class](#)

Pascal

```
public constructor Create(
  aServerEngine: TnxBaseServerEngine;
  aTimeout: TnxWord32;
  const aUserName: string;
  const aPassword: string;
  const aConnectedFrom: string
);
```

Parameters

Parameters	Description
aServerEngine	the server engine that owns the session
aTimeout	the timeout in msec for the session
aUserName	the user name used to logon to the server

aPassword

the password used to logon to the server

Remarks

Constructor

See Also

TnxAbstractSession Class

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Constructors > Create Constructor

25.1.56.2 Destructors

25.1.56.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.Destroy Destructor

TnxAbstractSession Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

TnxAbstractSession Class

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Destructors > Destroy Destructor

25.1.56.3 Methods

25.1.56.3.1 AfterConstruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.AfterConstruction Method

TnxAbstractSession Class

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

see TObject.AfterConstruction

See Also

TnxAbstractSession Class

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > AfterConstruction Method

25.1.56.3.2 AliasAdd Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.AliasAdd Method

TnxAbstractSession Class

Pascal

```
public function AliasAdd(
    const aAlias: string;
    const aPath: string
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aAlias	the name of the Alias.
aPath	the directory the Alias points to. The format is specific to the actual implementation

Remarks

This function implements the addition of a new Alias to the session.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > [TnxAbstractSession Class](#) > Methods > AliasAdd Method

25.1.56.3.3 AliasDelete Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.AliasDelete Method

[TnxAbstractSession Class](#)

Pascal

```
public function AliasDelete(
    const aAlias: string
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aAlias	the name of the Alias to be removed.

Remarks

This function implements the removal of an Alias from the session.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > [TnxAbstractSession Class](#) > Methods > AliasDelete Method

25.1.56.3.4 AliasGetList Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.AliasGetList Method

[TnxAbstractSession Class](#)

Pascal

```
public function AliasGetList(
    aList: TObjectList
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aList	The list that will be filled with descriptors.

Remarks

This function needs to return the List of Aliases as a list of TnxAliasDescriptor objects.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > AliasGetList Method

25.1.56.3.5 AliasGetPath Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.AliasGetPath Method

[TnxAbstractSession Class](#)

Pascal

```
public function AliasGetPath(
  const aAlias: string;
  out aPath: string
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aAlias	the name of the alias
aPath	the variable that will be set to the path.

Remarks

This function returns the Path of a directory. The format of aPath is specific to the actual implementation, in the default implementation it is a directory.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > AliasGetPath Method

25.1.56.3.6 AliasModify Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.AliasModify Method

[TnxAbstractSession Class](#)

Pascal

```
public function AliasModify(
  const aAlias: string;
  const aNewName: string;
  const aNewPath: string
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aAlias	the name of the Alias.

aNewName	a new name for the alias.
aPath	the directory the Alias points to. The format is specific to the actual implementation

Remarks

This function implements the modification of a an Alias. The function supports modifying name and/or path of the Alias.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > AliasModify Method

25.1.56.3.7 BeforeDestruction Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.BeforeDestruction Method

[TnxAbstractSession Class](#)

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeConstruction

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > BeforeDestruction Method

25.1.56.3.8 CloseInactiveFolders Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.CloseInactiveFolders Method

[TnxAbstractSession Class](#)

Pascal

```
public function CloseInactiveFolders: TnxResult; virtual; abstract;
```

Remarks

This function uncaches and closes all unused folders/directories.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > CloseInactiveFolders Method

25.1.56.3.9 CloseInactiveTables Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.CloseInactiveTables Method

[TnxAbstractSession Class](#)

Pascal

```
public function CloseInactiveTables: TnxResult; virtual; abstract;
```

Remarks

This function uncaches and closes all unused tables.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > CloseInactiveTables Method

25.1.56.3.10 DatabaseOpen Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.DatabaseOpen Method

[TnxAbstractSession Class](#)

Pascal

```
public function DatabaseOpen(
    out aDatabase: TnxAbstractDatabase;
    const aAliasType: string;
    const aAlias: string;
    aTransContext: TnxAbstractTransContext;
    aOpenMode: TnxOpenMode;
    aShareMode: TnxShareMode;
    aTimeout: TnxWord32
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aDatabase	the variable to receive the resulting instance.
aAliasType	the type of the alias. Currently nxatAliasName and nxatAliasPath
aAlias	the name of the alias to open
aTransContext	the transaction context this database belongs to. If this is nil the database creates it's own transaction context.
aOpenMode	the mode the database should be opened in
aShareMode	Share mode for opening the database
aTimeout	the timeout in msec for the database

Remarks

This function will open a alias/database and return the TnxAbstractDatabase object created it for it.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > DatabaseOpen Method

25.1.56.3.11 Failed Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.Failed Method

TnxAbstractSession Class

Pascal

```
public procedure Failed(
  aPermanent: Boolean
);
```

Remarks

Removes any failed locks and marks the session as failed.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > Failed Method

25.1.56.3.12 GetRegisteredClassList Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.GetRegisteredClassList Method

TnxAbstractSession Class

Pascal

```
public function GetRegisteredClassList(
  const aclassListType: TnxClassListType;
  aList: TStrings
): TnxResult; virtual; abstract;
```

Remarks

This function returns a list of registered engines of the requested type.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > GetRegisteredClassList Method

25.1.56.3.13 GetUserInfo Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.GetUserInfo Method

TnxAbstractSession Class

Pascal

```
public procedure GetUserInfo(
  aStrings: TStrings
); virtual;
```

Remarks

misc

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > GetUserInfo Method

25.1.56.3.14 LookupByID Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.LookupByID Method

TnxAbstractSession Class

Pascal

```
public class function LookupByID(
  aSessionID: TnxSessionID;
  out aSession
): TnxResult; virtual;
```

Parameters

Parameters	Description
aSessionID	the ID to look for
aSession	a var that will hold the class if found

Remarks

This is the class function that returns the object with a certain ID.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > LookupByID Method

25.1.56.3.15 PasswordAdd Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.PasswordAdd Method

TnxAbstractSession Class

Pascal

```
public function PasswordAdd(
  const aPassword: string
): TnxResult; virtual; abstract;
```

Remarks

Adds a password to the Session internal password list.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > PasswordAdd Method

25.1.56.3.16 PasswordRemove Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.PasswordRemove Method

TnxAbstractSession Class

Pascal

```
public function PasswordRemove(
    const aPassword: string
): TnxResult; virtual; abstract;
```

Remarks

Removes a password from the Session internal password list.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > PasswordRemove Method

25.1.56.3.17 PasswordRemoveAll Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.PasswordRemoveAll Method

[TnxAbstractSession Class](#)

Pascal

```
public function PasswordRemoveAll: TnxResult; virtual; abstract;
```

Remarks

Clears the password list.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > PasswordRemoveAll Method

25.1.56.3.18 TransContextCreate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractSession.TransContextCreate Method

[TnxAbstractSession Class](#)

Pascal

```
public function TransContextCreate(
    out aTransContext: TnxAbstractTransContext;
    aTimeout: TnxWord32
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aTransContext	the variable to receive the resulting instance.
aTimeout	the timeout in msec for the transaction context

Remarks

This function will create a transaction context and return the TnxAbstractTransContext object created it for it. The transaction context can then be passed into DatabaseOpen.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Methods > TransContextCreate Method

25.1.56.4 Properties

25.1.56.4.1 Authenticated Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.Authenticated Property

TnxAbstractSession Class

Pascal

```
public property Authenticated: Boolean;
```

Remarks

This property returns true, if the session was logged on successfully with User Name and Password.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > Authenticated Property

25.1.56.4.2 CleanedUp Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.CleanedUp Property

TnxAbstractSession Class

Pascal

```
public property CleanedUp: Boolean;
```

Remarks

This property returns true, if the session has failed and has been cleaned up. It no longer owns any databases.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > CleanedUp Property

25.1.56.4.3 ClientVersion Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.ClientVersion Property

TnxAbstractSession Class

Pascal

```
public property ClientVersion: Integer;
```

Remarks

This property returns the version of the client that opened the session.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > ClientVersion Property

25.1.56.4.4 ConnectedFrom Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.ConnectedFrom Property

TnxAbstractSession Class

Pascal

```
public property ConnectedFrom: string;
```

Remarks

This property returns the connection information for the session.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > ConnectedFrom Property

25.1.56.4.5 Password Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.Password Property

TnxAbstractSession Class

Pascal

```
public property Password: string;
```

Remarks

This property returns the password used to logon the session.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > Password Property

25.1.56.4.6 ServerEngine Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSession.ServerEngine Property

TnxAbstractSession Class

Pascal

```
public property ServerEngine: TnxBaseServerEngine;
```

Remarks

This property returns the server engine the session is attached to.

See Also

[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > ServerEngine Property

25.1.56.4.7 ServerVersion Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractSession.ServerVersion Property**[TnxAbstractSession Class](#)**Pascal**

```
public property ServerVersion: Integer;
```

Remarks

This property returns the version of the server that runs the session.

See Also[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > ServerVersion Property

25.1.56.4.8 UserName Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractSession.UserName Property**[TnxAbstractSession Class](#)**Pascal**

```
public property UserName: string;
```

Remarks

This property returns the user name used to logon the session.

See Also[TnxAbstractSession Class](#)

You are here: Symbol Reference > Classes > TnxAbstractSession Class > Properties > UserName Property

25.1.57 TnxAbstractStatement Class*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxAbstractStatement Class**[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxAbstractStatement = class(TnxAbstractTimeoutObject);
```

File[nxsdServerEngine](#)**Remarks**

The abstract declaration of a server statement. All actual statement implementations MUST be descendent from this class. This is the server side version of TnxQuery.

See Also
[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class

25.1.57.1 Constructors

25.1.57.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractStatement.Create Constructor

TnxAbstractStatement Class

Pascal

```
public constructor Create(
    aDatabase: TnxAbstractDatabase;
    aTimeout: TnxWord32
);
```

Remarks

constructor Parameters: aDatabase - the database instance the statement is linked to aTimeout - the timeout of the statement in msec

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Constructors > Create Constructor

25.1.57.2 Destructors

25.1.57.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractStatement.Destroy Destructor

TnxAbstractStatement Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Destructors > Destroy Destructor

25.1.57.3 Methods

25.1.57.3.1 AfterConstruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractStatement.AfterConstruction Method

TnxAbstractStatement Class

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

see TObject.AfterConstruction

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Methods > AfterConstruction Method

25.1.57.3.2 BeforeDestruction Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractStatement.BeforeDestruction Method

TnxAbstractStatement Class

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeConstruction

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Methods > BeforeDestruction Method

25.1.57.3.3 Exec Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractStatement.Exec Method

TnxAbstractStatement Class

Pascal

```
public function Exec(
    out aCursor: TnxAbstractCursor;
    aOpenMode: TnxOpenMode;
    aStream: TStream
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aCursor	the resulting cursor
aOpenMode	the mode you want the cursor to be opened
aStream	a stream instance that will be filled with result/reply

Remarks

This method execute the query and returns a cursor for it.

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Methods > Exec Method

25.1.57.3.4 GetParams Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractStatement.GetParams Method**[TnxAbstractStatement Class](#)**Pascal**

```
public function GetParams(
    const aParams: TnxSqlParamList;
    aStream: TStream
): TnxResult; virtual; abstract;
```

Remarks

Returns the given Parameters as stream.

See Also[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Methods > GetParams Method

25.1.57.3.5 LookupByID Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractStatement.LookupByID Method**[TnxAbstractStatement Class](#)**Pascal**

```
public class function LookupByID(
    aSessionID: TnxSessionID;
    aStatementID: TnxStatementID;
    out aStatement
): TnxResult; virtual;
```

Parameters

Parameters	Description
aStatementID	the ID to look for
aStatement	a var that will hold the class if found

Remarks

This is the class function that returns the object with a certain ID.

See Also[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Methods > LookupByID Method

25.1.57.3.6 Prepare Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractStatement.Prepare Method**[TnxAbstractStatement Class](#)**Pascal**

```
public function Prepare(
```

```
aStatementType: TnxStatementType;
const aQueryText: WideString;
aStream: TStream
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aQueryText	the SQL query to be executed
aStream	a stream instance that will be filled with result/reply

Remarks

This method prepares the sql query. It does NOT execute the query.

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Methods > Prepare Method

25.1.57.3.7 SetParams Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractStatement.SetParams Method

[TnxAbstractStatement Class](#)

Pascal

```
public function SetParams(
  const aParams: TnxSqlParamList;
  aStream: TStream
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aParams	the list of parameters
aStream	a stream instance that will be filled with result/reply.

Remarks

With SetParams the parameters of the query will be set.

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Methods > SetParams Method

25.1.57.4 Properties**25.1.57.4.1 Database Property**

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractStatement.Database Property

[TnxAbstractStatement Class](#)

Pascal

```
public property Database: TnxAbstractDatabase;
```

Remarks

This property returns the database the statement is linked to.

See Also

[TnxAbstractStatement Class](#)

You are here: Symbol Reference > Classes > TnxAbstractStatement Class > Properties > Database Property

25.1.58 TnxAbstractTaskInfo Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTaskInfo Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAbstractTaskInfo = class(TnxExtendableServerObject);
```

File

[nxsdServerEngine](#)

Remarks

An instance of a descendant of this class is created for every task executed and on the server. It will be freed once the task is finished.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class

25.1.58.1 Constructors

25.1.58.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTaskInfo.Create Constructor

[TnxAbstractTaskInfo Class](#)

Pascal

```
public constructor Create(  
  aSession: TnxAbstractSession  
) ;
```

Remarks

public comes before protected to fix problem with BCB constructor.

See Also

[TnxAbstractTaskInfo Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Constructors > Create Constructor

25.1.58.2 Destructors

25.1.58.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTaskInfo.Destroy Destructor

TnxAbstractTaskInfo Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxAbstractTaskInfo Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Destructors > Destroy Destructor

25.1.58.3 Methods

25.1.58.3.1 AfterConstruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTaskInfo.AfterConstruction Method

TnxAbstractTaskInfo Class

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

see TObject.AfterConstruction.

See Also

[TnxAbstractTaskInfo Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Methods > AfterConstruction Method

25.1.58.3.2 BeforeDestruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTaskInfo.BeforeDestruction Method

TnxAbstractTaskInfo Class

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeConstruction

See Also

[TnxAbstractTaskInfo Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Methods > BeforeDestruction Method

25.1.58.3.3 Cancel Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractTaskInfo.Cancel Method**[TnxAbstractTaskInfo Class](#)**Pascal**

```
public function Cancel: TnxResult; virtual; abstract;
```

Remarks

Call this method to cancel a Task. This method only triggers the cancel, the tasks may still be active until it reaches a safe state.

See Also[TnxAbstractTaskInfo Class](#)*You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Methods > Cancel Method*

25.1.58.3.4 GetStatus Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractTaskInfo.GetStatus Method**[TnxAbstractTaskInfo Class](#)**Pascal**

```
public function GetStatus(
    out aCompleted: Boolean;
    out aStatus: TnxTaskStatus
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aCompleted	is set to true if the task is completed
aStatus	is set to the task status

Remarks

This function returns the current status a server task.

See Also[TnxAbstractTaskInfo Class](#)*You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Methods > GetStatus Method*

25.1.58.3.5 LookupByID Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractTaskInfo.LookupByID Method**[TnxAbstractTaskInfo Class](#)**Pascal**

```
public class function LookupByID(
    aSessionID: TnxSessionID;
    aTaskID: TnxTaskID;
    out aTaskInfo
): TnxResult; virtual; overload;
```

Parameters

Parameters	Description
aTaskID	the ID to look for
aTaskInfo	a var that will hold the class if found

Remarks

This is the class function that returns the object with a certain ID.

See Also

[TnxAbstractTaskInfo Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Methods > LookupByID Method

25.1.58.4 Properties

25.1.58.4.1 Session Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTaskInfo.Session Property

[TnxAbstractTaskInfo Class](#)

Pascal

```
public property Session: TnxAbstractSession;
```

Remarks

This property returns the Server Session the Task was started with.

See Also

[TnxAbstractTaskInfo Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTaskInfo Class > Properties > Session Property

25.1.59 TnxAbstractTimeoutObject Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTimeoutObject Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAbstractTimeoutObject = class(TnxExtendableServerObject);
```

File

[nxsdServerEngine](#)

Remarks

The base class for all server classes that support timeout.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class

25.1.59.1 Constructors

25.1.59.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTimeoutObject.Create Constructor

TnxAbstractTimeoutObject Class

```
Pascal
public constructor Create(
    aParent: TnxExtendableServerObject;
    aTimeout: TnxWord32
);
```

Remarks

constructor.

See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Constructors > Create Constructor

25.1.59.2 Methods

25.1.59.2.1 OptionClear Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTimeoutObject.OptionClear Method

TnxAbstractTimeoutObject Class

```
Pascal
public function OptionClear(
    const aName: string
): TnxResult; override;
```

Remarks

This is OptionClear, a member of class TnxAbstractTimeoutObject.

See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Methods > OptionClear Method

25.1.59.2.2 OptionGet Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTimeoutObject.OptionGet Method

TnxAbstractTimeoutObject Class

```
Pascal
public function OptionGet(
    const aName: string;
    out aValue: string
```

```
) : TnxResult; override;
```

■ Remarks

This is OptionGet, a member of class TnxAbstractTimeoutObject.

■ See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Methods > OptionGet Method

25.1.59.2.3 OptionGetEffective Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTimeoutObject.OptionGetEffective Method

[TnxAbstractTimeoutObject Class](#)

Pascal

```
public function OptionGetEffective(
  const aName: string;
  out aValue: string
) : TnxResult; override;
```

■ Remarks

This is OptionGetEffective, a member of class TnxAbstractTimeoutObject.

■ See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Methods > OptionGetEffective Method

25.1.59.2.4 OptionList Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTimeoutObject.OptionList Method

[TnxAbstractTimeoutObject Class](#)

Pascal

```
public function OptionList(
  aList: TStrings
) : TnxResult; override;
```

■ Remarks

This is OptionList, a member of class TnxAbstractTimeoutObject.

■ See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Methods > OptionList Method

25.1.59.2.5 OptionListEffective Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTimeoutObject.OptionListEffective Method

[TnxAbstractTimeoutObject Class](#)

Pascal

```
public function OptionListEffective(
    aList: TStrings
): TnxResult; override;
```

Remarks

This is OptionListEffective, a member of class TnxAbstractTimeoutObject.

See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Methods > OptionListEffective Method

25.1.59.2.6 OptionSet Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTimeoutObject.OptionSet Method

[TnxAbstractTimeoutObject Class](#)

Pascal

```
public function OptionSet(
    const aName: string;
    const aValue: string
): TnxResult; override;
```

Remarks

This is OptionSet, a member of class TnxAbstractTimeoutObject.

See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Methods > OptionSet Method

25.1.59.3 Properties

25.1.59.3.1 Timeout Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTimeoutObject.Timeout Property

[TnxAbstractTimeoutObject Class](#)

Pascal

```
public property Timeout: TnxWord32;
```

Remarks

Property to read/write the timeout in msec for the object.

See Also

[TnxAbstractTimeoutObject Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTimeoutObject Class > Properties > Timeout Property

25.1.60 TnxAbstractTransContext Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTransContext Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAbstractTransContext = class(TnxAbstractTimeoutObject);
```

File

[nxsdServerEngine](#)

Remarks

The abstract declaration of a Server transaction context. All actual transaction context implementations MUST be descendent from this class. This is the server side version of TnxTransContext.

See Also

[nxsdServerEngine](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class

25.1.60.1 Constructors

25.1.60.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAbstractTransContext.Create Constructor

[TnxAbstractTransContext Class](#)

Pascal

```
public constructor Create(
    aSession: TnxAbstractSession;
    aTimeout: TnxWord32
);
```

Parameters

Parameters	Description
aSession	the session object the transaction context belongs to
aTimeout	the timeout for this TransContext in msec

Remarks

constructor

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Constructors > Create Constructor

25.1.60.2 Destructors

25.1.60.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTransContext.Destroy Destructor

TnxAbstractTransContext Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Destructors > Destroy Destructor

25.1.60.3 Methods

25.1.60.3.1 AfterConstruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTransContext.AfterConstruction Method

TnxAbstractTransContext Class

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

see TObject.AfterConstruction

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > AfterConstruction Method

25.1.60.3.2 BeforeDestruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractTransContext.BeforeDestruction Method

TnxAbstractTransContext Class

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeConstruction

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > BeforeDestruction Method

25.1.60.3.3 Failed Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractTransContext.Failed Method**[TnxAbstractTransContext Class](#)**Pascal**

```
public procedure Failed; virtual;
```

See Also[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > Failed Method

25.1.60.3.4 LookupByID Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractTransContext.LookupByID Method**[TnxAbstractTransContext Class](#)**Pascal**

```
public class function LookupByID(
    aSessionID: TnxSessionID;
    aTransContextID: TnxTransContextID;
    out aTransContext
) : TnxResult;
```

Parameters

Parameters	Description
aTransContextID	the ID to look for
aTransContext	a var that will hold the class if found

Remarks

This is the class function that returns the object with a certain ID.

See Also[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > LookupByID Method

25.1.60.3.5 TransactionCommit Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAbstractTransContext.TransactionCommit Method**[TnxAbstractTransContext Class](#)**Pascal**

```
public function TransactionCommit: TnxResult; virtual; abstract;
```

Remarks

This method commits a pending transaction (level). If an error occurs (e.g. deadlock detected or transaction level has been marked as corrupted) the transaction is **not** automatically rolled back.

See Also[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > TransactionCommit Method

25.1.60.3.6 TransactionCorrupted Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractTransContext.TransactionCorrupted Method

TnxAbstractTransContext Class

Pascal

```
public function TransactionCorrupted: TnxResult; virtual; abstract;
```

Remarks

This method marks the transaction (level) as corrupted and thus preventing it from ever being committed.

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > TransactionCorrupted Method

25.1.60.3.7 TransactionGetLevel Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractTransContext.TransactionGetLevel Method

TnxAbstractTransContext Class

Pascal

```
public function TransactionGetLevel(
    out aTransLevel: Integer
): TnxResult; virtual; abstract;
```

Remarks

This method returns the nesting level of the current transaction.

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > TransactionGetLevel Method

25.1.60.3.8 TransactionRollback Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractTransContext.TransactionRollback Method

TnxAbstractTransContext Class

Pascal

```
public function TransactionRollback: TnxResult; virtual; abstract;
```

Remarks

Rollback rolls (as the name says) back a pending transaction (level). All changes since the last StartTransaction (or equivalent) call are lost.

See Also

TnxAbstractTransContext Class

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > TransactionRollback Method

25.1.60.3.9 TransactionStart Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractTransContext.TransactionStart Method

TnxAbstractTransContext Class

Pascal

```
public function TransactionStart(
    aFailSafe: Boolean;
    aSnapShot: Boolean
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aFailSafe	if true it must be handled as fail safe.
aSnapShot	if true it is handled as a SnapshotTransaction

Remarks

- transaction management -

StartTransaction tries to start a transaction. If it fails an exception is triggered.

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > TransactionStart Method

25.1.60.3.10 TransactionStartWith Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxAbstractTransContext.TransactionStartWith Method

TnxAbstractTransContext Class

Pascal

```
public function TransactionStartWith(
    aFailSafe: Boolean;
    aSnapShot: Boolean;
    const aCursors: array of TnxAbstractCursor
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aFailSafe	if true it must be handled as fail safe.
aSnapShot	if true it is handled as a SnapshotTransaction
aCursors	the cursors that should be part of the transaction

Remarks

Start a transaction if an exclusive lock can be granted on all specified cursors. If the grant fails one or more tables or an error occurs, an error code is returned.

See Also[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Methods > TransactionStartWith Method

25.1.60.4 Properties

25.1.60.4.1 Session Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractTransContext.Session Property**[TnxAbstractTransContext Class](#)**Pascal**

```
public property Session: TnxAbstractSession;
```

Remarks

This property returns the session the transaction context is linked to.

See Also[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Properties > Session Property

25.1.60.4.2 SessionNext Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractTransContext.SessionNext Property**[TnxAbstractTransContext Class](#)**Pascal**

```
public property SessionNext: TnxAbstractTransContext;
```

Remarks

This is SessionNext, a member of class TnxAbstractTransContext.

See Also[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Properties > SessionNext Property

25.1.60.4.3 SessionPrev Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxAbstractTransContext.SessionPrev Property**[TnxAbstractTransContext Class](#)**Pascal**

```
public property SessionPrev: TnxAbstractTransContext;
```

Remarks

This is SessionPrev, a member of class TnxAbstractTransContext.

See Also

[TnxAbstractTransContext Class](#)

You are here: Symbol Reference > Classes > TnxAbstractTransContext Class > Properties > SessionPrev Property

25.1.61 TnxAdvAutoIncDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAdvAutoIncDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxAdvAutoIncDescriptor = class(TnxBaseAutoIncDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This is class TnxAdvAutoIncDescriptor.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxAdvAutoIncDescriptor Class

25.1.61.1 Methods

25.1.61.1.1 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAdvAutoIncDescriptor.IsEqual Method

[TnxAdvAutoIncDescriptor Class](#)

Pascal

```
public function IsEqual(
  aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxAdvAutoIncDescriptor.

See Also

[TnxAdvAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxAdvAutoIncDescriptor Class > Methods > IsEqual Method

25.1.61.1.2 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAdvAutoIncDescriptor.LoadFromReader Method

[TnxAdvAutoIncDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxAdvAutoIncDescriptor.

See Also

[TnxAdvAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxAdvAutoIncDescriptor Class > Methods > [LoadFromReader Method](#)

25.1.61.1.3 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAdvAutoIncDescriptor.SaveToWriter Method

[TnxAdvAutoIncDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxAdvAutoIncDescriptor.

See Also

[TnxAdvAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxAdvAutoIncDescriptor Class > Methods > [SaveToWriter Method](#)

25.1.61.2 Properties

25.1.61.2.1 InitialValue Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAdvAutoIncDescriptor.InitialValue Property

[TnxAdvAutoIncDescriptor Class](#)

Pascal

```
public property InitialValue: TnxAutoIncStepRange;
```

Remarks

This is InitialValue, a member of class TnxAdvAutoIncDescriptor.

See Also

[TnxAdvAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxAdvAutoIncDescriptor Class > Properties > [InitialValue Property](#)

25.1.61.2.2 Step Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxAdvAutoIncDescriptor.Step Property**[TnxAdvAutoIncDescriptor Class](#)**Pascal**

```
public property Step: TnxAutoIncStepRange;
```

Remarks

This is Step, a member of class TnxAdvAutoIncDescriptor.

See Also

[TnxAdvAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxAdvAutoIncDescriptor Class > Properties > Step Property

25.1.62 TnxAliasDescriptor Class*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxAliasDescriptor Class**[nxsdServerEngine](#)**Pascal**

```
public TnxAliasDescriptor = class(TnxObject);
```

File

[nxsdServerEngine](#)

Remarks

A Database Alias descriptor.

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Classes > TnxAliasDescriptor Class

25.1.63 TnxAutoGuidDefaultValueDescriptor Class*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxAutoGuidDefaultValueDescriptor Class**[nxsdDataDictionary](#)**Pascal**

```
public TnxAutoGuidDefaultValueDescriptor = class(TnxBaseDefaultValueDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This descriptor should be used if you want to set a Guid field to an automatically created new Guid.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > TnxAutoGuidDefaultValueDescriptor Class

25.1.64 TnxBaseAutoIncDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseAutoIncDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

This is the base class for AutoInc Descriptors. A AutoInc descriptor specifies the sub-engine that generates the AutoIncs.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class

25.1.64.1 Destructors

25.1.64.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.Destroy Destructor

[TnxBaseAutoIncDescriptor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxBaseAutoIncDescriptor.

See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Destructors > [Destroy Destructor](#)

25.1.64.2 Methods

25.1.64.2.1 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.CheckValid Method

[TnxBaseAutoIncDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseAutoIncDescriptor.

See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Methods > CheckValid Method

25.1.64.2.2 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.IsEqual Method

[TnxBaseAutoIncDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseAutoIncDescriptor.

See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Methods > IsEqual Method

25.1.64.2.3 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.LoadFromReader Method

[TnxBaseAutoIncDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseAutoIncDescriptor.

See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Methods > LoadFromReader Method

25.1.64.2.4 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.SaveToWriter Method

[TnxBaseAutoIncDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseAutoIncDescriptor.

See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Methods > SaveToWriter Method

25.1.64.3 Properties

25.1.64.3.1 AutoIncEngine Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.AutoIncEngine Property

[TnxBaseAutoIncDescriptor Class](#)

Pascal

```
public property AutoIncEngine: string;
```

Remarks

The name of the AutoInc Engine.

See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Properties > AutoIncEngine Property

25.1.64.3.2 FileNumber Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.FileNumber Property

[TnxBaseAutoIncDescriptor Class](#)

Pascal

```
public property FileNumber: Integer;
```

Remarks

Returns the index of the FileDescriptor of the associated file.

See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Properties > FileNumber Property

25.1.64.3.3 HeaderSection Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptor.HeaderSection Property

[TnxBaseAutoIncDescriptor Class](#)

Pascal
`public property HeaderSection: string;`

■ Remarks

Returns the header of the descriptor as string.

■ See Also

[TnxBaseAutoIncDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseAutoIncDescriptor Class > Properties > HeaderSection Property

25.1.65 TnxBaseBlobDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseBlobDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal
`public TnxBaseBlobDescriptor = class(TnxDictionaryItem);`

■ File

[nxsdDataDictionary](#)

■ Remarks

This is the base class for BLOB Descriptors. A Blob descriptor specifies the sub-engine that accesses the blobs.

■ See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class

25.1.65.1 Constructors

25.1.65.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseBlobDescriptor.Create Constructor

[TnxBaseBlobDescriptor Class](#)

Pascal
`public constructor Create(
 aParent: TPersistent;
 const aHeaderSection: string
); virtual;`

■ Remarks

constructor.

■ See Also

[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Constructors > Create Constructor

25.1.65.2 Destructors

25.1.65.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseBlobDescriptor.Destroy Destructor

TnxBaseBlobDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Destructors > Destroy Destructor

25.1.65.3 Methods

25.1.65.3.1 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseBlobDescriptor.CheckValid Method

TnxBaseBlobDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseBlobDescriptor.

See Also

[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Methods > CheckValid Method

25.1.65.3.2 GetList Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseBlobDescriptor.GetList Method

TnxBaseBlobDescriptor Class

Pascal

```
public class function GetList: PStringList; override;
```

Remarks

This is GetList, a member of class TnxBaseBlobDescriptor.

See Also

[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Methods > GetList Method

25.1.65.3.3 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseBlobDescriptor.IsEqual Method**[TnxBaseBlobDescriptor Class](#)**Pascal**

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
) : Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseBlobDescriptor.

See Also[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Methods > IsEqual Method

25.1.65.3.4 LoadFromReader Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseBlobDescriptor.LoadFromReader Method**[TnxBaseBlobDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseBlobDescriptor.

See Also[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Methods > LoadFromReader Method

25.1.65.3.5 SaveToWriter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseBlobDescriptor.SaveToWriter Method**[TnxBaseBlobDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseBlobDescriptor.

❑ **See Also**

[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Methods > SaveToWriter Method

25.1.65.4 Properties

25.1.65.4.1 BlobEngine Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseBlobDescriptor.BlobEngine Property

[TnxBaseBlobDescriptor Class](#)

Pascal

```
public property BlobEngine: string;
```

❑ **Remarks**

The name of the Blob Engine used to read the blobs.

❑ **See Also**

[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Properties > BlobEngine Property

25.1.65.4.2 FileNumber Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseBlobDescriptor.FileNumber Property

[TnxBaseBlobDescriptor Class](#)

Pascal

```
public property FileNumber: Integer;
```

❑ **Remarks**

Returns the index of the FileDescriptor of the associated file.

❑ **See Also**

[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Properties > FileNumber Property

25.1.65.4.3 HeaderSection Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseBlobDescriptor.HeaderSection Property

[TnxBaseBlobDescriptor Class](#)

Pascal

```
public property HeaderSection: string;
```

❑ **Remarks**

Returns the header of the descriptor as string.

See Also[TnxBaseBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobDescriptor Class > Properties > HeaderSection Property

25.1.66 TnxBaseBlobStream Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBaseBlobStream Class

[nxdb](#) | [Members](#) | [Methods](#) | [Related Topics](#)**Pascal**

```
public TnxBaseBlobStream = class(TStream);
```

File[nxdb](#)**Remarks**

Base class for NexusDB blob streams.

Never create instances directly! Always use TnxDataSet.CreateBlobStream, which will create the right type of blob stream based on the state of the dataset.

The blob stream should be freed again as soon as possible. It must be freed before calling Post or moving to another record.

Related Topics

[TnxDataSet.CreateBlobStream](#) [TnxBlobStream](#) [TnxBlockModeBlobStream](#)
[TnxBatchAppendBlobStream](#) [TnxMemoBlobStream](#)

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseBlobStream Class

25.1.66.1 Destructors

25.1.66.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBaseBlobStream.Destroy Destructor

[TnxBaseBlobStream Class](#)**Pascal**

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also[TnxBaseBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobStream Class > Destructors > Destroy Destructor

25.1.66.2 Methods

25.1.66.2.1 Truncate Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseBlobStream.Truncate Method

TnxBaseBlobStream Class

Pascal

```
public procedure Truncate; virtual; abstract;
```

Remarks

This method truncates the blob data at the current position.

See Also

[TnxBaseBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlobStream Class > Methods > Truncate Method

25.1.67 TnxBaseBlockHeapDescriptor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseBlockHeapDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseBlockHeapDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

This is the base class for specifying Block management classes. BlockHeap descriptors are used by the Heap Descriptor

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseBlockHeapDescriptor Class

25.1.67.1 Methods

25.1.67.1.1 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseBlockHeapDescriptor.AreEqual Method

TnxBaseBlockHeapDescriptor Class

Pascal

```
public function IsEqual(
```

```
aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

▀ Remarks

This is IsEqual, a member of class TnxBaseBlockHeapDescriptor.

▀ See Also

[TnxBaseBlockHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlockHeapDescriptor Class > Methods > IsEqual Method

25.1.67.1.2 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseBlockHeapDescriptor.LoadFromReader Method

[TnxBaseBlockHeapDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

▀ Remarks

This is LoadFromReader, a member of class TnxBaseBlockHeapDescriptor.

▀ See Also

[TnxBaseBlockHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlockHeapDescriptor Class > Methods > LoadFromReader Method

25.1.67.1.3 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseBlockHeapDescriptor.SaveToWriter Method

[TnxBaseBlockHeapDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

▀ Remarks

This is SaveToWriter, a member of class TnxBaseBlockHeapDescriptor.

▀ See Also

[TnxBaseBlockHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlockHeapDescriptor Class > Methods > SaveToWriter Method

25.1.67.2 Properties

25.1.67.2.1 BlockHeapEngine Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseBlockHeapDescriptor.BlockHeapEngine Property

TnxBaseBlockHeapDescriptor Class

Pascal

```
public property BlockHeapEngine: string;
```

Remarks

The name of the Block Heap Engine.

See Also

[TnxBaseBlockHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseBlockHeapDescriptor Class > Properties > BlockHeapEngine Property

25.1.68 TnxBaseCommandHandler Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseCommandHandler Class

[nxIITransport](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxBaseCommandHandler = class(TnxStateComponent);
```

File

[nxIITransport](#)

Remarks

The base for all command handlers. The command handler is responsible for translating received message into actual functionality by decoding the messages and calling the appropriate methods.

See Also

[nxIITransport](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxBaseCommandHandler Class

25.1.68.1 Constructors

25.1.68.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseCommandHandler.Create Constructor

TnxBaseCommandHandler Class

Pascal

```
public constructor Create(
  aOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxBaseCommandHandler Class](#)

You are here: Symbol Reference > Classes > TnxBaseCommandHandler Class > Constructors > Create Constructor

25.1.68.2 Destructors

25.1.68.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseCommandHandler.Destroy Destructor

[TnxBaseCommandHandler Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseCommandHandler Class](#)

You are here: Symbol Reference > Classes > TnxBaseCommandHandler Class > Destructors > [Destroy Destructor](#)

25.1.68.3 Methods

25.1.68.3.1 UISettingsVisible Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseCommandHandler.UISettingsVisible Method

[TnxBaseCommandHandler Class](#)

Pascal

```
public function UISettingsVisible: Boolean; override;
```

Remarks

Result is always set to false, cause the command handler settings are never visible in the UI.

See Also

[TnxBaseCommandHandler Class](#)

You are here: Symbol Reference > Classes > TnxBaseCommandHandler Class > Methods > [UISettingsVisible Method](#)

25.1.69 TnxBaseComponentConfiguration Class

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseComponentConfiguration Class

[nxIIComponent](#) | Members | Methods

Pascal

```
public TnxBaseComponentConfiguration = class;
```

File[nxIIComponent](#)**Remarks**

An abstract prototype for a Component configuration class. An actual implementation must implement all functions.

See Also
[nxIIComponent](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxBaseComponentConfiguration Class

25.1.69.1 Methods

25.1.69.1.1 GetValue Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseComponentConfiguration.GetValue Method**[TnxBaseComponentConfiguration Class](#)**Pascal**

```
public function GetValue(
  aName: String;
  aDefault: Variant
): Variant; virtual;
```

Remarks

Get the value associated with aName as Variant.

See Also[TnxBaseComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxBaseComponentConfiguration Class > Methods > GetValue Method

25.1.69.1.2 GetValueStream Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseComponentConfiguration.GetValueStream Method**[TnxBaseComponentConfiguration Class](#)**Pascal**

```
public procedure GetValueStream(
  aName: String;
  aStream: TStream
); virtual;
```

Remarks

Get the value associated with aName as stream.

See Also[TnxBaseComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxBaseComponentConfiguration Class > Methods > GetValueStream Method

25.1.69.1.3 SetValue Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseComponentConfiguration.SetValue Method**[TnxBaseComponentConfiguration Class](#)**Pascal**

```
public procedure SetValue(
    aName: String;
    aValue: Variant
); virtual;
```

Remarks

Set the value associated with aName as Variant.

See Also[TnxBaseComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxBaseComponentConfiguration Class > Methods > SetValue Method

25.1.69.1.4 SetValueStream Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseComponentConfiguration.SetValueStream Method**[TnxBaseComponentConfiguration Class](#)**Pascal**

```
public procedure SetValueStream(
    aName: String;
    aStream: TStream
); virtual;
```

Remarks

Set the value associated with aName as stream.

See Also[TnxBaseComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxBaseComponentConfiguration Class > Methods > SetValueStream Method

25.1.70 TnxBaseConnectionFilter Class*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxBaseConnectionFilter Class**[nxtwWinsockTransport](#)**Pascal**

```
public TnxBaseConnectionFilter = class(TnxStateComponent);
```

File[nxtwWinsockTransport](#)**Remarks**

Base Component used to filter connections.

See Also[nxtwWinsockTransport](#)*You are here:* Symbol Reference > Classes > TnxBaseConnectionFilter Class

25.1.71 TnxBaseCustomDescriptor Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBaseCustomDescriptor Class

[nxsdDataDictionary](#) | Members | Methods | Properties**Pascal**

```
public TnxBaseCustomDescriptor = class(TnxDictionaryItem);
```

File[nxsdDataDictionary](#)**Remarks**

A Custom Descriptor is the base class for Custom Descriptors. Developers should use this class as a base for creating their own Descriptors and store them with tables.

See Also[nxsdDataDictionary](#)
Members
Methods
Properties*You are here:* Symbol Reference > Classes > TnxBaseCustomDescriptor Class

25.1.71.1 Methods

25.1.71.1.1 CheckValid Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBaseCustomDescriptor.CheckValid Method

[TnxBaseCustomDescriptor Class](#)**Pascal**

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseCustomDescriptor.

See Also[TnxBaseCustomDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxBaseCustomDescriptor Class > Methods > CheckValid Method

25.1.71.1.2 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBaseCustomDescriptor.AreEqual Method

[TnxBaseCustomDescriptor Class](#)**Pascal**

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

■ Remarks

This is IsEqual, a member of class TnxBaseCustomDescriptor.

■ See Also

[TnxBaseCustomDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseCustomDescriptor Class > Methods > IsEqual Method

25.1.71.2 Properties

25.1.71.2.1 Name Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseCustomDescriptor.Name Property

[TnxBaseCustomDescriptor Class](#)

Pascal

```
public property Name: string;
```

■ Remarks

Returns the name of the Custom Descriptor.

■ See Also

[TnxBaseCustomDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseCustomDescriptor Class > Properties > Name Property

25.1.72 TnxBaseDefaultValueDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseDefaultValueDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseDefaultValueDescriptor = class(TnxDictionaryItem);
```

■ File

[nxsdDataDictionary](#)

■ Remarks

This is the base class for descriptor that describe how a default value of a Field is set. This is an abstract class and must be overridden!

■ See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class

25.1.72.1 Methods

25.1.72.1.1 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseDefaultValueDescriptor.CheckValid Method

TnxBaseDefaultValueDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseDefaultValueDescriptor.

See Also

[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Methods > [CheckValid Method](#)

25.1.72.1.2 GetList Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseDefaultValueDescriptor.GetList Method

TnxBaseDefaultValueDescriptor Class

Pascal

```
public class function GetList: PStringList; override;
```

Remarks

return the list of default value descriptors.

See Also

[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Methods > [GetList Method](#)

25.1.72.1.3 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseDefaultValueDescriptor.AreEqual Method

TnxBaseDefaultValueDescriptor Class

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseDefaultValueDescriptor.

See Also

[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Methods > [IsEqual Method](#)

25.1.72.1.4 LoadFromReader Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseDefaultValueDescriptor.LoadFromReader Method**[TnxBaseDefaultValueDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseDefaultValueDescriptor.

See Also[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Methods > LoadFromReader Method

25.1.72.1.5 SaveToWriter Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseDefaultValueDescriptor.SaveToWriter Method**[TnxBaseDefaultValueDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseDefaultValueDescriptor.

See Also[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Methods > SaveToWriter Method

25.1.72.2 Properties

25.1.72.2.1 ApplyAt Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseDefaultValueDescriptor.ApplyAt Property**[TnxBaseDefaultValueDescriptor Class](#)**Pascal**

```
public property ApplyAt: TnxApplyAtSet;
```

Remarks

Apply the validation object on the server and/or client?

See Also[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Properties > ApplyAt Property

25.1.72.2.2 ApplyOnInsert Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseDefaultValueDescriptor.ApplyOnInsert Property

TnxBaseDefaultValueDescriptor Class

Pascal

```
public property ApplyOnInsert: Boolean;
```

Remarks

Should the check be applied on Insert of new record?

See Also

[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Properties > ApplyOnInsert Property

25.1.72.2.3 ApplyOnModify Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseDefaultValueDescriptor.ApplyOnModify Property

TnxBaseDefaultValueDescriptor Class

Pascal

```
public property ApplyOnModify: Boolean;
```

Remarks

Should the check be applied on record modification?

See Also

[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Properties > ApplyOnModify Property

25.1.72.2.4 OverwriteNonNull Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseDefaultValueDescriptor.OverwriteNonNull Property

TnxBaseDefaultValueDescriptor Class

Pascal

```
public property OverwriteNonNull: Boolean;
```

Remarks

???

See Also

[TnxBaseDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseDefaultValueDescriptor Class > Properties > OverwriteNonNull Property

25.1.73 TnxBaseDirectTransport Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseDirectTransport Class

[nxIITransport](#)

Pascal

```
public TnxBaseDirectTransport = class(TnxBaseTransport);
```

File

[nxIITransport](#)

Remarks

This is the base class for all *real* transports.

See Also

[nxIITransport](#)

You are here: Symbol Reference > Classes > TnxBaseDirectTransport Class

25.1.74 TnxBaseEngineExtender Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineExtender Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseEngineExtender = class(TnxObject);
```

File

[nxsdServerEngine](#)

Remarks

This is the base for all Extenders used in NexusDB. An extender gets notified about actions in the database core and can change or extend the default behavior. It is generally always created from a TnxBaseEngineMonitor.

See Also

[nxsdServerEngine](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseEngineExtender Class

25.1.74.1 Constructors

25.1.74.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineExtender.Create Constructor

[TnxBaseEngineExtender Class](#)

Pascal

```
public constructor Create(
  aMonitor: TnxBaseEngineMonitor;
  aExtendableObject: TnxExtendableServerObject;
```

) ;

Parameters

Parameters	Description
aMonitor	the Monitor that creates the Extender
aExtendableObject	the object the Extender extends.

Remarks

constructor

See Also

[TnxBaseEngineExtender Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineExtender Class > Constructors > Create Constructor

25.1.74.2 Destructors

25.1.74.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineExtender.Destroy Destructor

[TnxBaseEngineExtender Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseEngineExtender Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineExtender Class > Destructors > Destroy Destructor

25.1.74.3 Methods

25.1.74.3.1 Notify Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineExtender.Notify Method

[TnxBaseEngineExtender Class](#)

Pascal

```
public function Notify(
  aAction: TnxEngineAction;
  aBefore: Boolean;
  const aArgs: array of const
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aAction	the type of action executed
aBefore	if this is true then the server engine is just about to execute the given action, otherwise it was just

	executed.
aArgs	the parameters used for the given action type.

Remarks

This function is called for all actions executed in the extended server object ExtendableObject.

See Also

[TnxBaseEngineExtender Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineExtender Class > Methods > Notify Method

25.1.74.4 Properties

25.1.74.4.1 ExtendableObject Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineExtender.ExtendableObject Property

[TnxBaseEngineExtender Class](#)

Pascal

```
public property ExtendableObject: TnxExtendableObject;
```

Remarks

This property returns the server object the extender extends.

See Also

[TnxBaseEngineExtender Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineExtender Class > Properties > ExtendableObject Property

25.1.74.4.2 Monitor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineExtender.Monitor Property

[TnxBaseEngineExtender Class](#)

Pascal

```
public property Monitor: TnxBaseEngineMonitor;
```

Remarks

This property returns the Monitor the extender was created from.

See Also

[TnxBaseEngineExtender Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineExtender Class > Properties > Monitor Property

25.1.75 TnxBaseEngineMonitor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineMonitor Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseEngineMonitor = class(TnxStateComponent);
```

File

[nxsdServerEngine](#)

Remarks

This is the base engine monitor. An engine monitor is responsible for creating and freeing engine extenders. A monitor is notified about every creation and destruction of a server object.

See Also

[nxsdServerEngine](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseEngineMonitor Class

25.1.75.1 Constructors

25.1.75.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineMonitor.Create Constructor

[TnxBaseEngineMonitor Class](#)

Pascal

```
public constructor Create(
    aOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxBaseEngineMonitor Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineMonitor Class > Constructors > Create Constructor

25.1.75.2 Destructors

25.1.75.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseEngineMonitor.Destroy Destructor

[TnxBaseEngineMonitor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseEngineMonitor Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineMonitor Class > Destructors > Destroy Destructor

25.1.75.3 Methods

25.1.75.3.1 ExtendableObjectCreated Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseEngineMonitor.ExtendableObjectCreated Method

TnxBaseEngineMonitor Class

Pascal

```
public procedure ExtendableObjectCreated(
    aExtendableObject: TnxExtendableServerObject
); virtual;
```

Parameters

Parameters

aExtendableObject

Description

This is the pointer to the object that was just created.

Remarks

This method is called every time a server object is created within the linked Server engine. Override this function to create an Extender or manipulate the initial settings of a server object.

See Also

[TnxBaseEngineMonitor Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineMonitor Class > Methods > ExtendableObjectCreated Method

25.1.75.3.2 ExtendableObjectDestroyed Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseEngineMonitor.ExtendableObjectDestroyed Method

TnxBaseEngineMonitor Class

Pascal

```
public procedure ExtendableObjectDestroyed(
    aExtendableObject: TnxExtendableServerObject
); virtual;
```

Remarks

This is ExtendableObjectDestroyed, a member of class TnxBaseEngineMonitor.

See Also

[TnxBaseEngineMonitor Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineMonitor Class > Methods > ExtendableObjectDestroyed Method

25.1.75.4 Properties

25.1.75.4.1 ServerEngine Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseEngineMonitor.ServerEngine Property

[TnxBaseEngineMonitor Class](#)**Pascal**

```
published property ServerEngine: TnxBaseServerEngine;
```

Remarks

This property links the Monitor to a server engine.

See Also

[TnxBaseEngineMonitor Class](#)

You are here: Symbol Reference > Classes > TnxBaseEngineMonitor Class > Properties > ServerEngine Property

25.1.76 TnxBaseFieldsValidationDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseFieldsValidationDescriptor Class

[nxsdDataDictionary](#)

Pascal

```
public TnxBaseFieldsValidationDescriptor = class(TnxBaseFieldValidationDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This is the Base class for descriptor that describe how a Table is validated. This is an abstract class and must be overridden!

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > TnxBaseFieldsValidationDescriptor Class

25.1.77 TnxBaseFieldValidationDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseFieldValidationDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseFieldValidationDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

This is the base class for descriptor that describe how a Field is validated. This is an abstract class and must be overridden!

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class

25.1.77.1 Methods

25.1.77.1.1 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldValidationDescriptor.CheckValid Method

TnxBaseFieldValidationDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseFieldValidationDescriptor.

See Also

[TnxBaseFieldValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class > Methods > CheckValid Method

25.1.77.1.2 GetList Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldValidationDescriptor.GetList Method

TnxBaseFieldValidationDescriptor Class

Pascal

```
public class function GetList: PStringList; override;
```

Remarks

Returns the internal List of Validation Descriptor classes.

See Also

[TnxBaseFieldValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class > Methods > GetList Method

25.1.77.1.3 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldValidationDescriptor.AreEqual Method

TnxBaseFieldValidationDescriptor Class

Pascal

```
public function IsEqual(
  aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseFieldValidationDescriptor.

See Also

[TnxBaseFieldValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class > Methods > IsEqual Method

25.1.77.1.4 LoadFromReader Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldValidationDescriptor.LoadFromReader Method

TnxBaseFieldValidationDescriptor Class

Pascal

```
public procedure LoadFromReader(
  aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseFieldValidationDescriptor.

See Also

[TnxBaseFieldValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class > Methods > LoadFromReader Method

25.1.77.1.5 SaveToWriter Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldValidationDescriptor.SaveToWriter Method

TnxBaseFieldValidationDescriptor Class

Pascal

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseFieldValidationDescriptor.

See Also

[TnxBaseFieldValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class > Methods > SaveToWriter Method

25.1.77.2 Properties

25.1.77.2.1 ApplyAt Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldValidationDescriptor.ApplyAt Property

TnxBaseFieldValidationDescriptor Class

Pascal

```
public property ApplyAt: TnxApplyAtSet;
```

Remarks

Defines if the validation should be done at client and/or server side.

See Also[TnxBaseFieldValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class > Properties > [ApplyAt Property](#)

25.1.77.2.2 Name Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseFieldValidationDescriptor.Name Property**[TnxBaseFieldValidationDescriptor Class](#)**Pascal**

```
public property Name: string;
```

Remarks

This is Name, a member of class TnxBaseFieldValidationDescriptor.

See Also[TnxBaseFieldValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseFieldValidationDescriptor Class > Properties > [Name Property](#)

25.1.78 TnxBaseHeapDescriptor Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor Class**[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxBaseHeapDescriptor = class(TnxDictionaryItem);
```

File[nxsdDataDictionary](#)**Remarks**

This is the base class for specifying the Heap management classes.

See Also
[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class

25.1.78.1 Destructors

25.1.78.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor.Destroy Destructor**[TnxBaseHeapDescriptor Class](#)**Pascal**

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Destructors > Destroy Destructor

25.1.78.2 Methods

25.1.78.2.1 AddBlockHeapDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseHeapDescriptor.AddBlockHeapDescriptor Method

[TnxBaseHeapDescriptor Class](#) | [Related Topics](#)

Pascal

```
public function AddBlockHeapDescriptor(
  aBlockHeapClass: TnxBaseBlockHeapDescriptorClass = nil
): TnxBaseBlockDescriptor;
```

Remarks

Sets the Block Heap Descriptor used for Block management

Related Topics

[AddBlockHeapDescriptor](#), [RemoveBlockHeapDescriptor](#), [BlockHeapDescriptor](#)

See Also

[TnxBaseHeapDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Methods > [AddBlockHeapDescriptor Method](#)

25.1.78.2.2 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseHeapDescriptor.CheckValid Method

[TnxBaseHeapDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseHeapDescriptor.

See Also

[TnxBaseHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Methods > [CheckValid Method](#)

25.1.78.2.3 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor.IsEqual Method**[TnxBaseHeapDescriptor Class](#)**Pascal**

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseHeapDescriptor.

See Also[TnxBaseHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Methods > IsEqual Method

25.1.78.2.4 LoadFromReader Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor.LoadFromReader Method**[TnxBaseHeapDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseHeapDescriptor.

See Also[TnxBaseHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Methods > LoadFromReader Method

25.1.78.2.5 RemoveBlockHeapDescriptor Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor.RemoveBlockHeapDescriptor Method**[TnxBaseHeapDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public procedure RemoveBlockHeapDescriptor;
```

Remarks

Sets the Block Heap descriptor to nil.

Related Topics[AddBlockHeapDescriptor](#), [RemoveBlockHeapDescriptor](#), [BlockHeapDescriptor](#).**See Also**[TnxBaseHeapDescriptor Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Methods > RemoveBlockHeapDescriptor Method

25.1.78.2.6 SaveToWriter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor.SaveToWriter Method**[TnxBaseHeapDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseHeapDescriptor.

See Also[TnxBaseHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Methods > SaveToWriter Method

25.1.78.3 Properties

25.1.78.3.1 BlockHeapDescriptor Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor.BlockHeapDescriptor Property**[TnxBaseHeapDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public property BlockHeapDescriptor: TnxBaseBlockHeapDescriptor;
```

Remarks

Returns the Block Heap Descriptor

Related Topics

[AddBlockHeapDescriptor](#), [RemoveBlockHeapDescriptor](#), [BlockHeapDescriptor](#).

See Also

[TnxBaseHeapDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Properties > BlockHeapDescriptor Property

25.1.78.3.2 HeapEngine Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseHeapDescriptor.HeapEngine Property**[TnxBaseHeapDescriptor Class](#)**Pascal**

```
public property HeapEngine: string;
```

Remarks

The name of the Heap Engine.

See Also

[TnxBaseHeapDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseHeapDescriptor Class > Properties > HeapEngine Property

25.1.79 TnxBaseIndicesDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseIndicesDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

TnxBaseIndicesDescriptor manages a list of IndexDescriptor classes associated with the data dictionary

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class

25.1.79.1 Constructors

25.1.79.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.Create Constructor

[TnxBaseIndicesDescriptor Class](#)

Pascal

```
public constructor Create(
  aParent: TPersistent;
  const aHeaderSection: string
);
```

Remarks

constructor

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Constructors > Create Constructor

25.1.79.2 Destructors

25.1.79.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseIndicesDescriptor.Destroy Destructor

TnxBaseIndicesDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Destructors > Destroy Destructor

25.1.79.3 Methods

25.1.79.3.1 AddIndex

25.1.79.3.1.1 AddIndex Method (TnxIndexDescriptor)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseIndicesDescriptor.AddIndex Method (TnxIndexDescriptor)

TnxBaseIndicesDescriptor Class

Pascal

```
public function AddIndex(
    aIndexDescriptor: TnxIndexDescriptor
): TnxIndexDescriptor; overload;
```

Parameters

Parameters

aIndexClass

Description

the class of the instance to be created.

Remarks

This function adds a new instance of an Index Descriptor to the indices descriptor and returns the newly created Instance.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > AddIndex > AddIndex Method (TnxIndexDescriptor)

25.1.79.3.1.2 AddIndex Method (string, Integer, Boolean, string, TnxKeyDescriptorClass, TnxIndexDescriptorClass)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseIndicesDescriptor.AddIndex Method (string, Integer, Boolean, string, TnxKeyDescriptorClass, TnxIndexDescriptorClass)

TnxBaseIndicesDescriptor Class

Pascal

```

public function AddIndex(
    const aName: string;
    aFileNumber: Integer;
    aDups: Boolean;
    const aDesc: string = '';
    aKeyClass: TnxKeyDescriptorClass = nil;
    aIndexClass: TnxIndexDescriptorClass = nil
) : TnxIndexDescriptor; virtual; overload;

```

Parameters

Parameters	Description
aName	the name of the index
aFileNumber	the file the Index is stored in
aDups	set to True if duplicates are allowed for this index
aDesc	a human readable description of the field
aKeyClass	the class for the Key Descriptor to use
aIndexClass	the class of the instance to be created.

Remarks

This function adds a new instance of an Index Descriptor to the indices descriptor and returns the newly created Instance.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > AddIndex > AddIndex Method (string, Integer, Boolean, string, TnxKeyDescriptorClass, TnxIndexDescriptorClass)

25.1.79.3.2 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.CheckValid Method

[TnxBaseIndicesDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseIndicesDescriptor.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > CheckValid Method

25.1.79.3.3 Clear Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.Clear Method

[TnxBaseIndicesDescriptor Class](#)

Pascal

```
public procedure Clear(
    aNotify: Boolean
); virtual;
```

Remarks

Empties the list of Index Descriptors

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > Clear Method

25.1.79.3.4 GetDescriptorFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.GetDescriptorFromName Method

[TnxBaseIndicesDescriptor Class](#)

Pascal

```
public function GetDescriptorFromName(
    const aIndexName: string
): TnxIndexDescriptor;
```

Remarks

Return the index descriptor for a given index name, or nil if not found.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > GetDescriptorFromName Method

25.1.79.3.5 GetIndexFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.GetIndexFromName Method

[TnxBaseIndicesDescriptor Class](#)

Pascal

```
public function GetIndexFromName(
    const aIndexName: string
): Integer;
```

Remarks

Return the index number for a given index name, or -1 if not found.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > GetIndexFromName Method

25.1.79.3.6 InsertIndexAt Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.InsertIndexAt Method

TnxBaseIndicesDescriptor Class

Pascal

```
public function InsertIndexAt(
    aInx: Integer;
    const aName: string;
    aFileNumber: Integer;
    aDups: Boolean;
    const aDesc: string = '';
    aKeyClass: TnxKeyDescriptorClass = nil;
    aIndexClass: TnxIndexDescriptorClass = nil
): TnxIndexDescriptor;
```

Parameters

Parameters	Description
aInx	the index of the new field
aName	the name of the index
aFileNumber	the file the Index is stored in
aDups	set to True if duplicates are allowed for this index
aDesc	a human readable description of the field
aKeyClass	the class for the Key Descriptor to use
aIndexClass	the class of the instance to be created.

Remarks

This function adds a new instance of an Index Descriptor to the indices descriptor and returns the newly created Instance. The index is inserted into the array at position aInx and the rest of the index array indexes are adjusted.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > InsertIndexAt Method

25.1.79.3.7 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.IsEqual Method

TnxBaseIndicesDescriptor Class

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseIndicesDescriptor.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > IsEqual Method

25.1.79.3.8 LoadFromReader Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseIndicesDescriptor.LoadFromReader Method**[TnxBaseIndicesDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseIndicesDescriptor.

See Also[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > [LoadFromReader Method](#)

25.1.79.3.9 MoveIndex Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseIndicesDescriptor.MoveIndex Method**[TnxBaseIndicesDescriptor Class](#)**Pascal**

```
public procedure MoveIndex(
    aFrom: Integer;
    aTo: Integer
);
```

Remarks

Moves an Index Descriptor in the list of descriptors from aFrom to the aTo position

See Also[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > [MoveIndex Method](#)

25.1.79.3.10 RemoveIndex

25.1.79.3.10.1 RemoveIndex Method (Integer)

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseIndicesDescriptor.RemoveIndex Method (Integer)**[TnxBaseIndicesDescriptor Class](#)**Pascal**

```
public procedure RemoveIndex(
    aInx: Integer
); overload;
```

Parameters**Parameters****Description**

aIdx

the index of the descriptor to be removed in the list of Index Descriptors.

Remarks

Remove an Index Descriptor from the Data Dictionary.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > RemoveIndex > RemoveIndex Method (Integer)

25.1.79.3.10.2 RemoveIndex Method (string)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.RemoveIndex Method (string)

[TnxBaseIndicesDescriptor Class](#)

Pascal

```
public procedure RemoveIndex(
    const aName: string
); overload;
```

Parameters**Parameters**

aName

Description

the name of the descriptor to be removed in the list of Index Descriptors.

Remarks

Remove an Index Descriptor from the Data Dictionary.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > RemoveIndex > RemoveIndex Method (string)

25.1.79.3.11 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseIndicesDescriptor.SaveToWriter Method

[TnxBaseIndicesDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseIndicesDescriptor.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Methods > SaveToWriter Method

25.1.79.4 Properties

25.1.79.4.1 HeaderSection Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseIndicesDescriptor.HeaderSection Property

TnxBaseIndicesDescriptor Class

Pascal

```
public property HeaderSection: string;
```

Remarks

Returns the Header of the descriptor as string.

See Also

[TnxBaseIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseIndicesDescriptor Class > Properties > HeaderSection Property

25.1.80 TnxBaseKeyDescriptor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseKeyDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseKeyDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

The base class of all Key Descriptors. A Key descriptor is used to create a key out of a record. Mainly it is used for creating indexes, but can of course also be used for other purposes.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseKeyDescriptor Class

25.1.80.1 Destructors

25.1.80.1.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseKeyDescriptor.Destroy Destructor

TnxBaseKeyDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxBaseKeyDescriptor.

See Also[TnxBaseKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseKeyDescriptor Class > Destructors > Destroy Destructor

25.1.80.2 Methods

25.1.80.2.1 LoadFromReader Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseKeyDescriptor.LoadFromReader Method**[TnxBaseKeyDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseKeyDescriptor.

See Also[TnxBaseKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseKeyDescriptor Class > Methods > LoadFromReader Method

25.1.80.2.2 SaveToWriter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseKeyDescriptor.SaveToWriter Method**[TnxBaseKeyDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseKeyDescriptor.

See Also[TnxBaseKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseKeyDescriptor Class > Methods > SaveToWriter Method

25.1.80.3 Properties

25.1.80.3.1 KeyLen Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseKeyDescriptor.KeyLen Property**[TnxBaseKeyDescriptor Class](#)

Pascal

```
public property KeyLen: TnxWord32;
```

Remarks

This property returns the length of the created keys.

See Also

[TnxBaseKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseKeyDescriptor Class > Properties > KeyLen Property

25.1.81 TnxBaseLog Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseLog Class

[nxlComponent](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseLog = class(TnxRegisterableComponent);
```

File

[nxlComponent](#)

Remarks

Base logging class.

See Also

[nxlComponent](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class

25.1.81.1 Methods

25.1.81.1.1 Clear Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseLog.Clear Method

[TnxBaseLog Class](#)

Pascal

```
public procedure Clear; virtual;
```

Remarks

Clear the log.

See Also

[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Methods > Clear Method

25.1.81.1.2 Flush Method

*NexusDB V2 VCL Reference***TnxBaseLog.Flush Method**[Contents | Index](#)

TnxBaseLog Class

Pascal

```
public procedure Flush; virtual;
```

Remarks

Flush the log.

See Also
[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Methods > Flush Method

25.1.81.1.3 WriteBlock Method

*NexusDB V2 VCL Reference***TnxBaseLog.WriteBlock Method**[Contents | Index](#)

TnxBaseLog Class

Pascal

```
public procedure WriteBlock(
  const S: string;
  Buf: Pointer;
  BufLen: TnxMemSize
); virtual; abstract;
```

Remarks

Writes a string and a block of memory to the log.

See Also
[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Methods > WriteBlock Method

25.1.81.1.4 WriteLogData Method

*NexusDB V2 VCL Reference***TnxBaseLog.WriteLogData Method**[Contents | Index](#)

TnxBaseLog Class

Pascal

```
public procedure WriteLogData(
  LogData: InxLogData
); virtual; abstract;
```

Remarks

Writes an object that implements InxLogData to the log.

See Also
[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Methods > WriteLogData Method

25.1.81.1.5 WriteString

25.1.81.1.5.1 WriteString Method (string)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseLog.WriteString Method (string)

TnxBaseLog Class

Pascal

```
public procedure WriteString(
    const aMsg: string
); virtual; abstract; overload;
```

Remarks

Writes a string to the log.

See Also

[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Methods > WriteString > WriteString Method (string)

25.1.81.1.5.2 WriteString Method (string, array of const)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseLog.WriteString Method (string, array of const)

TnxBaseLog Class

Pascal

```
public procedure WriteString(
    const aMsg: string;
    const aArgs: array of const
); virtual; abstract; overload;
```

Remarks

Writes a string to the log.

See Also

[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Methods > WriteString > WriteString Method (string, array of const)

25.1.81.1.6 WriteStrings Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseLog.WriteStrings Method

TnxBaseLog Class

Pascal

```
public procedure WriteStrings(
    const Msgs: array of string
); virtual; abstract;
```

Remarks

Writes a string to the log.

See Also

[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Methods > WriteStrings Method

25.1.81.2 Properties

25.1.81.2.1 Enabled Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseLog.Enabled Property**[TnxBaseLog Class](#)**Pascal**

```
public property Enabled: Boolean;
```

Remarks

True if the log is currently enabled/active.

See Also[TnxBaseLog Class](#)

You are here: Symbol Reference > Classes > TnxBaseLog Class > Properties > Enabled Property

25.1.82 TnxBasePluginCommandHandler Class*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxBasePluginCommandHandler Class**[nxllTransport](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxBasePluginCommandHandler = class(TnxBasePlugin);
```

File[nxllTransport](#)**Remarks**

The base for Plugin Command handlers. A plugin command handler is responsible for decoding and executing the appropriate methods for plugins.

See Also
[nxllTransport](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBasePluginCommandHandler Class

25.1.82.1 Methods

25.1.82.1.1 UISettingsVisible Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBasePluginCommandHandler.UISettingsVisible Method**[TnxBasePluginCommandHandler Class](#)**Pascal**

```
public function UISettingsVisible: Boolean; override;
```

Remarks

This function always returns false, cause the settings should never be visible in the GUI.

See Also

[TnxBasePluginCommandHandler Class](#)

You are here: Symbol Reference > Classes > TnxBasePluginCommandHandler Class > Methods > UISettingsVisible Method

25.1.82.2 Properties

25.1.82.2.1 CommandHandler Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePluginCommandHandler.CommandHandler Property

[TnxBasePluginCommandHandler Class](#)

Pascal

```
published property CommandHandler: TnxBaseCommandHandler;
```

Remarks

This is the main command handler the plugin command handler is attached to.

See Also

[TnxBasePluginCommandHandler Class](#)

You are here: Symbol Reference > Classes > TnxBasePluginCommandHandler Class > Properties > CommandHandler Property

25.1.83 TnxBasePooledTransport Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport Class

[nxptBasePooledTransport](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBasePooledTransport = class(TnxBaseDirectTransport);
```

File

[nxptBasePooledTransport](#)

Remarks

The base class for all transports that support connection pooling.

See Also

[nxptBasePooledTransport](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class

25.1.83.1 Constructors

25.1.83.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBasePooledTransport.Create Constructor

TnxBasePooledTransport Class

Pascal

```
public constructor Create(
    AOwner: TComponent
); override;
```

Remarks

constructor

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Constructors > Create Constructor

25.1.83.2 Destructors

25.1.83.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBasePooledTransport.Destroy Destructor

TnxBasePooledTransport Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Destructors > Destroy Destructor

25.1.83.3 Methods

25.1.83.3.1 Broadcast Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBasePooledTransport.Broadcast Method

TnxBasePooledTransport Class

Pascal

```
public procedure Broadcast(
    aMsgID: TnxMsgID;
    aThreadPriority: TnxThreadPriority;
    arequestData: Pointer;
    arequestDataLen: TnxWord32;
    aTimeOut: Integer
); override;
```

Remarks

This is Broadcast, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > Broadcast Method

25.1.83.3.2 ConnectionCount Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.ConnectionCount Method

[TnxBasePooledTransport Class](#)

Pascal

```
public function ConnectionCount: Integer; override;
```

Remarks

This is ConnectionCount, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > ConnectionCount Method

25.1.83.3.3 EstablishConnection Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.EstablishConnection Method

[TnxBasePooledTransport Class](#)

Pascal

```
public function EstablishConnection(
  aTransportID: TnxTransportID;
  const aUserName: string;
  const aPassword: string;
  aTimeout: Integer;
  aClientVersion: Integer;
  out aServerVersion: Integer;
  out aSessionID: TnxSessionID
): TnxResult; override;
```

Remarks

This is EstablishConnection, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > EstablishConnection Method

25.1.83.3.4 GetConfigSettings Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.GetConfigSettings Method

TnxBasePooledTransport Class

```
Pascal
public procedure GetConfigSettings(
    aSettings: TnxBaseSettings
); override;
```

Remarks

This is GetConfigSettings, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > [GetConfigSettings Method](#)

25.1.83.3.5 GetNetIdle Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.GetNetIdle Method**TnxBasePooledTransport Class**

```
Pascal
public class function GetNetIdle(
    out aNetIdle: TnxNetIdleEvent;
    out aInterval: Cardinal;
    out aMask: Cardinal
): Boolean;
```

Remarks

This is GetNetIdle, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > [GetNetIdle Method](#)

25.1.83.3.6 GetStatsCaptions Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.GetStatsCaptions Method**TnxBasePooledTransport Class**

```
Pascal
public procedure GetStatsCaptions(
    const aList: TStrings
); override;
```

Remarks

This is GetStatsCaptions, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > [GetStatsCaptions Method](#)

25.1.83.3.7 GetStatsValues Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.GetStatsValues Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public procedure GetStatsValues(
  const aList: TStrings
); override;
```

Remarks

This is GetStatsValues, a member of class TnxBasePooledTransport.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > [GetStatsValues Method](#)

25.1.83.3.8 IsConnected Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.IsConnected Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public function IsConnected: Boolean; override;
```

Remarks

This is IsConnected, a member of class TnxBasePooledTransport.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > [IsConnected Method](#)

25.1.83.3.9 LoadConfig Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.LoadConfig Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public procedure LoadConfig(
  aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

This is LoadConfig, a member of class TnxBasePooledTransport.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > [LoadConfig Method](#)

25.1.83.3.10 LoadSettingsFromStream Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.LoadSettingsFromStream Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public procedure LoadSettingsFromStream(
    aReader: TnxReader
); override;
```

Remarks

This is LoadSettingsFromStream, a member of class TnxBasePooledTransport.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > LoadSettingsFromStream Method

25.1.83.3.11 Post Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.Post Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public procedure Post(
    aTransportID: TnxTransportID;
    aSessionID: TnxSessionID;
    aThreadPriority: TnxThreadPriority;
    aMsgID: TnxMsgID;
    arequestData: Pointer;
    arequestDataLen: TnxWord32;
    aTimeout: Integer
); override;
```

Remarks

This is Post, a member of class TnxBasePooledTransport.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > Post Method

25.1.83.3.12 Request Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.Request Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public procedure Request(
    aTransportID: TnxTransportID;
    aSessionID: TnxSessionID;
    aThreadPriority: TnxThreadPriority;
    aMsgID: TnxMsgID;
    aTimeout: Integer;
```

```
arequestData: Pointer;
arequestDataLen: TnxWord32;
aReplyCallback: TnxReplyCallback;
aReplyCookie: Integer
); override;
```

Remarks

This is Request, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > Request Method

25.1.83.3.13 SaveConfig Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.SaveConfig Method

[TnxBasePooledTransport Class](#)

Pascal

```
public procedure SaveConfig(
    aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

This is SaveConfig, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > SaveConfig Method

25.1.83.3.14 SaveSettingsToStream Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.SaveSettingsToStream Method

[TnxBasePooledTransport Class](#)

Pascal

```
public procedure SaveSettingsToStream(
    aWriter: TnxWriter
); override;
```

Remarks

This is SaveSettingsToStream, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > SaveSettingsToStream Method

25.1.83.3.15 SetNetIdle Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBasePooledTransport.SetNetIdle Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public class procedure SetNetIdle(
    aNetIdle: TnxNetIdleEvent;
    aInterval: Cardinal;
    aMask: Cardinal
);
```

Remarks

This is SetNetIdle, a member of class TnxBasePooledTransport.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > SetNetIdle Method

25.1.83.3.16 TerminateConnection Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBasePooledTransport.TerminateConnection Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public procedure TerminateConnection(
    aTransportID: TnxTransportID;
    aSessionID: TnxSessionID;
    aTimeout: Integer
); override;
```

Remarks

This is TerminateConnection, a member of class TnxBasePooledTransport.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > TerminateConnection Method

25.1.83.3.17 Write Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBasePooledTransport.Write Method**[TnxBasePooledTransport Class](#)**Pascal**

```
public procedure Write(
    aConnection: Pointer;
    aTransportID: TnxTransportID;
    aSessionID: TnxSessionID;
    aThreadPriority: TnxThreadPriority;
    aMsgID: TnxMsgID;
    aMsgType: Byte;
```

```

    aErrorCode: TnxResult;
    aCompressType: Byte;
    aUncompressedSize: TnxWord32;
    aBuffer: Pointer;
    aBufferSize: TnxWord32
); override;

```

Remarks

This is Write, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Methods > Write Method

25.1.83.4 Properties

25.1.83.4.1 CallbackThreadCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.CallbackThreadCount Property

TnxBasePooledTransport Class

Pascal

```
public property CallbackThreadCount: Byte;
```

Remarks

Number of active callback threads.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > CallbackThreadCount Property

25.1.83.4.2 CallbackThreadPriority Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.CallbackThreadPriority Property

TnxBasePooledTransport Class

Pascal

```
public property CallbackThreadPriority: TThreadPriority;
```

Remarks

Priority of the callback thread.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > CallbackThreadPriority Property

25.1.83.4.3 CompressLimit Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.CompressLimit Property

[TnxBasePooledTransport Class](#)**Pascal**

```
published property CompressLimit: TnxWord32;
```

Remarks

Number of minimum length of the message for compression. If the message is shorter, compression is not used.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > CompressLimit Property

25.1.83.4.4 CompressType Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.CompressType Property**[TnxBasePooledTransport Class](#)**Pascal**

```
published property CompressType: Byte;
```

Remarks

The type of compression to use. Must be a registered compression engine id!!

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > CompressType Property

25.1.83.4.5 ConcurrentIOCPThreads Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.ConcurrentIOCPThreads Property**[TnxBasePooledTransport Class](#)**Pascal**

```
published property ConcurrentIOCPThreads: Byte;
```

Remarks

Number of IO Completion Port threads. Ideally the number of processors in the system.

See Also[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > ConcurrentIOCPThreads Property

25.1.83.4.6 HeartbeatInterval Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBasePooledTransport.HeartbeatInterval Property**[TnxBasePooledTransport Class](#)

Pascal

```
published property HeartbeatInterval: TnxWord32;
```

Remarks

The interval between keep alive heartbeat checks in ms.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > [TnxBasePooledTransport Class](#) > Properties > [HeartbeatInterval Property](#)

25.1.83.4.7 HeartbeatThreadPriority Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.HeartbeatThreadPriority Property

[TnxBasePooledTransport Class](#)

Pascal

```
published property HeartbeatThreadPriority: TThreadPriority;
```

Remarks

The priority of the "Heartbeat" background thread.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > [TnxBasePooledTransport Class](#) > Properties > [HeartbeatThreadPriority Property](#)

25.1.83.4.8 OverlappedClient Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.OverlappedClient Property

[TnxBasePooledTransport Class](#)

Pascal

```
published property OverlappedClient: Boolean;
```

Remarks

This is OverlappedClient, a member of class TnxBasePooledTransport.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > [TnxBasePooledTransport Class](#) > Properties > [OverlappedClient Property](#)

25.1.83.4.9 Port Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.Port Property

[TnxBasePooledTransport Class](#)

Pascal

```
published property Port: TnxWord16;
```

Remarks

The "Port" of the thread.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > Port Property

25.1.83.4.10 ServerThreadPriority Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.ServerThreadPriority Property

[TnxBasePooledTransport Class](#)

Pascal

```
published property ServerThreadPriority: TThreadPriority;
```

Remarks

The priority of the server thread.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > ServerThreadPriority Property

25.1.83.4.11 WatchdogInterval Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.WatchdogInterval Property

[TnxBasePooledTransport Class](#)

Pascal

```
published property WatchdogInterval: TnxWord32;
```

Remarks

The interval between keep alive watchdog checks in ms.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > WatchdogInterval Property

25.1.83.4.12 WatchdogThreadPriority Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBasePooledTransport.WatchdogThreadPriority Property

[TnxBasePooledTransport Class](#)

Pascal

```
published property WatchdogThreadPriority: TThreadPriority;
```

Remarks

The priority of the "Watchdog" background thread.

See Also

[TnxBasePooledTransport Class](#)

You are here: Symbol Reference > Classes > TnxBasePooledTransport Class > Properties > WatchdogThreadPriority Property

25.1.84 TnxBaseRecordCompressionDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordCompressionDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseRecordCompressionDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

A compression descriptor describes the compression engine used for the associated table.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseRecordCompressionDescriptor Class

25.1.84.1 Methods

25.1.84.1.1 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordCompressionDescriptor.IsEqual Method

[TnxBaseRecordCompressionDescriptor Class](#)

Pascal

```
public function IsEqual(
  aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseRecordCompressionDescriptor.

See Also

[TnxBaseRecordCompressionDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordCompressionDescriptor Class > Methods > IsEqual Method

25.1.84.1.2 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordCompressionDescriptor.LoadFromReader Method

[TnxBaseRecordCompressionDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseRecordCompressionDescriptor.

See Also

[TnxBaseRecordCompressionDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordCompressionDescriptor Class > Methods > LoadFromReader Method

25.1.84.1.3 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordCompressionDescriptor.SaveToWriter Method

[TnxBaseRecordCompressionDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseRecordCompressionDescriptor.

See Also

[TnxBaseRecordCompressionDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordCompressionDescriptor Class > Methods > SaveToWriter Method

25.1.84.2 Properties

25.1.84.2.1 RecordCompressionEngine Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordCompressionDescriptor.RecordCompressionEngine Property

[TnxBaseRecordCompressionDescriptor Class](#)

Pascal

```
public property RecordCompressionEngine: string;
```

Remarks

Returns the name name of the registered Record Compression Engine

See Also

[TnxBaseRecordCompressionDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordCompressionDescriptor Class > Properties > RecordCompressionEngine Property

25.1.85 TnxBaseRecordDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseRecordDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

This is the base class for Record Descriptors. A record descriptor is used for specifying the Record Engine used to read records from a table.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseRecordDescriptor Class

25.1.85.1 Destructors

25.1.85.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordDescriptor.Destroy Destructor

[TnxBaseRecordDescriptor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxBaseRecordDescriptor.

See Also

[TnxBaseRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Destructors > Destroy Destructor

25.1.85.2 Methods

25.1.85.2.1 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordDescriptor.CheckValid Method

[TnxBaseRecordDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxBaseRecordDescriptor.

See Also[TnxBaseRecordDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Methods > CheckValid Method

25.1.85.2.2 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseRecordDescriptor.IsEqual Method**[TnxBaseRecordDescriptor Class](#)**Pascal**

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseRecordDescriptor.

See Also[TnxBaseRecordDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Methods > IsEqual Method

25.1.85.2.3 LoadFromReader Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseRecordDescriptor.LoadFromReader Method**[TnxBaseRecordDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseRecordDescriptor.

See Also[TnxBaseRecordDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Methods > LoadFromReader Method

25.1.85.2.4 SaveToWriter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseRecordDescriptor.SaveToWriter Method**[TnxBaseRecordDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
```

```
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseRecordDescriptor.

See Also

[TnxBaseRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Methods > [SaveToWriter Method](#)

25.1.85.3 Properties

25.1.85.3.1 FileNumber Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordDescriptor.FileNumber Property

[TnxBaseRecordDescriptor Class](#)

Pascal

```
public property FileNumber: Integer;
```

Remarks

Returns the index of the FileDescriptor the record is associated with.

See Also

[TnxBaseRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Properties > [FileNumber Property](#)

25.1.85.3.2 HeaderSection Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordDescriptor.HeaderSection Property

[TnxBaseRecordDescriptor Class](#)

Pascal

```
public property HeaderSection: string;
```

Remarks

Returns the descriptor Header as string.

See Also

[TnxBaseRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Properties > [HeaderSection Property](#)

25.1.85.3.3 RecordEngine Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordDescriptor.RecordEngine Property

[TnxBaseRecordDescriptor Class](#)

Pascal

```
public property RecordEngine: string;
```

Remarks

The Name of the Record Engine used.

See Also

[TnxBaseRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseRecordDescriptor Class > Properties > RecordEngine Property

25.1.86 TnxBaseServer Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer Class

[Events](#) | [nxBaseServerComp](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseServer = class(TComponent);
```

File

[nxBaseServerComp](#)

Remarks

A base server component that encapsulates generic server behavior and actions. This component on its own does nothing. Look TnxServerComp for a functioning descendant.

See Also

[Events](#)
[nxBaseServerComp](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class

25.1.86.1 Constructors

25.1.86.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.Create Constructor

[TnxBaseServer Class](#)

Pascal

```
public constructor Create(
  Sender: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxBaseServer Class](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Constructors > Create Constructor

25.1.86.2 Destructors

25.1.86.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServer.Destroy Destructor

TnxBaseServer Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseServer Class](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Destructors > Destroy Destructor

25.1.86.3 Methods

25.1.86.3.1 ActivateAll Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServer.ActivateAll Method

[TnxBaseServer Class](#) | Related Topics

Pascal

```
public procedure ActivateAll; virtual;
```

Remarks

Calling this method activates all components that are descendent on the server engine.

Related Topics

[DeActivateAll](#), [StartStoppedModules](#), [StopStartedModules](#).

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > ActivateAll Method

25.1.86.3.2 Authenticate Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServer.Authenticate Method

[TnxBaseServer Class](#) | Related Topics

Pascal

```
public function Authenticate(
  aTries: Integer = 3
): Boolean;
```

Remarks

This function initiates a logon. If OnShowLogin is assigned it is called and the username/password validated. If it's ok the function returns true otherwise it tries again for up to aTries times. If the last try fails too it returns false.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > Authenticate Method

25.1.86.3.3 ClearAllStats Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.ClearAllStats Method

[TnxBaseServer Class](#)

Pascal

```
public procedure ClearAllStats;
```

Remarks

As the name tells, this methods clears the stats of all dependent components.

See Also

[TnxBaseServer Class](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > ClearAllStats Method

25.1.86.3.4 DeActivateAll Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.DeActivateAll Method

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
public procedure DeActivateAll; virtual;
```

Remarks

Call this method to de-activate all components that are descendent on the server engine.

Related Topics

[ActivateAll](#), [StartStoppedModules](#), [StopStartedModules](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > DeActivateAll Method

25.1.86.3.5 LoadSettings Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.LoadSettings Method

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
public procedure LoadSettings(
  aFileName: string
```

```
); virtual;
```

Remarks

Calling this, stops all activated components, loads the settings from the specified file and re-starts the components.

Related Topics

[LoadSettings](#), [SaveSettings](#), [AutoSaveConfig](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > LoadSettings Method

25.1.86.3.6 SaveSettings Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.SaveSettings Method

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
public procedure SaveSettings(
  aFileName: string
); virtual;
```

Remarks

This methods saves the settings of ALL dependent components based on TnxComponent to the specified.

Related Topics

[LoadSettings](#), [SaveSettings](#), [AutoSaveConfig](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > SaveSettings Method

25.1.86.3.7 StartStoppedModules Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.StartStoppedModules Method

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
public procedure StartStoppedModules; virtual;
```

Remarks

This method re-starts the components that where de-activated with StopStartedModules

Related Topics

[ActivateAll](#), [DeActivateAll](#), [StartStoppedModules](#), [StopStartedModules](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > StartStoppedModules Method

25.1.86.3.8 StopStartedModules Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServer.StopStartedModules Method

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
public procedure StopStartedModules; virtual;
```

Remarks

Stops all activated modules and flags them to be re-startable by StartStoppedModules.

Related Topics

[ActivateAll](#), [DeActivateAll](#), [StartStoppedModules](#), [StopStartedModules](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Methods > StopStartedModules Method

25.1.86.4 Events

25.1.86.4.1 OnError Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServer.OnError Property

[TnxBaseServer Class](#)

Pascal

```
published property OnError: TNotifyErrorEvent;
```

Remarks

This event is triggered if an error occurs in the internal processing.

See Also

[TnxBaseServer Class](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Events > OnError Property

25.1.86.4.2 OnShowLogin Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServer.OnShowLogin Property

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
published property OnShowLogin: TnxLoginCallback;
```

Remarks

Assign an event handler here if you want to have your own input or check. If the default platform implementations unit is linked into the project, you don't need a handler for showing the login dialog.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#), [Authenticate](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Events > OnShowLogin Property

25.1.86.5 Properties

25.1.86.5.1 AutoSaveConfig Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.AutoSaveConfig Property

[TnxBaseServer Class](#) | Related Topics

Pascal

```
published property AutoSaveConfig: Boolean;
```

Remarks

Set AutoSaveConfig to false if the configuration should not be saved on exit and re-loaded on startup. The default is true.

Related Topics

[LoadSettings](#), [SaveSettings](#), [AutoSaveConfig](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > AutoSaveConfig Property

25.1.86.5.2 IsServerActive Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.IsServerActive Property

[TnxBaseServer Class](#)

Pascal

```
published property IsServerActive: Boolean;
```

Remarks

This is IsServerActive, a member of class TnxBaseServer.

See Also

[TnxBaseServer Class](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > IsServerActive Property

25.1.86.5.3 LoginTries Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseServer.LoginTries Property**[TnxBaseServer Class](#) | [Related Topics](#)**Pascal**

```
published property LoginTries: Integer;
```

Remarks

Set the number of login tries possible. Only valid if SecureServer is active.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#), [Authenticate](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxBaseServer Class](#) > [Properties](#) > [LoginTries Property](#)

25.1.86.5.4 MaxUserCount Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseServer.MaxUserCount Property**[TnxBaseServer Class](#)**Pascal**

```
published property MaxUserCount: integer;
```

Remarks

The number of users that are allowed to be logged on to the server at any time.

See Also

[TnxBaseServer Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxBaseServer Class](#) > [Properties](#) > [MaxUserCount Property](#)

25.1.86.5.5 OnLoginError Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseServer.OnLoginError Property**[TnxBaseServer Class](#) | [Related Topics](#)**Pascal**

```
published property OnLoginError: TNotifyEvent;
```

Remarks

This is triggered every time the Security Monitor tries to validate an invalid username/password configuration.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#), [Authenticate](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > OnLoginError Property

25.1.86.5.6 Password Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.Password Property

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
published property Password: string;
```

Remarks

This is the password for logging onto the GUI of the server. Only valid if SecureServer is active.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#), [Authenticate](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > Password Property

25.1.86.5.7 SecureServer Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServer.SecureServer Property

[TnxBaseServer Class](#) | [Related Topics](#)

Pascal

```
public property SecureServer: Boolean;
```

Remarks

Set this property to true if the server should enforce logins for the GUI and sessions that want to connect. Please make sure to have at least one Admin user in the user list of the SecurityMonitor otherwise the user gets locked out of the server GUI.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#) , [Authenticate](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > SecureServer Property

25.1.86.5.8 SecurityMonitor Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseServer.SecurityMonitor Property**[TnxBaseServer Class](#) | Related Topics**Pascal**

```
public property SecurityMonitor: TnxAbstractSecurityMonitor;
```

Remarks

This property returns the attached Security Monitor, if any. It does this by iterating through the dependents list of the server engine and returning the FIRST found instance derived from TnxAbstractSecurityMonitor.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#), [Authenticate](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > SecurityMonitor Property

25.1.86.5.9 ServerConfiguration Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseServer.ServerConfiguration Property**[TnxBaseServer Class](#)**Pascal**

```
published property ServerConfiguration: TnxBaseServerConfiguration;
```

Remarks

This is ServerConfiguration, a member of class TnxBaseServer.

See Also

[TnxBaseServer Class](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > ServerConfiguration Property

25.1.86.5.10 Username Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseServer.Username Property**[TnxBaseServer Class](#) | Related Topics**Pascal**

```
published property Username: string;
```

Remarks

This is the username for logging onto the GUI of the server. Only valid if SecureServer is active.

Related Topics

[SecureServer](#), [SecurityMonitor](#), [UserName](#), [Password](#), [LoginTries](#), [OnShowLogin](#), [OnLoginError](#), [Authenticate](#)

See Also

[TnxBaseServer Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseServer Class > Properties > Username Property

25.1.87 TnxBaseServerConfiguration Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServerConfiguration Class

[nxllComponent](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxBaseServerConfiguration = class;
```

File

[nxllComponent](#)

Remarks

An abstract prototype for a Server configuration class. An actual implementation must override ALL functions.

See Also

[nxllComponent](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxBaseServerConfiguration Class

25.1.87.1 Constructors

25.1.87.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServerConfiguration.Create Constructor

[TnxBaseServerConfiguration Class](#)

Pascal

```
public constructor Create(
  aFileName: String
); virtual;
```

Remarks

This is Create, a member of class TnxBaseServerConfiguration.

See Also

[TnxBaseServerConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerConfiguration Class > Constructors > Create Constructor

25.1.87.2 Methods

25.1.87.2.1 EnsureComponentConfiguration Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerConfiguration.EnsureComponentConfiguration Method

TnxBaseServerConfiguration Class

Pascal

```
public procedure EnsureComponentConfiguration(
    aComponent: string;
    aSettings: TnxBaseSettings
); virtual; abstract;
```

Remarks

The implementation must make sure that all settings in the aSettings are available and of the right type; this method is NOT to write any values but the settings definitions only. A UI can then be built by reading that information.

See Also

[TnxBaseServerConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerConfiguration Class > Methods > [EnsureComponentConfiguration Method](#)

25.1.87.2.2 GetComponentConfiguration Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerConfiguration.GetComponentConfiguration Method

TnxBaseServerConfiguration Class

Pascal

```
public function GetComponentConfiguration(
    aComponent: string
): TnxBaseComponentConfiguration; virtual; abstract;
```

Remarks

Has to return a TnxBaseComponentConfiguration instance for the given Component.

See Also

[TnxBaseServerConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerConfiguration Class > Methods > [GetComponentConfiguration Method](#)

25.1.88 TnxBaseServerEngine Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerEngine Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseServerEngine = class(TnxStateComponent);
```

File

[nxsdServerEngine](#)

Remarks

This is the base for all Server Engine implementations in NexusDB. All actual server engines are descendent from this class. Please note that a server engine does not do a lot of stuff on its own. It's a mainly a center point that holds together all engines and their instances of server objects.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class

25.1.88.1 Constructors

25.1.88.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseServerEngine.Create Constructor

TnxBaseServerEngine Class

Pascal

```
public constructor Create(
    aOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Constructors > Create Constructor

25.1.88.2 Destructors

25.1.88.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseServerEngine.Destroy Destructor

TnxBaseServerEngine Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Destructors > Destroy Destructor

25.1.88.3 Methods

25.1.88.3.1 ClearStats Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerEngine.ClearStats Method

TnxBaseServerEngine Class

Pascal

```
public procedure ClearStats; override;
```

Remarks

This function resets all statistical values.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Methods > ClearStats Method

25.1.88.3.2 GetStatsCaptions Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerEngine.GetStatsCaptions Method

TnxBaseServerEngine Class

Pascal

```
public procedure GetStatsCaptions(
  const aList: TStrings
); override;
```

Remarks

This function returns the captions for the statistics page on the server gui.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Methods > GetStatsCaptions Method

25.1.88.3.3 GetStatsValues Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerEngine.GetStatsValues Method

TnxBaseServerEngine Class

Pascal

```
public procedure GetStatsValues(
  const aList: TStrings
); override;
```

Remarks

This function returns the values for the statistics page on the server gui.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Methods > GetStatsValues Method

25.1.88.3.4 SessionOpen Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerEngine.SessionOpen Method

TnxBaseServerEngine Class

Pascal

```
public function SessionOpen(
    out aSession: TnxAbstractSession;
    const aUserName: string;
    const aPassword: string;
    const aConnectedFrom: string;
    aTimeout: TnxWord32
): TnxResult; virtual;
```

Parameters

Parameters	Description
aSession	the returned instance of a TnxAbstractSession descendant
aUserName	the user name used to authenticate the session if security is enabled. A security monitor needs to be attached for this to work.
aPassword	the password used to authenticate the session if security is enabled
aTimeout	the timeout for the newly created session in msec

Remarks

This function opens creates a new session.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Methods > SessionOpen Method

25.1.88.3.5 SessionOpenEx Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerEngine.SessionOpenEx Method

TnxBaseServerEngine Class

Pascal

```
public function SessionOpenEx(
    out aSession: TnxAbstractSession;
    const aUserName: string;
    const aPassword: string;
    const aConnectedFrom: string;
    aTimeout: TnxWord32;
    aClientVersion: Integer
): TnxResult; virtual; abstract;
```

Parameters

Parameters	Description
aSession	the returned instance of a TnxAbstractSession descendant
aUserName	the user name used to authenticate the session if security is enabled. A security monitor needs to be attached for this to work.
aPassword	the password used to authenticate the session if security is enabled
aTimeout	the timeout for the newly created session in msec
aClientVersion	the version of the NexusDB core on the client side

Remarks

This function opens creates a new session. It additionally passes the Client Version. This function can be used to support older clients on a newer version on the server.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Methods > [SessionOpenEx Method](#)

25.1.88.4 Properties

25.1.88.4.1 CursorCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServerEngine.CursorCount Property

[TnxBaseServerEngine Class](#)

Pascal

```
public property CursorCount: Integer;
```

Remarks

CursorCount returns the number of open cursors in this server engine

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Properties > CursorCount Property

25.1.88.4.2 DatabaseCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseServerEngine.DatabaseCount Property

[TnxBaseServerEngine Class](#)

Pascal

```
public property DatabaseCount: Integer;
```

Remarks

DatabaseCount returns the number of open databases in this server engine.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Properties > DatabaseCount Property

25.1.88.4.3 ServerGuid Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseServerEngine.ServerGuid Property

TnxBaseServerEngine Class

Pascal

```
public property ServerGuid: TnxGuid;
```

Remarks

Returns a unique ID for this server.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Properties > ServerGuid Property

25.1.88.4.4 SessionCount Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseServerEngine.SessionCount Property

TnxBaseServerEngine Class

Pascal

```
public property SessionCount: Integer;
```

Remarks

SessionCount returns the number of open session in this server engine

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Properties > SessionCount Property

25.1.88.4.5 StatementCount Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseServerEngine.StatementCount Property

TnxBaseServerEngine Class

Pascal

```
public property StatementCount: Integer;
```

Remarks

StatementCount returns the number of open (SQL) statements in this server engine.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Properties > StatementCount Property

25.1.88.4.6 TransContextCount Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseServerEngine.TransContextCount Property

TnxBaseServerEngine Class

Pascal

```
public property TransContextCount: Integer;
```

Remarks

TransContextCount returns the number of transaction contexts in this server engine.

See Also

[TnxBaseServerEngine Class](#)

You are here: Symbol Reference > Classes > TnxBaseServerEngine Class > Properties > TransContextCount Property

25.1.89 TnxBaseSession Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSession Class

[Events](#) | [nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseSession = class(TnxDataAccessStateComponent, IUnknown, InxSessionRequests);
```

File

[nxdb](#)

Remarks

The mother of all sessions. It implements a component that is used the same way as Borland's TSession and adds some features unique to NexusDB.

See Also

[Events](#)
[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class

25.1.89.1 Constructors

25.1.89.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSession.Create Constructor

TnxBaseSession Class

Pascal

```
public constructor Create(
  AOwner: TComponent
```

```
); override;
```

Remarks

constructor.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Constructors > Create Constructor

25.1.89.2 Destructors

25.1.89.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.Destroy Destructor

[TnxBaseSession Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Destructors > Destroy Destructor

25.1.89.3 Methods

25.1.89.3.1 AddAlias Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.AddAlias Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public procedure AddAlias(
    const aName: string;
    const aPath: string
);
```

Parameters

Parameters	Description
aName	Aliasname
aPath	Path the alias should point to

Remarks

Adds a new alias. If an error occurs, it triggers an exception.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > AddAlias Method

25.1.89.3.2 AddAliasEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.AddAliasEx Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public function AddAliasEx(
    const aName: string;
    const aPath: string
): TnxResult;
```

Parameters

Parameters	Description
aName	Aliasname
aPath	Path the alias should point to

Remarks

Adds a new alias and returns the error code.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > AddAliasEx Method

25.1.89.3.3 CancelProcessing Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.CancelProcessing Method

[TnxBaseSession Class](#)

Pascal

```
public procedure CancelProcessing;
```

Remarks

This procedure can be called from another thread to terminate any currently ongoing processing.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > CancelProcessing Method

25.1.89.3.4 CloseInactiveFolders Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseSession.CloseInactiveFolders Method**[TnxBaseSession Class](#)**Pascal**

```
public procedure CloseInactiveFolders;
```

Remarks

This method closes all folders of the session that were kept open by the server caching mechanism.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > CloseInactiveFolders Method

25.1.89.3.5 CloseInactiveTables Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseSession.CloseInactiveTables Method**[TnxBaseSession Class](#)**Pascal**

```
public procedure CloseInactiveTables;
```

Remarks

This method closes all tables of the session that were kept open by the server caching mechanism.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > CloseInactiveTables Method

25.1.89.3.6 DefaultDatabase Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseSession.DefaultDatabase Method**[TnxBaseSession Class](#)**Pascal**

```
public function DefaultDatabase: TnxDatabase;
```

Remarks

Returns the default database for this session or nil if non exist.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > DefaultDatabase Method

25.1.89.3.7 DefaultSession Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.DefaultSession Method**[TnxBaseSession Class](#)**Pascal**

```
public class function DefaultSession: TnxBaseSession;
```

Remarks

Returns the default session for the current thread or nil if non exist.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > DefaultSession Method

25.1.89.3.8 DefaultTransContext Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.DefaultTransContext Method**[TnxBaseSession Class](#)**Pascal**

```
public function DefaultTransContext: TnxTransContext;
```

Remarks

Returns the default TransContext for this session or nil if non exist.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > DefaultTransContext Method

25.1.89.3.9 DeleteAlias Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.DeleteAlias Method**[TnxBaseSession Class](#) | [Related Topics](#)**Pascal**

```
public procedure DeleteAlias(
  const aName: string
);
```

Parameters**Parameters**

aName

Description

Aliasname

Remarks

Deletes the given alias. Triggers an exception if it fails.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > DeleteAlias Method

25.1.89.3.10 DeleteAliasEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.DeleteAliasEx Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public function DeleteAliasEx(
    const aName: string
): TnxResult;
```

Parameters

Parameters	Description
aName	Aliasname

Remarks

Deletes the given alias and returns the error code.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > DeleteAliasEx Method

25.1.89.3.11 FindDatabase Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.FindDatabase Method

[TnxBaseSession Class](#)

Pascal

```
public function FindDatabase(
    const aAliasName: string
): TnxDatabase; virtual;
```

Remarks

Returns the implicit database for the given Alias. May return nil. The returned database may be closed.

do not free the returned database

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > FindDatabase Method

25.1.89.3.12 GetAliasNames Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSession.GetAliasNames Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public procedure GetAliasNames(
  aList: TStrings
);
```

Parameters

Parameters

aList

Description

will hold the names after the call; must no be nil;
always clears the list before adding the names

Remarks

Fills the given list with all alias names.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#),
[GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > GetAliasNames Method

25.1.89.3.13 GetAliasNamesEx Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSession.GetAliasNamesEx Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public function GetAliasNamesEx(
  aList: TStrings;
  aEmptyList: Boolean
): TnxResult;
```

Parameters

Parameters

aList

Description

will hold the names after the call; must no be nil

aEmptyList

clears the list before adding, if true

Remarks

Fills the given list with all alias names and returns the error code.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#),
[GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > GetAliasNamesEx Method

25.1.89.3.14 GetAliasPath Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.GetAliasPath Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public procedure GetAliasPath(
  const aName: string;
  out aPath: string
);
```

Parameters

Parameters	Description
aName	AliasName
aPath	will hold the path after the method returns

Remarks

Returns the path an alias points to. Remember that this is always a server relative path! The method triggers an exception if it fails.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > GetAliasPath Method

25.1.89.3.15 GetAliasPathEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.GetAliasPathEx Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public function GetAliasPathEx(
  const aName: string;
  out aPath: string
): TnxResult;
```

Parameters

Parameters	Description
aName	AliasName
aPath	will hold the path after the method returns

Remarks

Returns the path an alias points to. Remember that this is always a server relative path! It returns the error code as function result.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > GetAliasPathEx Method

25.1.89.3.16 GetChangedTables Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.GetChangedTables Method

[TnxBaseSession Class](#)

Pascal

```
public procedure GetChangedTables(
  const aAliasName: string;
  aList: TStrings
);
```

Parameters

Parameters	Description
aAliasName	the name of the alias
aList	the list to hold the names; must not be nil

Remarks

Returns the changed tables of an alias in the given list.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > GetChangedTables Method

25.1.89.3.17 GetRegisteredClassList Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.GetRegisteredClassList Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public procedure GetRegisteredClassList(
  const aClassListType: TnxClassListType;
  aList: TStrings
);
```

Parameters

Parameters	Description
aClassListType	the type of engines or descriptors list to return

aList

an instance of a `TStrings` descendant; will hold the list on return

Remarks

Returns a list of engines or descriptors of a particular type from the server.

Related Topics

[TnxClassListType](#)

See Also

[TnxBaseSession Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > [TnxBaseSession Class](#) > Methods > [GetRegisteredClassList Method](#)

25.1.89.3.18 GetStoredProcNames Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.GetStoredProcNames Method

[TnxBaseSession Class](#)

Pascal

```
public procedure GetStoredProcNames (
  const aAliasName: string;
  aList: TStrings
);
```

Parameters

Parameters

aList

Description

will hold the names after the method returns;
must not be nil

Remarks

`GetStoredProcNames` returns a list of all stored procedures of a database/alias.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > [TnxBaseSession Class](#) > Methods > [GetStoredProcNames Method](#)

25.1.89.3.19 GetStoredProcParams Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.GetStoredProcParams Method

[TnxBaseSession Class](#)

Pascal

```
public procedure GetStoredProcParams (
  const aAliasName: string;
  const aStoredProcName: string;
  out aParams: TnxSqlParamList
);
```

Remarks

Returns the parameter definitions for a named stored procedure of the given Alias.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > GetStoredProcParams Method

25.1.89.3.20 GetTableNames Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.GetTableNames Method

[TnxBaseSession Class](#)

Pascal

```
public procedure GetTableNames(
  const aAliasName: string;
  aList: TStrings
);
```

Parameters

Parameters	Description
aAliasName	the name of the alias
aList	the list to hold the names; must not be nil

Remarks

Returns the tables of an alias in the given list.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > GetTableNames Method

25.1.89.3.21 IsAlias Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.IsAlias Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public function IsAlias(
  const aName: string
): Boolean;
```

Parameters

Parameters	Description
aName	AliasName

Remarks

Returns true if an alias with the given name exists, otherwise false.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > IsAlias Method

25.1.89.3.22 IsConnected Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.IsConnected Method

[TnxBaseSession Class](#)

Pascal

```
public function IsConnected: Boolean;
```

Remarks

This function returns true if the session is active/connected.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > IsConnected Method

25.1.89.3.23 ModifyAlias Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.ModifyAlias Method

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public procedure ModifyAlias(
  const aName: string;
  const aNewName: string;
  const aNewPath: string
);
```

Parameters

Parameters	Description
aName	Aliasname
aNewName	new Name of the Alias
aNewPath	new Path of the Alias

Remarks

This function allows to change the name and/or path of an alias.

It triggers an exception if it fails.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > ModifyAlias Method

25.1.89.3.24 ModifyAliasEx Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.ModifyAliasEx Method**[TnxBaseSession Class](#) | [Related Topics](#)**Pascal**

```
public function ModifyAliasEx(
    const aName: string;
    const aNewName: string;
    const aNewPath: string
): TnxResult;
```

Parameters

Parameters	Description
aName	Aliasname
aNewName	new Name of the Alias
aNewPath	new Path of the Alias

Remarks

This function allows to change the name and/or path of an alias and returns the error code.

Related Topics

[AddAliasEx](#), [DeleteAlias](#), [DeleteAliasEx](#), [ModifyAlias](#), [ModifyAliasEx](#), [GetAliasNames](#), [GetAliasNamesEx](#), [GetAliasPath](#), [GetAliasPathEx](#), [IsAlias](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > ModifyAliasEx Method

25.1.89.3.25 OpenDatabase Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.OpenDatabase Method**[TnxBaseSession Class](#)**Pascal**

```
public function OpenDatabase(
    const aAliasName: string
): TnxDatabase; virtual;
```

Remarks

Returns the implicit database for the given Alias, creates it if required. The returned database is guaranteed to be open.

do not free the returned database**See Also**

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > OpenDatabase Method

25.1.89.3.26 PasswordAdd Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseSession.PasswordAdd Method**[TnxBaseSession Class](#)**Pascal**

```
public procedure PasswordAdd(
    const aPassword: string
);
```

Remarks

Adds a password to the session password list.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > PasswordAdd Method

25.1.89.3.27 PasswordRemove Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseSession.PasswordRemove Method**[TnxBaseSession Class](#)**Pascal**

```
public procedure PasswordRemove(
    const aPassword: string
);
```

Remarks

Remove a password to the session password list.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > PasswordRemove Method

25.1.89.3.28 PasswordRemoveAll Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseSession.PasswordRemoveAll Method**[TnxBaseSession Class](#)**Pascal**

```
public procedure PasswordRemoveAll;
```

Remarks

Clear the session password list.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > PasswordRemoveAll Method

25.1.89.3.29 TableExists Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.TableExists Method**[TnxBaseSession Class](#)**Pascal**

```
public function TableExists(
    const aAliasName: string;
    const aTableName: string;
    const aPassword: string
): Boolean;
```

Parameters

Parameters	Description
aAliasName	the name of the alias
aTableName	the name of the table

Remarks

Checks if a table exist in a specific alias.

See Also[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Methods > TableExists Method

25.1.89.4 Events

25.1.89.4.1 OnLogin Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.OnLogin Property**[TnxBaseSession Class | Related Topics](#)**Pascal**

```
published property OnLogin: TnxLoginEvent;
```

Parameters

Parameters	Description
aSender	the session calling the event
aUserName	the username supplied to the Session, change if wanted/needed
aPassword	the password supplied to the Session, change if wanted/needed
aResult	return true for successful logon, false if you want the logon to fail

Remarks

This event is triggered whenever the session is set to active and the connected server engine is in secure mode, thus requesting users to logon.

Related Topics[UserName](#), [BeepOnLoginError](#), [Password](#), [OnLogin](#), [PasswordRetries](#)**See Also**

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Events > OnLogin Property

25.1.89.5 Properties

25.1.89.5.1 AbstractSession Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSession.AbstractSession Property

[TnxBaseSession Class](#)

Pascal

```
public property AbstractSession: TnxAbstractSession;
```

Remarks

This property gives direct access to the internal session state.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > AbstractSession Property

25.1.89.5.2 BeepOnLoginError Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSession.BeepOnLoginError Property

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
published property BeepOnLoginError: Boolean;
```

Remarks

If true, a Beep is triggered whenever a failed Login occurs.

Related Topics

[UserName](#), [BeepOnLoginError](#), [Password](#), [OnLogin](#), [PasswordRetries](#)

See Also

[TnxBaseSession Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > BeepOnLoginError Property

25.1.89.5.3 CurrentUserName Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSession.CurrentUserName Property

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
published property CurrentUserName: string;
```

Remarks

The user name used for logging in this session. It is only assigned while the session is active.

Related Topics

[UserName](#), [BeepOnLoginError](#), [Password](#), [OnLogin](#), [PasswordRetries](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > CurrentUserName Property

25.1.89.5.4 DatabaseCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.DatabaseCount Property

[TnxBaseSession Class](#) | Related Topics

Pascal

```
public property DatabaseCount: Integer;
```

Remarks

Returns the number of tnxDatabase instances assigned to the session.

Related Topics

[Databases](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > DatabaseCount Property

25.1.89.5.5 Databases Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSession.Databases Property

[TnxBaseSession Class](#) | Related Topics

Pascal

```
public property Databases [aInx: Integer]: TnxDatabase;
```

Remarks

Returns the instance of tnxDatabase with index aInx in the list of assigned databases.

Related Topics

[DatabaseCount](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > Databases Property

25.1.89.5.6 Default Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.Default Property**[TnxBaseSession Class](#)**Pascal**

```
published property Default: Boolean;
```

Remarks

If true this session is the default session for the current thread.

See Also[TnxBaseSession Class](#)*You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > Default Property*

25.1.89.5.7 Password Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.Password Property**[TnxBaseSession Class | Related Topics](#)**Pascal**

```
published property Password: string;
```

Remarks

The password for the session. It is only used when the connected Server Engine is in secure mode.

Related Topics[UserName](#), [BeepOnLoginError](#), [Password](#), [OnLogin](#), [PasswordRetries](#)**See Also**[TnxBaseSession Class](#)[Related Topics](#)*You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > Password Property*

25.1.89.5.8 PasswordRetries Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSession.PasswordRetries Property**[TnxBaseSession Class | Related Topics](#)**Pascal**

```
published property PasswordRetries: Integer;
```

Remarks

The number of retries for logon before the session returns a DBIERR_INVALIDUSRPASS error is triggered.

Related Topics[UserName](#), [BeepOnLoginError](#), [Password](#), [OnLogin](#), [PasswordRetries](#)**See Also**[TnxBaseSession Class](#)[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > PasswordRetries Property

25.1.89.5.9 ServerEngine Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSession.ServerEngine Property

TnxBaseSession Class

Pascal

```
public property ServerEngine: TnxBaseServerEngine;
```

Remarks

Returns the Server Engine the session is connected to. Note, that this is read only in the base class. The Set method might be implemented in descendants.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > ServerEngine Property

25.1.89.5.10 ServerVersion Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSession.ServerVersion Property

TnxBaseSession Class

Pascal

```
public property ServerVersion: Integer;
```

Remarks

Returns the Version of the server engine.

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > ServerVersion Property

25.1.89.5.11 Timeout Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSession.Timeout Property

TnxBaseSession Class

Pascal

```
published property Timeout: Integer;
```

Remarks

Sets the Timeout for the Session commands (in ms).

See Also

[TnxBaseSession Class](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > Timeout Property

25.1.89.5.12 TransContextCount Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSession.TransContextCount Property

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public property TransContextCount: Integer;
```

Remarks

Returns the number of TnxTransContext instances assigned to the session.

Related Topics

[TransContexts](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > TransContextCount Property

25.1.89.5.13 TransContexts Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSession.TransContexts Property

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
public property TransContexts [aInx: Integer]: TnxTransContext;
```

Remarks

Returns the instance of TnxTransContext with index aInx in the list of assigned TransContexts.

Related Topics

[TransContextCount](#)

See Also

[TnxBaseSession Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > TransContexts Property

25.1.89.5.14 UserName Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSession.UserName Property

[TnxBaseSession Class](#) | [Related Topics](#)

Pascal

```
published property UserName: string;
```

Remarks

The user name for the session. It is only used when the connected Server Engine is in secure mode.

Related Topics

UserName, BeepOnLoginError, Password, OnLogin, PasswordRetries

See Also

[TnxBaseSession Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSession Class > Properties > UserName

Property

25.1.90 TnxBaseSessionPool Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSessionPool Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseSessionPool = class(TnxDataAccessStateComponent, IUnknown, InxSessionRequests);
```

File

[nxdb](#)

Remarks

This is class TnxBaseSessionPool.

See Also

[nxdb](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class

25.1.90.1 Constructors

25.1.90.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSessionPool.Create Constructor

[TnxBaseSessionPool Class](#)

Pascal

```
public constructor Create(
  AOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxBaseSessionPool Class](#)

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class > Constructors > Create Constructor

25.1.90.2 Destructors

25.1.90.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSessionPool.Destroy Destructor

TnxBaseSessionPool Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseSessionPool Class](#)

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class > Destructors > Destroy Destructor

25.1.90.3 Methods

25.1.90.3.1 AcquireSession Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSessionPool.AcquireSession Method

TnxBaseSessionPool Class

Pascal

```
public function AcquireSession: TnxPooledSession;
```

Remarks

This is AcquireSession, a member of class TnxBaseSessionPool.

See Also

[TnxBaseSessionPool Class](#)

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class > Methods > AcquireSession Method

25.1.90.4 Properties

25.1.90.4.1 Password Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSessionPool.Password Property

[TnxBaseSessionPool Class](#) | Related Topics

Pascal

```
published property Password: string;
```

Remarks

The password for the session pool. It is only used when the connected Server Engine is in secure mode.

Related Topics

[UserName](#), [Password](#)

See Also

[TnxBaseSessionPool Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class > Properties > Password Property

25.1.90.4.2 ServerEngine Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSessionPool.ServerEngine Property

[TnxBaseSessionPool Class](#)

Pascal

```
public property ServerEngine: TnxBaseServerEngine;
```

Remarks

Returns the Server Engine the session pool is connected to. Note, that this is read only in the base class. The Set method might be implemented in descendants.

See Also

[TnxBaseSessionPool Class](#)

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class > Properties > ServerEngine Property

25.1.90.4.3 Timeout Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSessionPool.Timeout Property

[TnxBaseSessionPool Class](#)

Pascal

```
published property Timeout: Integer;
```

Remarks

Sets the Timeout for the Session pool commands (in ms).

See Also

[TnxBaseSessionPool Class](#)

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class > Properties > Timeout Property

25.1.90.4.4 UserName Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSessionPool.UserName Property

[TnxBaseSessionPool Class](#) | [Related Topics](#)

Pascal

```
published property UserName: string;
```

Remarks

The user name for the session pool. It is only used when the connected Server Engine is in secure mode.

Related Topics

UserName, Password

See Also

TnxBaseSessionPool Class
Related Topics

You are here: Symbol Reference > Classes > TnxBaseSessionPool Class > Properties > UserName Property

25.1.91 TnxBaseSetting Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSetting Class

[nxllComponent](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxBaseSetting = class;
```

File

[nxllComponent](#)

Remarks

The base class used for defining storeable component settings. Each setting can be fully defined by this class.

See Also

[nxllComponent](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class

25.1.91.1 Properties

25.1.91.1.1 DefaultValue Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseSetting.DefaultValue Property

[TnxBaseSetting Class](#)

Pascal

```
public property DefaultValue: Variant;
```

Remarks

Default Value.

See Also

[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > DefaultValue Property

25.1.91.1.2 EnforceValues Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSetting.EnforceValues Property**[TnxBaseSetting Class](#)**Pascal**

```
public property EnforceValues: Boolean;
```

Remarks

If true then only values of Listvalues are possible.

See Also[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > EnforceValues Property

25.1.91.1.3 Hint Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSetting.Hint Property**[TnxBaseSetting Class](#)**Pascal**

```
public property Hint: string;
```

Remarks

A hint to be shown in a UI.

See Also[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > Hint Property

25.1.91.1.4 Max Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSetting.Max Property**[TnxBaseSetting Class](#)**Pascal**

```
public property Max: Variant;
```

Remarks

Max value.

See Also[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > Max Property

25.1.91.1.5 Min Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseSetting.Min Property**

[TnxBaseSetting Class](#)**Pascal**

```
public property Min: Variant;
```

Remarks

Min value.

See Also

[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > Min Property

25.1.91.1.6 Name Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSetting.Name Property[TnxBaseSetting Class](#)**Pascal**

```
public property Name: string;
```

Remarks

Name of the Setting.

See Also

[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > Name Property

25.1.91.1.7 PropertyName Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSetting.PropertyName Property[TnxBaseSetting Class](#)**Pascal**

```
public property PropertyName: string;
```

Remarks

Name of the implementation Property

See Also

[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > PropertyName Property

25.1.91.1.8 SettingType Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBaseSetting.SettingType Property[TnxBaseSetting Class](#)**Pascal**

```
public property SettingType: TnxSettingType;
```

Remarks

Data type of the setting.

See Also

[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > SettingType Property

25.1.91.1.9 ValueList Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSetting.ValueList Property

[TnxBaseSetting Class](#)

Pascal

```
public property ValueList: string;
```

Remarks

List of possible values (usually Comma separated).

See Also

[TnxBaseSetting Class](#)

You are here: Symbol Reference > Classes > TnxBaseSetting Class > Properties > ValueList Property

25.1.92 TnxBaseSettings Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSettings Class

[nxIIComponent](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseSettings = class;
```

File

[nxIIComponent](#)

Remarks

A collection of TBaseSetting instances. Each nxComponent that wants to save/load settings, must return this list.

See Also

[nxIIComponent](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseSettings Class

25.1.92.1 Methods

25.1.92.1.1 AddSetting Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSettings.AddSetting Method

TnxBaseSettings Class

```
Pascal
public function AddSetting(
    aSetting: TnxBaseSetting
) : TnxBaseSetting; virtual; abstract;
```

Remarks

Adds a setting to the collection.

See Also

[TnxBaseSettings Class](#)

You are here: Symbol Reference > Classes > TnxBaseSettings Class > Methods > AddSetting Method

25.1.92.2 Properties

25.1.92.2.1 Count Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSettings.Count Property[TnxBaseSettings Class](#)

```
Pascal
public property Count: integer;
```

Remarks

Returns the number of settings in the collection.

See Also

[TnxBaseSettings Class](#)

You are here: Symbol Reference > Classes > TnxBaseSettings Class > Properties > Count Property

25.1.92.2.2 Setting Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseSettings.Setting Property[TnxBaseSettings Class](#)

```
Pascal
public property Setting [anIndex: integer]: TnxBaseSetting;
```

Remarks

Returns the setting at position anIndex in the collection.

See Also

[TnxBaseSettings Class](#)

You are here: Symbol Reference > Classes > TnxBaseSettings Class > Properties > Setting Property

25.1.93 TnxBaseStreamDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseStreamDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseStreamDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

This is the base class for Stream Descriptors. A Stream descriptor specifies the sub-engine that generates the Streams.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseStreamDescriptor Class

25.1.93.1 Methods

25.1.93.1.1 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseStreamDescriptor.IsEqual Method

[TnxBaseStreamDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxBaseStreamDescriptor.

See Also

[TnxBaseStreamDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseStreamDescriptor Class > Methods > IsEqual Method

25.1.93.1.2 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseStreamDescriptor.LoadFromReader Method

[TnxBaseStreamDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseStreamDescriptor.

See Also

[TnxBaseStreamDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseStreamDescriptor Class > Methods > LoadFromReader Method

25.1.93.1.3 SaveToWriter Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseStreamDescriptor.SaveToWriter Method

TnxBaseStreamDescriptor Class

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseStreamDescriptor.

See Also

[TnxBaseStreamDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseStreamDescriptor Class > Methods > [SaveToWriter Method](#)

25.1.93.2 Properties

25.1.93.2.1 HeaderSection Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseStreamDescriptor.HeaderSection Property

TnxBaseStreamDescriptor Class

Pascal

```
public property HeaderSection: string;
```

Remarks

Returns the header of the descriptor as string.

See Also

[TnxBaseStreamDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseStreamDescriptor Class > Properties > [HeaderSection Property](#)

25.1.93.2.2 StreamEngine Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseStreamDescriptor.StreamEngine Property

TnxBaseStreamDescriptor Class

Pascal

```
public property StreamEngine: string;
```

Remarks

The name of the Stream Engine.

See Also

TnxBaseStreamDescriptor Class

You are here: Symbol Reference > Classes > TnxBaseStreamDescriptor Class > Properties > StreamEngine Property

25.1.94 TnxBaseTableDescriptor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTableDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseTableDescriptor = class(TnxLocalizedDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

A table descriptor holds all information about a table as well as an instance of TnxFieldsDescriptor, TnxCustomDescsDescriptor, TnxTablesDescriptor, TnxBaseRecordDescriptor, TnxMainIndicesDescriptor, TnxBaseAutoIncDescriptor to fully define a table and its attributes.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class

25.1.94.1 Constructors

25.1.94.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTableDescriptor.Create Constructor

TnxBaseTableDescriptor Class

Pascal

```
public constructor Create(
  aParent: TPersistent
); virtual;
```

Remarks

constructor.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Constructors > Create Constructor

25.1.94.2 Destructors

25.1.94.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTableDescriptor.Destroy Destructor

[TnxBaseTableDescriptor Class](#)

```
Pascal
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Destructors > Destroy Destructor

25.1.94.3 Methods

25.1.94.3.1 AddAutoIncDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.AddAutoIncDescriptor Method

[TnxBaseTableDescriptor Class](#) | Related Topics

```
Pascal
public function AddAutoIncDescriptor(
  aAutoIncClass: TnxBaseAutoIncDescriptorClass = nil
): TnxBaseAutoIncDescriptor;
```

Remarks

This function sets the AutoInc Descriptor class to the given aAutoIncClass, creates and instance, adds it to the data dictionary and returns the instance.

Related Topics

[AddAutoIncDescriptor](#), [RemoveAutoIncDescriptor](#), [AutoIncDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
Related Topics

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > [AddAutoIncDescriptor Method](#)

25.1.94.3.2 AddBlobDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.AddBlobDescriptor Method

[TnxBaseTableDescriptor Class](#) | Related Topics

```
Pascal
public function AddBlobDescriptor(
  aBlobClass: TnxBaseBlobDescriptorClass = nil
): TnxBaseBlobDescriptor;
```

Remarks

This function sets the Blob Descriptor class to the given aBlobClass, creates and instance, adds it to the data dictionary and returns the instance.

Related Topics

[AddBlobDescriptor](#), [RemoveBlobDescriptor](#), [BlobDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > AddBlobDescriptor Method

25.1.94.3.3 AddIndicesDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.AddIndicesDescriptor Method

[TnxBaseTableDescriptor Class](#) | [Related Topics](#)

Pascal

```
public function AddIndicesDescriptor(
    aIndicesClass: TnxMainIndicesDescriptorClass = nil
): TnxMainIndicesDescriptor;
```

Remarks

This function sets the Indices Descriptor class to the given aIndicesClass, creates and instance, adds it to the data dictionary and returns the instance.

Related Topics

[AddIndicesDescriptor](#), [RemoveIndicesDescriptor](#), [IndicesDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > AddIndicesDescriptor Method

25.1.94.3.4 AddRecordDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.AddRecordDescriptor Method

[TnxBaseTableDescriptor Class](#) | [Related Topics](#)

Pascal

```
public function AddRecordDescriptor(
    aRecordClass: TnxBaseRecordDescriptorClass = nil
): TnxBaseRecordDescriptor;
```

Remarks

This function sets the Record Descriptor class to the given aRecordClass, creates and instance, adds it to the data dictionary and returns the instance.

Related Topics

[AddRecordDescriptor](#), [RemoveRecordDescriptor](#), [RecordDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > AddRecordDescriptor Method

25.1.94.3.5 AssignOnlyFields Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTableDescriptor.AssignOnlyFields Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public procedure AssignOnlyFields(
    aSource: TnxBaseTableDescriptor
); virtual;
```

Remarks

This is AssignOnlyFields, a member of class TnxBaseTableDescriptor.

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > AssignOnlyFields Method

25.1.94.3.6 CheckValid Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTableDescriptor.CheckValid Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public procedure CheckValid; override;
```

Remarks

Checks if the Data Dictionary is valid. Raises an exception if not.

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > CheckValid Method

25.1.94.3.7 Clear Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTableDescriptor.Clear Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public procedure Clear;
```

Remarks

This function clears the Data Dictionary.

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > Clear Method

25.1.94.3.8 EnsureIndicesDescriptor Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTableDescriptor.EnsureIndicesDescriptor Method**[TnxBaseTableDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public function EnsureIndicesDescriptor: TnxMainIndicesDescriptor;
```

Remarks

This function ensures there is an indices descriptor.

Related Topics

[AddIndicesDescriptor](#), [RemoveIndicesDescriptor](#), [IndicesDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > EnsureIndicesDescriptor Method

25.1.94.3.9 FindRelatedDescriptorOfType Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTableDescriptor.FindRelatedDescriptorOfType Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public function FindRelatedDescriptorOfType(
  aType: TnxDictionaryItemClass;
  out aDescriptor;
  aFrom: TnxDictionaryItem = nil
) : Boolean; override;
```

Remarks

This is FindRelatedDescriptorOfType, a member of class TnxBaseTableDescriptor.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > FindRelatedDescriptorOfType Method

25.1.94.3.10 GetChild Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTableDescriptor.GetChild Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public function GetChild(
  const aChildName: string
) : TnxBaseTableDescriptor;
```

Remarks

Returns a named sub table descriptor.

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > GetChild Method

25.1.94.3.11 GetConstraintByName Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTableDescriptor.GetConstraintByName Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public function GetConstraintByName(
  const aName: string
): TnxDictionaryItem;
```

Remarks

This is GetConstraintByName, a member of class TnxBaseTableDescriptor.

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > GetConstraintByName Method

25.1.94.3.12 GetConstraints Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTableDescriptor.GetConstraints Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public procedure GetConstraints(
  aStrings: TStrings
);
```

Remarks

This is GetConstraints, a member of class TnxBaseTableDescriptor.

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > GetConstraints Method

25.1.94.3.13 GetFieldFromName Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTableDescriptor.GetFieldFromName Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public function GetFieldFromName(
  const aFieldName: string
): Integer;
```

Remarks

Return the field number for a given field name, or -1 if not found.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > GetFieldFromName Method

25.1.94.3.14 GetIndexFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.GetIndexFromName Method

[TnxBaseTableDescriptor Class](#)

Pascal

```
public function GetIndexFromName(
    const aIndexName: string
): Integer;
```

Remarks

Return the index number for a given index name, or -1 if not found.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > GetIndexFromName Method

25.1.94.3.15 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.AreEqual Method

[TnxBaseTableDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Parameters

Parameters	Description
aDataDictionary	the Data Dictionary to compare the current one to.

Remarks

Compares the current Data Dictionary to another. Returns true if they are equal.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > IsEqual Method

25.1.94.3.16 IsIndexDescValid Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseTableDescriptor.IsIndexDescValid Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public function IsIndexDescValid(
    aIndexDesc: TnxIndexDescriptor
): Boolean; virtual;
```

Parameters**Parameters**

aIndexDesc

Description

The Index Descriptor to check.

Remarks

Returns true if an Index Descriptor validates ok.

See Also[TnxBaseTableDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > IsIndexDescValid Method

25.1.94.3.17 LoadFromReader Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseTableDescriptor.LoadFromReader Method**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxBaseTableDescriptor.

See Also[TnxBaseTableDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > LoadFromReader Method

25.1.94.3.18 RemoveAutoIncDescriptor Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseTableDescriptor.RemoveAutoIncDescriptor Method**[TnxBaseTableDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public procedure RemoveAutoIncDescriptor;
```

Remarks

This function frees the Instance of the associated AutoInc Descriptor.

Related Topics

AddAutoIncDescriptor, RemoveAutoIncDescriptor, AutoIncDescriptor

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > [RemoveAutoIncDescriptor Method](#)

25.1.94.3.19 RemoveBlobDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.RemoveBlobDescriptor Method

[TnxBaseTableDescriptor Class](#) | [Related Topics](#)

Pascal

```
public procedure RemoveBlobDescriptor;
```

Remarks

This function frees the Instance of the associated Blob Descriptor.

Related Topics

[AddBlobDescriptor](#), [RemoveBlobDescriptor](#), [BlobDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > [RemoveBlobDescriptor Method](#)

25.1.94.3.20 RemoveIndicesDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.RemoveIndicesDescriptor Method

[TnxBaseTableDescriptor Class](#) | [Related Topics](#)

Pascal

```
public procedure RemoveIndicesDescriptor;
```

Remarks

This function frees the Instance of the associated Indices Descriptor.

Related Topics

[AddIndicesDescriptor](#), [RemoveIndicesDescriptor](#), [IndicesDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > [RemoveIndicesDescriptor Method](#)

25.1.94.3.21 RemoveRecordDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.RemoveRecordDescriptor Method

[TnxBaseTableDescriptor Class](#) | [Related Topics](#)

Pascal

```
public procedure RemoveRecordDescriptor;
```

Remarks

This function frees the Instance of the associated Record Descriptor.

Related Topics

[AddRecordDescriptor](#), [RemoveRecordDescriptor](#), [RecordDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > RemoveRecordDescriptor Method

25.1.94.3.22 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.SaveToWriter Method

[TnxBaseTableDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxBaseTableDescriptor.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Methods > SaveToWriter Method

25.1.94.4 Properties

25.1.94.4.1 AutoIncDescriptor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.AutoIncDescriptor Property

[TnxBaseTableDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property AutoIncDescriptor: TnxBaseAutoIncDescriptor;
```

Remarks

This property returns the associated AutoInc Descriptor.

Related Topics

[AddAutoIncDescriptor](#), [RemoveAutoIncDescriptor](#), [AutoIncDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > AutoIncDescriptor Property

25.1.94.4.2 BlobDescriptor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.BlobDescriptor Property

[TnxBaseTableDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property BlobDescriptor: TnxBaseBlobDescriptor;
```

Remarks

This property returns the associated Blob Descriptor.

Related Topics

[AddBlobDescriptor](#), [RemoveBlobDescriptor](#), [BlobDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > BlobDescriptor Property

25.1.94.4.3 BookmarkSize Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.BookmarkSize Property

[TnxBaseTableDescriptor Class](#)

Pascal

```
public property BookmarkSize [aInx : Integer]: TnxWord32;
```

Remarks

The length of a bookmark for the given index.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > BookmarkSize Property

25.1.94.4.4 CustomDescsDescriptor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.CustomDescsDescriptor Property

[TnxBaseTableDescriptor Class](#)

Pascal

```
public property CustomDescsDescriptor: TnxCustomDescsDescriptor;
```

Remarks

Holds all CustomDescriptor items of the table

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > [CustomDescsDescriptor Property](#)

25.1.94.4.5 FieldsDescriptor Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTableDescriptor.FieldsDescriptor Property**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public property FieldsDescriptor: TnxFieldsDescriptor;
```

Remarks

Holds all FieldDescriptor items of the table

See Also[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > [FieldsDescriptor Property](#)

25.1.94.4.6 IndicesDescriptor Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTableDescriptor.IndicesDescriptor Property**[TnxBaseTableDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public property IndicesDescriptor: TnxMainIndicesDescriptor;
```

Remarks

This property returns the associated Indices Descriptor.

Related Topics
[AddIndicesDescriptor](#), [RemoveIndicesDescriptor](#), [IndicesDescriptor](#)
See Also[TnxBaseTableDescriptor Class](#)[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > [IndicesDescriptor Property](#)

25.1.94.4.7 KeyLen Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTableDescriptor.KeyLen Property**[TnxBaseTableDescriptor Class](#)**Pascal**

```
public property KeyLen [aInx : Integer]: TnxWord32;
```

Remarks

The length of the key for the given index.

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > KeyLen Property

25.1.94.4.8 RecordDescriptor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.RecordDescriptor Property

[TnxBaseTableDescriptor Class](#) | Related Topics

Pascal

```
public property RecordDescriptor: TnxBaseRecordDescriptor;
```

Remarks

This property returns the associated Record Descriptor.

Related Topics

[AddRecordDescriptor](#), [RemoveRecordDescriptor](#), [RecordDescriptor](#)

See Also

[TnxBaseTableDescriptor Class](#)
Related Topics

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > RecordDescriptor Property

25.1.94.4.9 TablesDescriptor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptor.TablesDescriptor Property

[TnxBaseTableDescriptor Class](#)

Pascal

```
public property TablesDescriptor: TnxTablesDescriptor;
```

Remarks

Holds all Sub TableDescriptor items of the table

See Also

[TnxBaseTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxBaseTableDescriptor Class > Properties > TablesDescriptor Property

25.1.95 TnxBaseTransport Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport Class

[Events](#) | [nxIITransport](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxBaseTransport = class(TnxStateComponent);
```

File[nxIITransport](#)**Remarks**

The base class for all transports.

See Also
[Events](#)
[nxIITransport](#)
[Members](#)
[Methods](#)
[Properties](#)
You are here: Symbol Reference > Classes > TnxBaseTransport Class**25.1.95.1 Constructors**

25.1.95.1.1 Create Constructor

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.Create Constructor**

TnxBaseTransport Class

Pascal

```
public constructor Create(
    aOwner: TComponent
); override;
```

Remarks

constructor.

See Also[TnxBaseTransport Class](#)*You are here:* Symbol Reference > Classes > TnxBaseTransport Class > Constructors > Create Constructor**25.1.95.2 Destructors**

25.1.95.2.1 Destroy Destructor

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.Destroy Destructor**

TnxBaseTransport Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also[TnxBaseTransport Class](#)*You are here:* Symbol Reference > Classes > TnxBaseTransport Class > Destructors > Destroy Destructor

25.1.95.3 Methods

25.1.95.3.1 BeginBroadcast Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.BeginBroadcast Method

TnxBaseTransport Class

Pascal

```
public function BeginBroadcast(
    aHandler: InxBroadcastReplyHandler;
    aInterval: Integer
): InxActiveBroadcast; virtual; abstract;
```

Remarks

This function returns an InxActiveBroadcast and will start sending out server discovery broadcast every aInterval seconds. The broadcasting can be suspended or stopped using the returned InxActiveBroadcast interface. To get an event triggered for each reply received from a server, pass in an InxBroadcastReplyHandler. The ReplyReceived method of the passed in aHandler will be called.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > BeginBroadcast Method

25.1.95.3.2 Broadcast Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.Broadcast Method

TnxBaseTransport Class

Pascal

```
public procedure Broadcast(
    aMsgID: TnxMsgID;
    aThreadPriority: TnxThreadPriority;
    arequestData: Pointer;
    arequestDataLen: TnxWord32;
    aTimeOut: Integer
); virtual; abstract;
```

Parameters

Parameters	Description
aMsgID	the id of the message
arequestData	the data of the broadcast request
arequestDataLen	the length of arequestData
aTimeOut	timeout in msec

Remarks

This function initiates a broadcast for servers.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > Broadcast Method

25.1.95.3.3 btDoCallBack Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseTransport.btDoCallBack Method**[TnxBaseTransport Class](#)**Pascal**

```
public procedure btDoCallBack(
    aReplyCallback: TnxReplyCallback;
    aMsgID: TnxMsgID;
    aErrorCode: TnxResult;
    aReplyData: Pointer;
    aReplyDataLen: Integer;
    aReplyCookie: Integer;
    aReplyCompressedSize: integer = 0
);
```

Parameters

Parameters	Description
aReplyCallback	the pointer to the callback function that should be called
aMsgID	the message id
aErrorCode	the error code to be transferred
aReplyData	the data returned for the request
aReplyDataLen	the length of aReplyData
aReplyCookie	additional info to be returned
aReplyCompressedSize	the compressed data size

Remarks

This function executes a callback function with the given parameters.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > btDoCallBack Method

25.1.95.3.4 ClearStats Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxBaseTransport.ClearStats Method**[TnxBaseTransport Class](#)**Pascal**

```
public procedure ClearStats; override;
```

Remarks

This function resets all statistics for the transport.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > ClearStats Method

25.1.95.3.5 ConnectionCount Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.ConnectionCount Method**[TnxBaseTransport Class](#)**Pascal**

```
public function ConnectionCount: Integer; virtual; abstract;
```

Remarks

This function returns the number of physical connections.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > ConnectionCount Method

25.1.95.3.6 CurrentTransport Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.CurrentTransport Method**[TnxBaseTransport Class](#)**Pascal**

```
public class function CurrentTransport: TnxBaseTransport;
```

Remarks

This function returns the currently active transport.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > CurrentTransport Method

25.1.95.3.7 EstablishConnection Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.EstablishConnection Method**[TnxBaseTransport Class](#)**Pascal**

```
public function EstablishConnection(
    aTransportID: TnxTransportID;
    const aUserName: string;
    const aPassword: string;
    aTimeOut: Integer;
    aClientVersion: Integer;
    out aServerVersion: Integer;
    out aSessionID: TnxSessionID
): TnxResult; virtual; abstract;
```

Parameters**Parameters**

aTransportID

Description

the id of the transport

aUserName	the user name used to connect
aPassword	the password used to connect
aTimeOut	timeout in msec
aSessionID	the session id for the connection

Remarks

This function establishes a connection and returns the session id. This function sends a request to the server and then waits for the reply.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > EstablishConnection Method

25.1.95.3.8 GetBoundAddresses Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.GetBoundAddresses Method

[TnxBaseTransport Class](#)

Pascal

```
public procedure GetBoundAddresses (
  aList: TStringList
); virtual;
```

Remarks

This function returns a list of addresses for the transport.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > GetBoundAddresses Method

25.1.95.3.9 GetConfigSettings Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.GetConfigSettings Method

[TnxBaseTransport Class](#)

Pascal

```
public procedure GetConfigSettings (
  aSettings: TnxBaseSettings
); override;
```

Remarks

Returns the Configuration settings for the UI

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > GetConfigSettings Method

25.1.95.3.10 GetName Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.GetName Method**[TnxBaseTransport Class](#)**Pascal**

```
public function GetName: string; virtual; abstract;
```

Remarks

This function returns the name of the transport.

See Also[TnxBaseTransport Class](#)*You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > GetName Method*

25.1.95.3.11 GetServerNames Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.GetServerNames Method**[TnxBaseTransport Class](#)**Pascal**

```
public procedure GetServerNames(
  aList: TStrings;
  aTimeOut: TnxWord32
); virtual;
```

Remarks

This function returns the addresses of all servers found for the current transport by using broadcast. aTimeOut specifies the timeout in msec for the broadcast.

See Also[TnxBaseTransport Class](#)*You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > GetServerNames Method*

25.1.95.3.12 GetSessionCallback Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.GetSessionCallback Method**[TnxBaseTransport Class](#)**Pascal**

```
public class function GetSessionCallback: InxSessionCallback;
```

Remarks

This function returns an interface that can be used to send callbacks to the session that initiated the currently active call. May return nil.

See Also[TnxBaseTransport Class](#)*You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > GetSessionCallback Method*

25.1.95.3.13 GetStatsCaptions Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.GetStatsCaptions Method

TnxBaseTransport Class

Pascal

```
public procedure GetStatsCaptions(
  const aList: TStrings
); override;
```

Remarks

This function returns the captions for the statistics items.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > GetStatsCaptions Method

25.1.95.3.14 GetStatsValues Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.GetStatsValues Method

TnxBaseTransport Class

Pascal

```
public procedure GetStatsValues(
  const aList: TStrings
); override;
```

Remarks

This function returns the values for the statistics items.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > GetStatsValues Method

25.1.95.3.15 IsConnected Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.IsConnected Method

TnxBaseTransport Class

Pascal

```
public function IsConnected: Boolean; virtual; abstract;
```

Remarks

IsConnected returns true if the current transport has an open connection.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > IsConnected Method

25.1.95.3.16 LoadConfig Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.LoadConfig Method**[TnxBaseTransport Class](#)**Pascal**

```
public procedure LoadConfig(
    aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

Loads the transport settings.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > LoadConfig Method

25.1.95.3.17 LoadSettingsFromStream Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.LoadSettingsFromStream Method**[TnxBaseTransport Class](#)**Pascal**

```
public procedure LoadSettingsFromStream(
    aReader: TnxReader
); override;
```

Remarks

This function loads the Transport settings from a stream using aReader.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > LoadSettingsFromStream Method

25.1.95.3.18 Post Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.Post Method**[TnxBaseTransport Class](#)**Pascal**

```
public procedure Post(
    aTransportID: TnxTransportID;
    aSessionID: TnxSessionID;
    aThreadPriority: TnxThreadPriority;
    aMsgID: TnxMsgID;
    arequestData: Pointer;
    arequestDataLen: TnxWord32;
    aTimeOut: Integer
); virtual; abstract;
```

Parameters**Parameters****Description**

aTransportID	generally the current transport id
aSessionID	the session id
aMsgID	the message id
arequestData	the data of the broadcast request
arequestDataLen	the length of arequestData
aTimeOut	timeout in msec

Remarks

This functions posts a request and does NOT wait for the answer.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > Post Method

25.1.95.3.19 ProtocolName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.ProtocolName Method

[TnxBaseTransport Class](#)

Pascal

```
public class function ProtocolName: string; virtual;
```

Remarks

This function returns the name of the transport.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > ProtocolName Method

25.1.95.3.20 Reply Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.Reply Method

[TnxBaseTransport Class](#)

Pascal

```
public class procedure Reply(
    aMsgID: TnxMsgID;
    aErrorCode: TnxResult;
    aReplyData: Pointer;
    aReplyDataLen: TnxWord32
);
```

Parameters

Parameters	Description
aMsgID	the message id to reply to
aErrorCode	the error code to be returned
aReplyData	the data to be returned
aReplyDataLen	the length of aReplyData

Remarks

This function is called to reply to a request

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > Reply Method

25.1.95.3.21 Request Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.Request Method

[TnxBaseTransport Class](#)

Pascal

```
public procedure Request(
  aTransportID: TnxTransportID;
  aSessionID: TnxSessionID;
  aThreadPriority: TnxThreadPriority;
  aMsgID: TnxMsgID;
  aTimeOut: Integer;
  arequestData: Pointer;
  arequestDataLen: TnxWord32;
  aReplyCallback: TnxReplyCallback;
  aReplyCookie: Integer
); virtual; abstract;
```

Parameters

Parameters	Description
aTransportID	generally the current transport id
aSessionID	the session id
aMsgID	the message id
aTimeOut	timeout in msec
arequestData	the data of the broadcast request
arequestDataLen	the length of arequestData
aReplyCallback	the callback function to be called once the reply arrived.
aReplyCookie	additional information received.

Remarks

This functions sends a request and waits for the answer.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > Request Method

25.1.95.3.22 ResetMsgCount Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.ResetMsgCount Method

[TnxBaseTransport Class](#)

Pascal

```
public procedure ResetMsgCount; virtual;
```

Remarks

This function sets the MsgCount stats to 0.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > ResetMsgCount Method

25.1.95.3.23 SaveConfig Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.SaveConfig Method

[TnxBaseTransport Class](#)

Pascal

```
public procedure SaveConfig(
    aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

Saves the transport settings.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > SaveConfig Method

25.1.95.3.24 SaveSettingsToStream Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.SaveSettingsToStream Method

[TnxBaseTransport Class](#)

Pascal

```
public procedure SaveSettingsToStream(
    aWriter: TnxWriter
); override;
```

Remarks

This function saves the Transport settings to a stream using aWriter.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > SaveSettingsToStream Method

25.1.95.3.25 Sleep Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.Sleep Method

TnxBaseTransport Class

```
Pascal
public function Sleep(
  const aTimeOut: Integer
): Boolean; virtual;
```

Remarks

This function forces the transport to sleep for aTimeOut msec.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > Sleep Method

25.1.95.3.26 TerminateConnection Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.TerminateConnection Method

TnxBaseTransport Class

```
Pascal
public procedure TerminateConnection(
  aTransportID: TnxTransportID;
  aSessionID: TnxSessionID;
  aTimeOut: Integer
); virtual; abstract;
```

Parameters

Parameters	Description
aTransportID	the transport
aSessionID	the session to be disconnected
aTimeOut	timeout in msec

Remarks

This function closes a connection.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > TerminateConnection Method

25.1.95.3.27 Write Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.Write Method

TnxBaseTransport Class

```
Pascal
public procedure Write(
  aConnection: Pointer;
  aTransportID: TnxTransportID;
  aSessionID: TnxSessionID;
  aThreadPriority: TnxThreadPriority;
  aMsgID: TnxMsgID;
```

```

    aMsgType: Byte;
    aErrorCode: TnxResult;
    aCompressType: Byte;
    aUncompressedSize: TnxWord32;
    aBuffer: Pointer;
    aBufferSize: TnxWord32
); virtual;

```

Parameters

Parameters	Description
aConnection	the connection to be used
aTransportID	generally the current transport id
aMsgID	the message id
aMsgType	the type of the message
aErrorCode	the error code to be transferred
aCompressType	the type of compression used for the data
aUncompressedSize	the uncompressed size of the message
aBuffer	the data
aBufferSize	the length of aBuffer in bytes
aRequestID	the request id

Remarks

This functions transfers the given information to the other connection point by wrapping it into a TnxDataMessage and sending it over an open connection.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Methods > Write Method

25.1.95.4 Events

25.1.95.4.1 OnAddSession Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.OnAddSession Property

[TnxBaseTransport Class](#)

Pascal

```
published property OnAddSession: TnxAddSessionEvent;
```

Remarks

OnAddSession is called every time a session connects to the transport if it is in listening mode.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Events > OnAddSession Property

25.1.95.4.2 OnChooseServer Property*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxBaseTransport.OnChooseServer Property**[TnxBaseTransport Class](#)**Pascal**

```
published property OnChooseServer: TnxChooseServerEvent;
```

Remarks

OnChooseServer is called in sending mode, if more than one server was found for broadcast

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Events > OnChooseServer Property

25.1.95.4.3 OnConnectionLost Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.OnConnectionLost Property**[TnxBaseTransport Class](#)**Pascal**

```
published property OnConnectionLost: TnxConnectionLostEvent;
```

Remarks

OnConnectionLost is called if the transport detects a drop of a connection.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Events > OnConnectionLost Property

25.1.95.4.4 OnFindServers Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.OnFindServers Property**[TnxBaseTransport Class](#)**Pascal**

```
published property OnFindServers: TnxFindServersEvent;
```

Remarks

OnFindServers is called before and after broadcasting for servers.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Events > OnFindServers Property

25.1.95.4.5 OnRemoveSession Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.OnRemoveSession Property

TnxBaseTransport Class

Pascal

```
published property OnRemoveSession: TnxRemoveSessionEvent;
```

Remarks

`OnAddSession` is called every time a session disconnects from a transport if it is in listening mode.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Events > OnRemoveSession Property

25.1.95.5 Properties

25.1.95.5.1 ActualBytesReceived Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.ActualBytesReceived Property

TnxBaseTransport Class

Pascal

```
public property ActualBytesReceived: TnxInt64;
```

Remarks

The number of bytes received since the last clearing.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ActualBytesReceived Property

25.1.95.5.2 ActualBytesSent Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseTransport.ActualBytesSent Property

TnxBaseTransport Class

Pascal

```
public property ActualBytesSent: TnxInt64;
```

Remarks

The number of bytes sent since the last clearing.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ActualBytesSent Property

25.1.95.5.3 CommandHandler Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.CommandHandler Property**[TnxBaseTransport Class](#)**Pascal**

```
published property CommandHandler: TnxBaseCommandHandler;
```

Remarks

This property points to the connected Command Handler.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > CommandHandler Property

25.1.95.5.4 CompressTime Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.CompressTime Property**[TnxBaseTransport Class](#)**Pascal**

```
public property CompressTime: TnxInt64;
```

Remarks

The total time used for compressing data.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > CompressTime Property

25.1.95.5.5 EventLogOptions Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.EventLogOptions Property**[TnxBaseTransport Class](#)**Pascal**

```
published property EventLogOptions: TnxTransportLogOptions;
```

Remarks

The log options for the event.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > EventLogOptions Property

25.1.95.5.6 KeepStats Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.KeepStats Property**[TnxBaseTransport Class](#)**Pascal**

```
published property KeepStats: Boolean;
```

Remarks

Should the stats be updated or not. On high volume servers on the edge of capacity switch this off

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > KeepStats Property

25.1.95.5.7 MaxBytesPerSecond Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.MaxBytesPerSecond Property**[TnxBaseTransport Class](#)**Pascal**

```
published property MaxBytesPerSecond: TnxWord32;
```

Remarks

This is a maximum bytes per second setting for the transport. It is experimental and helps to test the transport for slow connections.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > MaxBytesPerSecond Property

25.1.95.5.8 MessagesReceived Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.MessagesReceived Property**[TnxBaseTransport Class](#)**Pascal**

```
public property MessagesReceived: TnxInt64;
```

Remarks

The number of Messages received since the last clearing.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > MessagesReceived Property

25.1.95.5.9 MessagesSent Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.MessagesSent Property**[TnxBaseTransport Class](#)**Pascal**

```
public property MessagesSent: TnxInt64;
```

Remarks

The number of Messages sent since the last clearing.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > MessagesSent Property

25.1.95.5.10 Mode Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.Mode Property**[TnxBaseTransport Class](#)**Pascal**

```
published property Mode: TnxTransportMode;
```

Remarks

The mode of the transport.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > Mode Property

25.1.95.5.11 MsgCount Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.MsgCount Property**[TnxBaseTransport Class](#)**Pascal**

```
public property MsgCount: Integer;
```

Remarks

The total number of messages transferred.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > MsgCount Property

25.1.95.5.12 PingTime Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxBaseTransport.PingTime Property**

[TnxBaseTransport Class](#)**Pascal**

```
published property PingTime: TnxWord32;
```

Remarks

This is a minimum turnaround time setting in msec for the transport. It is experimental and helps to test the transport for slow connections.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > PingTime Property

[25.1.95.5.13 RespondToBroadcasts Property](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTransport.RespondToBroadcasts Property**[TnxBaseTransport Class](#)**Pascal**

```
published property RespondToBroadcasts: Boolean;
```

Remarks

If the server is in listening it responds to broadcast requests if this property is set to true.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > RespondToBroadcasts Property

[25.1.95.5.14 ServerGUID Property](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTransport.ServerGUID Property**[TnxBaseTransport Class](#)**Pascal**

```
public property ServerGUID: TnxGuid;
```

Remarks

This is ServerGUID, a member of class TnxBaseTransport.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ServerGUID Property

[25.1.95.5.15 ServerName Property](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTransport.ServerName Property**[TnxBaseTransport Class](#)

Pascal

```
public property ServerName: string;
```

Remarks

The Server Name. If in listening mode it's the name of the clients have to connect to. In sending mode it's the name the transport gets connected to.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ServerName Property

25.1.95.5.16 ServerNameDesigntime Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.ServerNameDesigntime Property

[TnxBaseTransport Class](#)

Pascal

```
published property ServerNameDesigntime: string;
```

Remarks

This is the Server Name property at design time.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ServerNameDesigntime Property

25.1.95.5.17 ServerNameRuntime Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.ServerNameRuntime Property

[TnxBaseTransport Class](#)

Pascal

```
published property ServerNameRuntime: string;
```

Remarks

This is the Server Name property at runtime.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ServerNameRuntime Property

25.1.95.5.18 ThreadPriorityMax Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.ThreadPriorityMax Property

[TnxBaseTransport Class](#)

Pascal

```
published property ThreadPriorityMax: TnxThreadPriority;
```

Remarks

Maximum possible Threadpriority

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ThreadPriorityMax Property

25.1.95.5.19 ThreadPriorityMin Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.ThreadPriorityMin Property

[TnxBaseTransport Class](#)

Pascal

```
published property ThreadPriorityMin: TnxThreadPriority;
```

Remarks

Minimal possible Threadpriority

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > ThreadPriorityMin Property

25.1.95.5.20 Timeout Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.Timeout Property

[TnxBaseTransport Class](#)

Pascal

```
published property Timeout: TnxWord32;
```

Remarks

The timeout of the transport in msec.

See Also

[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > Timeout Property

25.1.95.5.21 UncompressedBytesReceived Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransport.UncompressedBytesReceived Property

[TnxBaseTransport Class](#)

Pascal

```
public property UncompressedBytesReceived: TnxInt64;
```

Remarks

The number of bytes received after uncompression since the last clearing.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > [UncompressedBytesReceived Property](#)

25.1.95.5.22 UncompressedBytesSent Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTransport.UncompressedBytesSent Property**[TnxBaseTransport Class](#)**Pascal**

```
public property UncompressedBytesSent: TnxInt64;
```

Remarks

The number of bytes sent after uncompression since the last clearing.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > [UncompressedBytesSent Property](#)

25.1.95.5.23 UncompressTime Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTransport.UncompressTime Property**[TnxBaseTransport Class](#)**Pascal**

```
public property UncompressTime: TnxInt64;
```

Remarks

The total time used for uncompressing data.

See Also[TnxBaseTransport Class](#)

You are here: Symbol Reference > Classes > TnxBaseTransport Class > Properties > [UncompressTime Property](#)

25.1.96 TnxBaseTransportWrapper Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBaseTransportWrapper Class**[nxIITransport](#)**Pascal**

```
public TnxBaseTransportWrapper = class(TnxBaseTransport);
```

File[nxIITransport](#)**Remarks**

This is the base class for all wrapper transports. A wrapper transports wraps another transport, e.g. for security extensions.

See Also

[nxlITransport](#)

You are here: Symbol Reference > Classes > TnxBaseTransportWrapper Class

25.1.97 TnxBaseUpdateObject Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseUpdateObject Class

[nxdb](#)

Pascal

```
public TnxBaseUpdateObject = class(TnxComponent);
```

File

[nxdb](#)

Remarks

Base Update object for cached datasets.

See Also

[nxdb](#)

You are here: Symbol Reference > Classes > TnxBaseUpdateObject Class

25.1.98 TnxBatchAppendBlobStream Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBatchAppendBlobStream Class

[nxdb](#) | [Members](#) | [Methods](#) | [Related Topics](#)

Pascal

```
public TnxBatchAppendBlobStream = class(TnxBaseBlobStream);
```

File

[nxdb](#)

Remarks

This class is used if the dataset is in batch append mode.

Never create instances directly! Always use `TnxDataSet.CreateBlobStream`, which will create the right type of blob stream based on the state of the dataset.

The blob stream should be freed again as soon as possible. It must be freed before calling Post or moving to another record.

Related Topics

[TnxDataSet.CreateBlobStream](#) [TnxBaseBlobStream](#) [TnxBlobStream](#) [TnxBlockModeBlobStream](#)
[TnxMemoBlobStream](#)

See Also

[nxdb](#)
[Members](#)
[Methods](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBatchAppendBlobStream Class

25.1.98.1 Destructors

25.1.98.1.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBatchAppendBlobStream.Destroy Destructor

TnxBatchAppendBlobStream Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxBatchAppendBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBatchAppendBlobStream Class > Destructors > Destroy Destructor

25.1.98.2 Methods

25.1.98.2.1 Read Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBatchAppendBlobStream.Read Method

TnxBatchAppendBlobStream Class

Pascal

```
public function Read(
  var aBuffer;
  aCount: Integer
): Integer; override;
```

Parameters**Parameters****Description**

aBuffer	variable to store the read bytes
---------	----------------------------------

aCount	number of bytes to read
--------	-------------------------

Remarks

With this method you can read a number of bytes of the blob data.

See Also

[TnxBatchAppendBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBatchAppendBlobStream Class > Methods > Read Method

25.1.98.2.2 Seek Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxBatchAppendBlobStream.Seek Method

TnxBatchAppendBlobStream Class

```
Pascal
public function Seek(
    aOffset: Integer;
    aOrigin: Word
): Integer; override;
```

Parameters

Parameters	Description
aOffset	the offset in bytes to move the pointer
aOrigin	the origin of the aOffset

Remarks

Seek sets the current position in the blob stream.

See Also

[TnxBatchAppendBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBatchAppendBlobStream Class > Methods > Seek Method

25.1.98.2.3 Truncate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBatchAppendBlobStream.Truncate Method

[TnxBatchAppendBlobStream Class](#)

```
Pascal
public procedure Truncate; override;
```

Remarks

This method truncates the blob data at the current position.

See Also

[TnxBatchAppendBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBatchAppendBlobStream Class > Methods > Truncate Method

25.1.98.2.4 Write Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBatchAppendBlobStream.Write Method

[TnxBatchAppendBlobStream Class](#)

```
Pascal
public function Write(
    const aBuffer;
    aCount: Integer
): Integer; override;
```

Parameters

Parameters	Description
aBuffer	data to be written
aCount	number of bytes to write

Remarks

Use Write to write a number of bytes to the blob at the current position. Take care to not accidentally overwrite data by not positioning with Seek first.

See Also

[TnxBatchAppendBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBatchAppendBlobStream Class > Methods > Write Method

25.1.99 TnxBlobStream Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlobStream Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#) | [Related Topics](#)

Pascal

```
public TnxBlobStream = class(TnxBaseBlobStream);
```

File

nxdb

Remarks

This class is used to access a blob directly. All calls are directly passed through to the server engine.

Never create instances directly! Always use TnxDataSet.CreateBlobStream, which will create the right type of blob stream based on the state of the dataset.

The blob stream should be freed again as soon as possible. It must be freed before calling Post or moving to another record.

Related Topics

[TnxDataSet.CreateBlobStream](#) [TnxBaseBlobStream](#) [TnxBlockModeBlobStream](#)
[TnxBatchAppendBlobStream](#) [TnxMemoBlobStream](#)

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class

25.1.99.1 Destructors

25.1.99.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlobStream.Destroy Destructor

[TnxBlobStream Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also[TnxBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class > Destructors > Destroy Destructor

25.1.99.2 Methods

25.1.99.2.1 Read Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBlobStream.Read Method**[TnxBlobStream Class](#)

Pascal
<pre>public function Read(var aBuffer; aCount: Integer): Integer; override;</pre>

Parameters

Parameters	Description
aBuffer	variable to store the read bytes
aCount	number of bytes to read

Remarks

With this method you can read a number of bytes of the blob data.

See Also[TnxBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class > Methods > Read Method

25.1.99.2.2 Seek

25.1.99.2.2.1 Seek Method (Int64, TSeekOrigin)

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxBlobStream.Seek Method (Int64, TSeekOrigin)**[TnxBlobStream Class](#)

Pascal
<pre>public function Seek(const aOffset: Int64; aOrigin: TSeekOrigin): Int64; override;</pre>

Parameters

Parameters	Description
aOffset	the offset in bytes to move the pointer
aOrigin	the origin of the aOffset

Remarks

Seek sets the current position in the blob stream.

See Also

[TnxBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class > Methods > Seek > Seek Method (Int64, TSeekOrigin)

25.1.99.2.2.2 Seek Method (Integer, Word)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlobStream.Seek Method (Integer, Word)

[TnxBlobStream Class](#)

Pascal

```
public function Seek(
  aOffset: Integer;
  aOrigin: Word
): Integer; override;
```

Parameters

Parameters	Description
aOffset	the offset in bytes to move the pointer
aOrigin	the origin of the aOffset

Remarks

Seek sets the current position in the blob stream.

See Also

[TnxBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class > Methods > Seek > Seek Method (Integer, Word)

25.1.99.2.3 Truncate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlobStream.Truncate Method

[TnxBlobStream Class](#)

Pascal

```
public procedure Truncate; override;
```

Remarks

This method truncates the blob data at the current position.

See Also

[TnxBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class > Methods > Truncate Method

25.1.99.2.4 Write Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlobStream.Write Method

[TnxBlobStream Class](#)

Pascal

```
public function Write(
    const aBuffer;
    aCount: Integer
): Integer; override;
```

Parameters

Parameters	Description
aBuffer	data to be written
aCount	number of bytes to write

Remarks

Use Write to write a number of bytes to the blob at the current position. Take care to not accidentally overwrite data by not positioning with Seek first.

See Also

[TnxBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class > Methods > Write Method

25.1.99.3 Properties

25.1.99.3.1 ChunkSize Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlobStream.ChunkSize Property

[TnxBlobStream Class](#)

Pascal

```
public property ChunkSize: Integer;
```

Remarks

Chunksize specifies the number of byte that are read/written in one file access.

See Also

[TnxBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlobStream Class > Properties > ChunkSize Property

25.1.10(TnxBlockModeBlobStream Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlockModeBlobStream Class

[nxdb](#) | [Members](#) | [Methods](#) | [Related Topics](#)

Pascal

```
public TnxBlockModeBlobStream = class(TnxBaseBlobStream);
```

File

[nxdb](#)

Remarks

This class is used if the dataset is in block read mode.

Never create instances directly! Always use TnxDataSet.CreateBlobStream, which will create the right type of blob stream based on the state of the dataset.

The blob stream should be freed again as soon as possible. It must be freed before calling Post or moving to another record.

Related Topics

[TnxDataSet.CreateBlobStream](#) [TnxBaseBlobStream](#) [TnxBlobStream](#) [TnxBatchAppendBlobStream](#)
[TnxMemoBlobStream](#)

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > **TnxBlockModeBlobStream Class**

25.1.100. Methods

25.1.100.1.1 Read Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBlockModeBlobStream.Read Method

[TnxBlockModeBlobStream Class](#)

Pascal

```
public function Read(
  var aBuffer;
  aCount: Integer
): Integer; override;
```

Parameters

Parameters	Description
aBuffer	variable to store the read bytes
aCount	number of bytes to read

Remarks

With this method you can read a number of bytes of the blob data.

See Also

[TnxBlockModeBlobStream Class](#)

You are here: Symbol Reference > Classes > **TnxBlockModeBlobStream Class** > Methods > **Read Method**

25.1.100.1.2 Seek Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBlockModeBlobStream.Seek Method

[TnxBlockModeBlobStream Class](#)

Pascal

```
public function Seek(
  aOffset: Integer;
  aOrigin: Word
): Integer; override;
```

Parameters

Parameters	Description
aOffset	the offset in bytes to move the pointer
aOrigin	the origin of the aOffset

Remarks

Seek sets the current position in the blob stream.

See Also

[TnxBlockModeBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlockModeBlobStream Class > Methods > Seek Method

25.1.100.1.3 Truncate Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlockModeBlobStream.Truncate Method

[TnxBlockModeBlobStream Class](#)

Pascal

```
public procedure Truncate; override;
```

Remarks

This method truncates the blob data at the current position.

See Also

[TnxBlockModeBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlockModeBlobStream Class > Methods > Truncate Method

25.1.100.1.4 Write Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlockModeBlobStream.Write Method

[TnxBlockModeBlobStream Class](#)

Pascal

```
public function Write(
  const aBuffer;
  aCount: Integer
): Integer; override;
```

Parameters

Parameters	Description
aBuffer	data to be written
aCount	number of bytes to write

Remarks

Use Write to write a number of bytes to the blob at the current position. Take care to not accidentally overwrite data by not positioning with Seek first.

See Also

[TnxBlockModeBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxBlockModeBlobStream Class > Methods > Write Method

25.1.10 TnxBroadcastReply Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBroadcastReply Class

[nxIITransport](#) | [Members](#)

Pascal

```
public TnxBroadcastReply = class(TnxInterfacedObject, InxBroadcastReply);
```

File

[nxIITransport](#)

Remarks

This is class TnxBroadcastReply.

See Also

[nxIITransport](#)
[Members](#)

You are here: Symbol Reference > Classes > TnxBroadcastReply Class

25.1.101 Constructors

25.1.101.1 Create

25.1.101.1.1 Create Constructor (string, TnxGuid, Integer)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBroadcastReply.Create Constructor (string, TnxGuid, Integer)

TnxBroadcastReply Class

Pascal

```
public constructor Create(
    const aServerName: string;
    const aServerID: TnxGuid;
    const aServerVersion: Integer
); overload;
```

Remarks

This is Create, a member of class TnxBroadcastReply.

See Also

[TnxBroadcastReply Class](#)

You are here: Symbol Reference > Classes > TnxBroadcastReply Class > Constructors > Create > Create Constructor (string, TnxGuid, Integer)

25.1.101.1.1.2 Create Constructor (string, string, string)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBroadcastReply.Create Constructor (string, string, string)

TnxBroadcastReply Class

Pascal

```
public constructor Create(
    const aServerName: string;
    const aServerID: string;
    const aServerVersion: string
); overload;
```

Remarks

This is Create, a member of class TnxBroadcastReply.

See Also

[TnxBroadcastReply Class](#)

You are here: Symbol Reference > Classes > TnxBroadcastReply Class > Constructors > Create > Create Constructor (string, string, string)

25.1.10 TnxCachedDataSet Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCachedDataSet Class

[Events](#) | [nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxCachedDataSet = class(TnxIndexDataSet);
```

File

nxdb

Remarks

A natively caching Dataset. Attach a TnxSqlUpdateObject for allowing writeback to underlying datasets.

See Also

[Events](#)
[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class

25.1.102 Constructors

25.1.102.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCachedDataSet.Create Constructor

[TnxCachedDataSet Class](#)

Pascal

```
public constructor Create(
    AOwner: TComponent
); override;
```

Remarks

constructor.

See Also

TnxCachedDataSet Class

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Constructors > Create Constructor

25.1.102.1.Destructors

25.1.102.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.Destroy Destructor

TnxCachedDataSet Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Destructors > Destroy Destructor

25.1.102.2.Events

25.1.102.3.1 OnCreateTable Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.OnCreateTable Property

TnxCachedDataSet Class

Pascal

```
published property OnCreateTable: TnxCreateTableEvent;
```

Remarks

This event is fired before the local table is create by the server.

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Events > OnCreateTable Property

25.1.102.3.Properties

25.1.102.4.1 AddIndexDefs Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.AddIndexDefs Property

TnxCachedDataSet Class

Pascal

```
published property AddIndexDefs: TIndexDefs;
```

Remarks

???

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > AddIndexDefs Property

25.1.102.4.2 BatchSize Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxCachedDataSet.BatchSize Property**[TnxCachedDataSet Class](#)**Pascal**

```
published property BatchSize: Integer;
```

Remarks

Batch/Block size for in bytes.

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > BatchSize Property

25.1.102.4.3 FieldDefs Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxCachedDataSet.FieldDefs Property**[TnxCachedDataSet Class](#)**Pascal**

```
public property FieldDefs;
```

Remarks

see tTable.FieldDefs

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > FieldDefs Property

25.1.102.4.4 IndexDefs Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxCachedDataSet.IndexDefs Property**[TnxCachedDataSet Class](#)**Pascal**

```
public property IndexDefs;
```

Remarks

see tTable.IndexDefs.

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > IndexDefs Property

25.1.102.4.5 IndexFieldCount Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.IndexFieldCount Property

TnxCachedDataSet Class

Pascal

```
public property IndexFieldCount;
```

Remarks

see tTable.IndexFieldCount.

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > IndexFieldCount Property

25.1.102.4.6 IndexFieldNames Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.IndexFieldNames Property

TnxCachedDataSet Class

Pascal

```
published property IndexFieldNames;
```

Remarks

see tTable.IndexFieldNames

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > IndexFieldNames Property

25.1.102.4.7 IndexFields Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.IndexFields Property

TnxCachedDataSet Class

Pascal

```
public property IndexFields;
```

Remarks

see tTable.IndexFields

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > IndexFields Property

25.1.102.4.8 IndexName Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCachedDataSet.IndexName Property**[TnxCachedDataSet Class](#)**Pascal**

```
published property IndexName;
```

Remarkssee [tTable.IndexName](#)**See Also**[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > IndexName Property

25.1.102.4.9 KeyExclusive Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCachedDataSet.KeyExclusive Property**[TnxCachedDataSet Class](#)**Pascal**

```
public property KeyExclusive;
```

Remarkssee [tTable.KeyExclusive](#)**See Also**[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > KeyExclusive Property

25.1.102.4.10 KeyFieldCount Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCachedDataSet.KeyFieldCount Property**[TnxCachedDataSet Class](#)**Pascal**

```
public property KeyFieldCount;
```

Remarkssee [tTable.KeyFieldCount](#)**See Also**[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > KeyFieldCount Property

25.1.102.4.11 KeyPartialLen Property*NexusDB V2 VCL Reference*[Contents](#) | [Index](#)**TnxCachedDataSet.KeyPartialLen Property**[TnxCachedDataSet Class](#)**Pascal**

```
public property KeyPartialLen;
```

Remarks

see [tTable.KeyPartialLen](#)

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > KeyPartialLen Property

25.1.102.4.12 MasterFields Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxCachedDataSet.MasterFields Property**[TnxCachedDataSet Class](#)**Pascal**

```
published property MasterFields;
```

Remarks

see [tTable.MasterFields](#)

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > MasterFields Property

25.1.102.4.13 MasterSource Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxCachedDataSet.MasterSource Property**[TnxCachedDataSet Class](#)**Pascal**

```
published property MasterSource;
```

Remarks

see [tTable.MasterSource](#)

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > MasterSource Property

25.1.102.4.14 Options Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.Options Property

TnxCachedDataSet Class

Pascal

```
published property Options: TnxCachedDataSetOptions;
```

Remarks

Options for the cached dataset.

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > Options Property

25.1.102.4.15 ReadOnly Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.ReadOnly Property

TnxCachedDataSet Class

Pascal

```
published property ReadOnly: Boolean;
```

Remarks

see tTable.ReadOnly

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > ReadOnly Property

25.1.102.4.16 SourceDataSet Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSet.SourceDataSet Property

TnxCachedDataSet Class

Pascal

```
published property SourceDataSet: TnxDataSet;
```

Remarks

The dataset the data is fetched from.

See Also

[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > SourceDataSet Property

25.1.102.4.17 TableName Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCachedDataSet.TableName Property**[TnxCachedDataSet Class](#)**Pascal**

```
published property TableName: string;
```

Remarks

Original Tablename. ???

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > TableName Property

25.1.102.4.18 UpdateObject Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCachedDataSet.UpdateObject Property**[TnxCachedDataSet Class](#)**Pascal**

```
published property UpdateObject: TnxBaseUpdateObject;
```

Remarks

The object responsible for updating the underlying datasets.

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > UpdateObject Property

25.1.102.4.19 UsedTableName Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCachedDataSet.UsedTableName Property**[TnxCachedDataSet Class](#)**Pascal**

```
published property UsedTableName: string;
```

Remarks

???

See Also[TnxCachedDataSet Class](#)

You are here: Symbol Reference > Classes > TnxCachedDataSet Class > Properties > UsedTableName Property

25.1.10.TnxCompKeyDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxCompKeyDescriptor = class(TnxBaseKeyDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

The Compound Key descriptor uses a number of Key Field descriptor to create the keys.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class

25.1.103.Destructors

25.1.103.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor.Destroy Destructor

TnxCompKeyDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

public comes before protected to fix problem with BCB destructor.

See Also

[TnxCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Destructors > Destroy Destructor

25.1.103.Methods

25.1.103.2.1 Add Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor.Add Method

[TnxCompKeyDescriptor Class](#) | [Related Topics](#)

Pascal

```
public function Add(
  aFieldNumber: Integer;
  aClass: TnxKeyFieldDescriptorClass = nil
): TnxKeyFieldDescriptor; virtual;
```

Remarks

This function adds a new descriptor to the list of processed Key Field Descriptors.

Related Topics

[Delete](#), [Clear](#), [KeyFieldCount](#), [KeyFields](#)

See Also

[TnxCompKeyDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Methods > Add Method

25.1.103.2.2 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor.CheckValid Method

[TnxCompKeyDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxCompKeyDescriptor.

See Also

[TnxCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Methods > CheckValid Method

25.1.103.2.3 Clear Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor.Clear Method

[TnxCompKeyDescriptor Class](#) | [Related Topics](#)

Pascal

```
public procedure Clear;
```

Remarks

Clears the list of Key Field Descriptors.

Related Topics

[Add](#), [Delete](#), [Clear](#), [KeyFieldCount](#), [KeyFields](#)

See Also

[TnxCompKeyDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Methods > Clear Method

25.1.103.2.4 Delete Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor.Delete Method

[TnxCompKeyDescriptor Class](#) | [Related Topics](#)

Pascal

```
public procedure Delete(
    aInx: Integer
);
```

Remarks

This function deletes a descriptor from the list of processed Key Field Descriptors.

Related Topics

[Add](#), [Delete](#), [Clear](#), [KeyFieldCount](#), [KeyFields](#)

See Also

[TnxCompKeyDescriptor Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Methods > Delete Method

25.1.103.2.5 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor.IsEqual Method

[TnxCompKeyDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxCompKeyDescriptor.

See Also

[TnxCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Methods > IsEqual Method

25.1.103.2.6 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCompKeyDescriptor.LoadFromReader Method

[TnxCompKeyDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxCompKeyDescriptor.

See Also

[TnxCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Methods > LoadFromReader Method

25.1.103.2.7 SaveToWriter Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxCompKeyDescriptor.SaveToWriter Method**[TnxCompKeyDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxCompKeyDescriptor.

See Also

[TnxCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Methods > [SaveToWriter Method](#)

25.1.103.Properties

25.1.103.3.1 KeyFieldCount Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxCompKeyDescriptor.KeyFieldCount Property**[TnxCompKeyDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public property KeyFieldCount: Integer;
```

Remarks

This property returns the number of items in the Key Field descriptor List.

Related Topics

[Add](#), [Delete](#), [Clear](#), [KeyFieldCount](#), [KeyFields](#)

See Also

[TnxCompKeyDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Properties > [KeyFieldCount Property](#)

25.1.103.3.2 KeyFields Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxCompKeyDescriptor.KeyFields Property**[TnxCompKeyDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public property KeyFields [aInx: Integer]: TnxKeyFieldDescriptor;
```

Remarks

This property return the Key Field descriptor with the given index out of the Key Field Descriptor list.

Related Topics

[Add](#), [Delete](#), [Clear](#), [KeyFieldCount](#), [KeyFields](#)

See Also

[TnxCompKeyDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxCompKeyDescriptor Class > Properties > KeyFields Property

25.1.10 TnxComponent Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxComponent Class

[nxllComponent](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TrxComponent = class(TComponent);
```

File

[nxllComponent](#)

Remarks

The mother of most NexusDB components. Includes version info, UI settings, persisting options, error checking.

See Also

[nxllComponent](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxComponent Class

25.1.104 Constructors

25.1.104.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxComponent.Create Constructor

[TnxComponent Class](#)

Pascal

```
public constructor Create(
  aOwner: TComponent
); override;
```

Remarks

constructor

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Constructors > Create Constructor

25.1.104.:Destructors

25.1.104.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.Destroy Destructor

TnxComponent Class

Pascal

```
public destructor Destroy; override;
```

Remarks

denstructor

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Destructors > Destroy Destructor

25.1.104.:Methods

25.1.104.3.1 BeforeDestruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.BeforeDestruction Method

TnxComponent Class

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TComponent.BeforeDestruction

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > BeforeDestruction Method

25.1.104.3.2 ClearStats Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.ClearStats Method

TnxComponent Class

Pascal

```
public procedure ClearStats; virtual;
```

Remarks

Resets all internal stats.

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > ClearStats Method

25.1.104.3.3 FindClassRecursive Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxComponent.FindClassRecursive Method**[TnxComponent Class](#)**Pascal**

```
public function FindClassRecursive(
    aSuffix: String
): TClass; virtual;
```

Remarks

Tries to find the highest class in the class hierarchy with a certain suffix added to the current classname.

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > FindClassRecursive Method

25.1.104.3.4 FreeInstance Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxComponent.FreeInstance Method**[TnxComponent Class](#)**Pascal**

```
public procedure FreeInstance; override;
```

Remarks

Overriden to take advantage of the NexusDB Memory Manager.

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > FreeInstance Method

25.1.104.3.5 GetConfigSettings Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxComponent.GetConfigSettings Method**[TnxComponent Class](#)**Pascal**

```
public procedure GetConfigSettings(
    aSettings: TnxBaseSettings
); virtual;
```

Remarks

Returns the defintion of all persistable settings.

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > GetConfigSettings Method

25.1.104.3.6 GetStatsCaptions Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxComponent.GetStatsCaptions Method**[TnxComponent Class](#)**Pascal**

```
public procedure GetStatsCaptions(
  const aList: TStrings
); virtual;
```

Remarks

Returns the captions for the returned statistic values.

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > GetStatsCaptions Method

25.1.104.3.7 GetStatsValues Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxComponent.GetStatsValues Method**[TnxComponent Class](#)**Pascal**

```
public procedure GetStatsValues(
  const aList: TStrings
); virtual;
```

Remarks

Returns statistic values.

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > GetStatsValues Method

25.1.104.3.8 IterateDependents Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxComponent.IterateDependents Method**[TnxComponent Class](#)**Pascal**

```
public function IterateDependents(
  aIterationListClass: TnxIterationListClass = nil
): TnxIterationList;
```

Remarks

Adds all dependent classes to a TnxIterationList. You can enforce the usage of an alternative Iteration List class that can e.g. do some filtering for certain class types.

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > IterateDependents Method

25.1.104.3.9 LoadConfig Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.LoadConfig Method

TnxComponent Class

Pascal

```
public procedure LoadConfig(
    aConfig: TnxBaseComponentConfiguration
); virtual;
```

Remarks

Loads persisted settings.

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > LoadConfig Method

25.1.104.3.10 LoadSettingsFromStream Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.LoadSettingsFromStream Method

TnxComponent Class

Pascal

```
public procedure LoadSettingsFromStream(
    aReader: TnxReader
); virtual;
```

Remarks

Loads persisted settings. (legacy V1)

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > LoadSettingsFromStream Method

25.1.104.3.11 New Instance Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.NewInstance Method

TnxComponent Class

Pascal

```
public class function NewInstance: TObject; override;
```

Remarks

Overridden to take advantage of the NexusDB Memory Manager.

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > NewInstance Method

25.1.104.3.12 SaveConfig Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.SaveConfig Method

TnxComponent Class

Pascal

```
public procedure SaveConfig(
    aConfig: TnxBaseComponentConfiguration
); virtual;
```

Remarks

Persists settings.

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > SaveConfig Method

25.1.104.3.13 SaveSettingsToStream Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.SaveSettingsToStream Method

TnxComponent Class

Pascal

```
public procedure SaveSettingsToStream(
    aWriter: TnxWriter
); virtual;
```

Remarks

Persists settings. (legacy V1)

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > SaveSettingsToStream Method

25.1.104.3.14 Supported Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.Supported Method

TnxComponent Class

Pascal

```
public function Supported: Boolean; virtual;
```

Remarks

This function returns true if the component is supported in the current environment (e.g. windows version)

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > Supported Method

25.1.104.3.15 UISettingsVisible Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.UISettingsVisible Method

TnxComponent Class

Pascal

```
public function UISettingsVisible: Boolean; virtual;
```

Remarks

Are there UI settings visible for this component? (legacy V1)

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > UISettingsVisible Method

25.1.104.3.16 UIStatsVisible Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.UIStatsVisible Method

TnxComponent Class

Pascal

```
public function UIStatsVisible: Boolean; virtual;
```

Remarks

Are there UI statistics visible for this component? (legacy V1)

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Methods > UIStatsVisible Method

25.1.104.Properties

25.1.104.4.1 DisplayCategory Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxComponent.DisplayCategory Property

TnxComponent Class

Pascal

```
published property DisplayCategory: String;
```

Remarks

Name of the category if used in a UI. Can be multiple hierarchies by using / (legacy V1)

See Also

[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Properties > DisplayCategory Property

25.1.104.4.2 DisplayName Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxComponent.DisplayName Property**[TnxComponent Class](#)**Pascal**

```
published property DisplayName: string;
```

Remarks

Name of the component if used in a UI. (legacy V1)

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Properties > DisplayName Property

25.1.104.4.3 Version Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxComponent.Version Property**[TnxComponent Class](#)**Pascal**

```
published property Version: string;
```

Remarks

Version of the component.

See Also[TnxComponent Class](#)

You are here: Symbol Reference > Classes > TnxComponent Class > Properties > Version Property

25.1.105TnxConstDefaultValueDescriptor Class*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxConstDefaultValueDescriptor Class**[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxConstDefaultValueDescriptor = class(TnxBaseDefaultValueDescriptor);
```

File[nxsdDataDictionary](#)**Remarks**

This is the implementation for constant default values.

See Also[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxConstDefaultValueDescriptor Class

25.1.105. Methods

25.1.105.1.1 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxConstDefaultValueDescriptor.IsEqual Method

TnxConstDefaultValueDescriptor Class

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxConstDefaultValueDescriptor.

See Also

[TnxConstDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxConstDefaultValueDescriptor Class > Methods > [IsEqual Method](#)

25.1.105.1.2 LoadFromReader Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxConstDefaultValueDescriptor.LoadFromReader Method

TnxConstDefaultValueDescriptor Class

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxConstDefaultValueDescriptor.

See Also

[TnxConstDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxConstDefaultValueDescriptor Class > Methods > [LoadFromReader Method](#)

25.1.105.1.3 SaveToWriter Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxConstDefaultValueDescriptor.SaveToWriter Method

TnxConstDefaultValueDescriptor Class

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxConstDefaultValueDescriptor.

See Also

TnxConstDefaultValueDescriptor Class

You are here: Symbol Reference > Classes > TnxConstDefaultValueDescriptor Class > Methods > SaveToWriter Method

25.1.105.1Properties

25.1.105.2.1 AsVariant Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxConstDefaultValueDescriptor.AsVariant Property

TnxConstDefaultValueDescriptor Class

Pascal

```
public property AsVariant: OleVariant;
```

Remarks

Gets/Sets the constant value as Variant.

See Also

[TnxConstDefaultValueDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxConstDefaultValueDescriptor Class > Properties > AsVariant Property

25.1.106TnxCurrentDateTimeDefaultValueDescriptor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCurrentDateTimeDefaultValueDescriptor Class

nxsdDataDictionary

Pascal

```
public TnxCurrentDateTimeDefaultValueDescriptor = class(TnxBaseDefaultValueDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This descriptor should be used if you want to set a DateTime field to the current server DateTime.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > TnxCurrentDateTimeDefaultValueDescriptor Class

25.1.107TnxCurrentUserDefaultValueDescriptor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCurrentUserDefaultValueDescriptor Class

nxsdDataDictionary

Pascal

```
public TnxCurrentUserDefaultValueDescriptor = class(TnxBaseDefaultValueDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This descriptor should be used if you want to set a string field to the current logged in user.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > TnxCurrentUserDefaultValueDescriptor Class

25.1.10{TnxCursor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCursor Class

[nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxCursor = class(TnxDataAccessStateComponent);
```

File

[nxdb](#)

Remarks

Client side base dataset cursor class.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxCursor Class

25.1.108. Constructors

25.1.108.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCursor.Create Constructor

[TnxCursor Class](#)

Pascal

```
public constructor Create(
    aDataSet: TnxDataset
); reintroduce;
```

Remarks

constructor.

See Also

[TnxCursor Class](#)

You are here: Symbol Reference > Classes > TnxCursor Class > Constructors > Create Constructor

25.1.108.Properties

25.1.108.2.1 Database Property

NexusDB V2 VCL Reference
TnxCursor.Database Property

[Contents](#) | [Index](#)

TnxCursor Class

Pascal

```
public property Database: TnxDatabase;
```

Remarks

The database the cursor belongs to.

See Also

[TnxCursor Class](#)

You are here: Symbol Reference > Classes > TnxCursor Class > Properties > Database Property

25.1.10 TnxCustomConnectionFilter Class

NexusDB V2 VCL Reference
TnxCustomConnectionFilter Class

[Contents](#) | [Index](#)

[Events](#) | [nxtwWinsockTransport](#) | [Members](#) | [Related Topics](#)

Pascal

```
public TnxCustomConnectionFilter = class(TnxBaseConnectionFilter);
```

File

[nxtwWinsockTransport](#)

Remarks

Component used to filter connections based on IP address. Assign this component to the ConnectionFilter property of a TnxWinsockTransport component, and implement the OnAcceptConnection event.

Related Topics

[ConnectionFilter](#), [TnxWinsockTransport](#), [OnAcceptConnection](#)

See Also

[Events](#)
[nxtwWinsockTransport](#)
[Members](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxCustomConnectionFilter Class

25.1.109.Events

25.1.109.1.1 OnAcceptConnection Property

NexusDB V2 VCL Reference
TnxCustomConnectionFilter.OnAcceptConnection Property

[Contents](#) | [Index](#)

[TnxCustomConnectionFilter Class](#)

Pascal

```
published property OnAcceptConnection: TnxOnAcceptConnection;
```

Remarks

Implement this event to selectively accept connections on different IP addresses.

See Also

[TnxCustomConnectionFilter Class](#)

You are here: Symbol Reference > Classes > TnxCustomConnectionFilter Class > Events > OnAcceptConnection Property

25.1.11(TnxCustomDescsDescriptor Class)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxCustomDescsDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

Holds a list of CustomDescriptors.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class

25.1.110.Constructors

25.1.110.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptor.Create Constructor

[TnxCustomDescsDescriptor Class](#)

Pascal

```
public constructor Create(
  aParent: TPersistent
); virtual;
```

Remarks

constructor.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Constructors > Create Constructor

25.1.110.:Destructors

25.1.110.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCustomDescsDescriptor.Destroy Destructor

TnxCustomDescsDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Destructors > Destroy Destructor

25.1.110.:Methods

25.1.110.3.1 AddCustom Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCustomDescsDescriptor.AddCustom Method

TnxCustomDescsDescriptor Class

Pascal

```
public function AddCustom(
  const aName: string;
  aClass: TnxBaseCustomDescriptorClass = nil
): TnxBaseCustomDescriptor;
```

Parameters

Parameters

aName

Description

the name of the Custom Descriptor

aClass

the class of the instance to be created.

Remarks

This function adds a new instance of a Custom Descriptor to the data dictionary and returns the newly created Instance.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > AddCustom Method

25.1.110.3.2 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCustomDescsDescriptor.CheckValid Method

TnxCustomDescsDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxCustomDescsDescriptor.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > [CheckValid Method](#)

25.1.110.3.3 Clear Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptor.Clear Method

[TnxCustomDescsDescriptor Class](#)

Pascal

```
public procedure Clear;
```

Remarks

This function clears the Custom Descriptors Descriptor.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > [Clear Method](#)

25.1.110.3.4 GetCustomDescriptorFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptor.GetCustomDescriptorFromName Method

[TnxCustomDescsDescriptor Class](#)

Pascal

```
public function GetCustomDescriptorFromName(
  const aCustomDescriptorName: string
): Integer;
```

Remarks

Return the custom number for a given custom name, or -1 if not found.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > [GetCustomDescriptorFromName Method](#)

25.1.110.3.5 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptor.AreEqual Method

[TnxCustomDescsDescriptor Class](#)

Pascal

```
public function IsEqual(
  aDictionaryItem: TnxDictionaryItem
```

```
    ): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxCustomDescsDescriptor.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > IsEqual Method

25.1.110.3.6 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptor.LoadFromReader Method

[TnxCustomDescsDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxCustomDescsDescriptor.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > LoadFromReader Method

25.1.110.3.7 RemoveCustom

25.1.110.3.7.1 RemoveCustom Method (Integer)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptor.RemoveCustom Method (Integer)

[TnxCustomDescsDescriptor Class](#)

Pascal

```
public procedure RemoveCustom(
    aInx: Integer
); overload;
```

Parameters

Parameters	Description
aInx	the index of the descriptor to be removed in the list of Custom Descriptors.

Remarks

Remove a Custom Descriptor from the Data Dictionary.

See Also

[TnxCustomDescsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > RemoveCustom > RemoveCustom Method (Integer)

25.1.110.3.7.2 RemoveCustom Method (string)

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCustomDescsDescriptor.RemoveCustom Method (string)**[TnxCustomDescsDescriptor Class](#)**Pascal**

```
public procedure RemoveCustom(
  const aName: string
); overload;
```

Parameters**Parameters**

aName

Description

the name of the descriptor to be removed.

Remarks

Remove a Custom Descriptor from the Data Dictionary.

See Also[TnxCustomDescsDescriptor Class](#)*You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > RemoveCustom > RemoveCustom Method (string)*

25.1.110.3.8 SaveToWriter Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxCustomDescsDescriptor.SaveToWriter Method**[TnxCustomDescsDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxCustomDescsDescriptor.

See Also[TnxCustomDescsDescriptor Class](#)*You are here: Symbol Reference > Classes > TnxCustomDescsDescriptor Class > Methods > SaveToWriter Method***25.1.11 TnxDataAccessStateComponent Class***NexusDB V2 VCL Reference*[Contents | Index](#)**TnxDataAccessStateComponent Class**

nxdb

Pascal

```
public TnxDataAccessStateComponent = class(TnxStateComponent);
```

File

nxdb

Remarks

The base class for all client side data components

See Also

[nxdb](#)

You are here: Symbol Reference > Classes > TnxDataAccessStateComponent Class

25.1.11. TnxDatabase Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxDatabase = class(TnxSessionOwnedDataAccessStateComponent);
```

File

[nxdb](#)

Remarks

Implementation of a TDatabase similar component.

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxDatabase Class

25.1.112. Constructors

25.1.112.1. Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.Create Constructor

[TnxDatabase Class](#)

Pascal

```
public constructor Create(
  AOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Constructors > Create Constructor

25.1.112.Destructors

25.1.112.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.Destroy Destructor

TnxDatabase Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Destructors > Destroy Destructor

25.1.112.Methods

25.1.112.3.1 BackupTable Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.BackupTable Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public function BackupTable(
    const aTableName: string;
    const aPassword: string;
    aTarget: TnxDatabase
): TnxAbstractTaskInfo;
```

Parameters

Parameters	Description
aTarget	the database where the backup is created
aName	table name

Remarks

This method starts an asynchronous backup of the table. For this purpose a new table with identical fields is created and the records are copied over from the old one. Raises an exception on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTableEx](#), [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > BackupTable Method

25.1.112.3.2 BackupTableEx Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.BackupTableEx Method

[TnxDatabase Class](#) | [Related Topics](#)

```
Pascal
public function BackupTableEx(
    const aTableName: string;
    const aPassword: string;
    aTarget: TnxDatabase;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult;
```

Parameters

Parameters	Description
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aName	table name

Remarks

This method starts an asynchronous backup of the table. For this purpose a new table with identical fields is created and the records are copied over from the old one. Returns an error code on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > BackupTableEx Method

25.1.112.3.3 ChangePassw ord Method

[NexusDB V2 VCL Reference](#)

[Contents](#) | [Index](#)

TnxDatabase.ChangePassword Method

[TnxDatabase Class](#)

```
Pascal
public procedure ChangePassword(
    const aTableName: string;
    const aOldPassword: string;
    const aNewPassword: string
);
```

Remarks

Changes the password of a table and triggers an exception if it failed.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ChangePassword Method

25.1.112.3.4 ChangePassw ordEx Method

[NexusDB V2 VCL Reference](#)

[Contents](#) | [Index](#)

TnxDatabase.ChangePasswordEx Method

TnxDatabase Class

```
Pascal
public function ChangePasswordEx(
    const aTableName: string;
    const aOldPassword: string;
    const aNewPassword: string
) : TnxResult;
```

Remarks

Changes the password of a table and returns an error if failed.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ChangePasswordEx Method

25.1.112.3.5 Commit Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.Commit Method

[TnxDatabase Class](#) | Related Topics

```
Pascal
public procedure Commit;
```

Remarks

This method commits a pending transaction (level). If an error occurs (e.g. deadlock detected or transaction level has been marked as corrupted) the transaction is **not** automatically rolled back. You have to call Rollback in that case.

For more info see the nexus manual.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > Commit Method

25.1.112.3.6 CreateTable Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.CreateTable Method

[TnxDatabase Class](#) | Related Topics

```
Pascal
public procedure CreateTable(
    aOverWrite: Boolean;
    const aTableName: string;
    const aPassword: string;
    aDictionary: TnxDataDictionary;
    aTableScope: TnxTableScope = tsPersistent
);
```

Parameters

Parameters	Description
aOverWrite	if true, an existing table is overwritten if it is not in use
aTableName	the name of the new table
aDictionary	the dictionary that defines the table and index structure

Remarks

Use CreateTable to create a new table with the structure defined by the given dictionary. Raises an exception on failure.

Related Topics

[DeleteTable](#), [CreateTableEx](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > CreateTable Method

25.1.112.3.7 CreateTableEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.CreateTableEx Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public function CreateTableEx(
  aOverWrite: Boolean;
  const aTableName: string;
  const aPassword: string;
  aDictionary: TnxDataDictionary;
  aTableScope: TnxTableScope = tsPersistent
): TnxResult;
```

Parameters

Parameters	Description
aOverWrite	if true, an existing table is overwritten if it is not in use
aTableName	the name of the new table
aDictionary	the dictionary that defines the table and index structure

Remarks

Use CreateTable to create a new table with the structure defined by the given dictionary. Returns an error code on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > CreateTableEx Method

25.1.112.3.8 DeleteTable Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.DeleteTable Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public procedure DeleteTable(
    const aTableName: string;
    const aPassword: string
);
```

Parameters

Parameters	Description
aTableName	table name

Remarks

This method deletes the table with the given name.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > DeleteTable Method

25.1.112.3.9 EmptyTable Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.EmptyTable Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public procedure EmptyTable(
    const aTableName: string;
    const aPassword: string
);
```

Parameters

Parameters	Description
aTableName	table name

Remarks

EmptyTable deletes **all** records of the given table.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > EmptyTable Method

25.1.112.3.10 ExecQuery Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.ExecQuery Method

TnxDatabase Class

Pascal

```
public function ExecQuery(
  const aQuery: string;
  const aParams: array of const
): TnxDynVariantArray;
```

Remarks

statement support

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ExecQuery Method

25.1.112.3.11 ExecStoredProcedure Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.ExecStoredProcedure Method

TnxDatabase Class

Pascal

```
public function ExecStoredProcedure(
  const aStoredProcedureName: string;
  const aParams: array of const
): TnxDynVariantArray;
```

Parameters

Parameters	Description
aStoredProcedureName	the name of the procedure
aParams	an array of parameter values

Remarks

Executes a SQL Stored Procedure and returns it's result.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ExecStoredProcedure Method

25.1.112.3.12 Exists Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.Exists Method

TnxDatabase Class

Pascal

```
public function Exists: Boolean;
```

Remarks

Returns true if the database exists.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > Exists Method

25.1.112.3.13 GetAutoIncValue Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.GetAutoIncValue Method

[TnxDatabase Class](#)

Pascal

```
public procedure GetAutoIncValue(
  const aTableName: string;
  const aPassword: string;
  out aValue: TnxWord32
);
```

Remarks

Returns the next AutoInc value for the given table.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > GetAutoIncValue Method

25.1.112.3.14 GetChangedTables Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.GetChangedTables Method

[TnxDatabase Class](#)

Pascal

```
public procedure GetChangedTables(
  aList: TStrings
);
```

Parameters

Parameters	Description
aList	will hold the names after the method returns; must not be nil

Remarks

GetChangedTables returns a list of all open tables in this database that have changed since the function was last called.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > GetChangedTables Method

25.1.112.3.15 GetDataDictionary Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDatabase.GetDataDictionary Method**[TnxDatabase Class](#)**Pascal**

```
public procedure GetDataDictionary(
  const aTableName: string;
  const aPassword: string;
  aDictionary: TnxDataDictionary
);
```

Parameters

Parameters	Description
aTableName	table name
aDictionary	will hold the dictionary after the method returns; must not be nil

Remarks

The method returns the data dictionary of a given table. Raises an exception on failure.

See Also[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > GetDataDictionary Method

25.1.112.3.16 GetDataDictionaryEx Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDatabase.GetDataDictionaryEx Method**[TnxDatabase Class](#)**Pascal**

```
public function GetDataDictionaryEx(
  const aTableName: string;
  const aPassword: string;
  aDictionary: TnxDataDictionary
): TnxResult;
```

Parameters

Parameters	Description
aTableName	table name
aDictionary	will hold the dictionary after the method returns; must not be nil

Remarks

The method returns the data dictionary of a given table. Returns an error code on failure.

See Also[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > GetDataDictionaryEx Method

25.1.112.3.17 GetFreeDiskSpace Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.GetFreeDiskSpace Method

TnxDatabase Class

Pascal

```
public function GetFreeDiskSpace: TnxInt64;
```

Parameters

Parameters

aFreeSpace

Description

will hold the free space info after the method returns

Remarks

This method returns the remaining disk space for the disk this database/alias is stored on.

See Also

[TnxDatabase Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDatabase Class](#) > [Methods](#) > [GetFreeDiskSpace Method](#)

25.1.112.3.18 GetFreeDiskSpaceEx Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.GetFreeDiskSpaceEx Method

TnxDatabase Class

Pascal

```
public function GetFreeDiskSpaceEx(
    out aFreeSpace: TnxInt64
): TnxResult;
```

Parameters

Parameters

aFreeSpace

Description

will hold the free space info after the method returns

Remarks

This method returns the remaining disk space for the disk this database/alias is stored on.

See Also

[TnxDatabase Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDatabase Class](#) > [Methods](#) > [GetFreeDiskSpaceEx Method](#)

25.1.112.3.19 GetPath Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.GetPath Method

TnxDatabase Class

Pascal

```
public function GetPath: string;
```

Remarks

Returns the full (server based!) system path of the database.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > GetPath Method

25.1.112.3.20 GetStoredProcNames Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.GetStoredProcNames Method

[TnxDatabase Class](#)

Pascal

```
public procedure GetStoredProcNames (
    aList: TStrings
);
```

Parameters

Parameters	Description
aList	will hold the names after the method returns; must not be nil

Remarks

GetStoredProcNames returns a list of all stored procedures of the database.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > GetStoredProcNames Method

25.1.112.3.21 GetStoredProcParams Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.GetStoredProcParams Method

[TnxDatabase Class](#)

Pascal

```
public procedure GetStoredProcParams (
    const aStoredProcName: string;
    out aParams: TnxSqlParameterList
);
```

Remarks

Returns the parameter definitions for a named stored procedure.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > GetStoredProcParams Method

25.1.112.3.22 GetTableNames Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDatabase.GetTableNames Method**[TnxDatabase Class](#)**Pascal**

```
public procedure GetTableNames(
  aList: TStrings
);
```

Parameters

Parameters	Description
aList	will hold the names after the method returns; must not be nil

Remarks

GetTableNames returns a list of all tables of the database.

See Also[TnxDatabase Class](#)*You are here:* Symbol Reference > Classes > TnxDatabase Class > Methods > GetTableNames Method

25.1.112.3.23 OpenQuery Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDatabase.OpenQuery Method**[TnxDatabase Class](#)**Pascal**

```
public function OpenQuery(
  const aQuery: string;
  const aParams: array of const;
  aBlockReadSize: Integer = 0;
  aRequestLive: Boolean = False
) : TnxQuery;
```

Parameters

Parameters	Description
aQuery	the SQL query as string
aParams	array of parameter values
aBlockReadSize	if > 0 then it's the size in bytes for block reading the resulting cursor
aRequestLive	if true then tries to get a live result set

Remarks

Executes a SQL query and returns a cursor (if applicable)

See Also[TnxDatabase Class](#)*You are here:* Symbol Reference > Classes > TnxDatabase Class > Methods > OpenQuery Method

25.1.112.3.24 OpenStoredProc Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDatabase.OpenStoredProc Method**[TnxDatabase Class](#)**Pascal**

```
public function OpenStoredProc(
    const aStoredProcName: string;
    const aParams: array of const;
    aBlockReadSize: Integer = 0
): TnxStoredProc;
```

Parameters

Parameters	Description
aStoredProcName	the name of the procedure
aParams	an array of parameter values

Remarks

Opens SQL Stored Procedure and returns it's local instance

See Also[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > OpenStoredProc Method

25.1.112.3.25 OpenTable Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDatabase.OpenTable Method**[TnxDatabase Class](#)**Pascal**

```
public function OpenTable(
    const aTableName: string;
    const aIndexName: string = '';
    aBlockReadSize: Integer = 0
): TnxTable;
```

Parameters

Parameters	Description
aTableName	the name of the table
aIndexName	the name of the index
aBlockReadSize	if > 0 then it's the size in bytes for block reading the resulting cursor

Remarks

Opens a Table.

See Also[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > OpenTable Method

25.1.112.3.26 PackTable Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDatabase.PackTable Method**[TnxDatabase Class](#) | [Related Topics](#)**Pascal**

```
public function PackTable(
  const aTableName: string;
  const aPassword: string
): TnxAbstractTaskInfo;
```

Parameters

Parameters	Description
aName	table name
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.

Remarks

This method starts an asynchronous packing of the table. For this purpose a new identical table is created and the records are copied over from the old one. Raises an exception on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTableEx](#), [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > PackTable Method

25.1.112.3.27 PackTableEx Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDatabase.PackTableEx Method**[TnxDatabase Class](#) | [Related Topics](#)**Pascal**

```
public function PackTableEx(
  const aTableName: string;
  const aPassword: string;
  out aTaskInfo: TnxAbstractTaskInfo
): TnxResult;
```

Parameters

Parameters	Description
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aName	table name

Remarks

This method starts an asynchronous packing of the table. For this purpose a new identical table is created and the records are copied over from the old one. Returns an error code on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > PackTableEx Method

25.1.112.3.28 RecoverTable Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.RecoverTable Method

[TnxDatabase Class](#)

Pascal

```
public function RecoverTable(
    const aTableName: string;
    const aPassword: string
): TnxAbstractTaskInfo;
```

Parameters

Parameters	Description
aTableName	table name

Remarks

The method attempts to recover records from a broken table. Recovered records are placed in a new table named TableName_Recovered. Triggers and exception on failure. Unrecoverable records are placed in a new table named TableName_Failed.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > RecoverTable Method

25.1.112.3.29 RecoverTableEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.RecoverTableEx Method

[TnxDatabase Class](#)

Pascal

```
public function RecoverTableEx(
    const aTableName: string;
    const aPassword: string;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult;
```

Parameters

Parameters	Description
aTableName	table name

Remarks

The method attempts to recover records from a broken table. Recovered records are placed in a new table named TableName_Recovered. Returns an error value on failure. Unrecoverable records are placed in a new table named TableName_Failed.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > RecoverTableEx Method

25.1.112.3.30 ReIndexTable

25.1.112.3.30.1 ReIndexTable Method (string, string, Integer)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.ReIndexTable Method (string, string, Integer)

[TnxDatabase Class](#) | Related Topics

Pascal

```
public function ReIndexTable(
  const aTableName: string;
  const aPassword: string;
  aIndexNum: Integer
): TnxAbstractTaskInfo; overload;
```

Parameters

Parameters	Description
aIndexNum	the index number of the index in the list of indexes
aName	table name
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.

Remarks

ReindexTable starts an asynchronous re indexing of a certain index of a table. Raises an exception on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#) [ReIndexTableEx](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ReIndexTable > ReIndexTable Method (string, string, Integer)

25.1.112.3.30.2 ReIndexTable Method (string, string, string)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.ReIndexTable Method (string, string, string)

[TnxDatabase Class](#) | Related Topics

Pascal

```
public function ReIndexTable(
  const aTableName: string;
```

```

    const aPassword: string;
    const aIndexName: string
  ): TnxAbstractTaskInfo; overload;

```

Parameters

Parameters	Description
aIndexName	the name of the index
aName	table name
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.

Remarks

ReIndexTable starts an asynchronous re indexing of a certain index of a table. Raises an exception on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTableEx](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ReIndexTable > ReIndexTable Method (string, string, string)

25.1.112.3.31 ReIndexTableEx

25.1.112.3.31.1 ReIndexTableEx Method (string, string, Integer, TnxAbstractTaskInfo)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.ReIndexTableEx Method (string, string, Integer, TnxAbstractTaskInfo)

[TnxDatabase Class](#) | Related Topics

Pascal

```

public function ReIndexTableEx(
  const aTableName: string;
  const aPassword: string;
  aIndexNum: Integer;
  out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; overload;

```

Parameters

Parameters	Description
aIndexNum	the index number of the index in the list of indexes
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aName	table name

Remarks

ReindexTableEx starts an asynchronous re indexing of a certain index of a table. Returns an error code on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#) [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ReIndexTableEx > ReIndexTableEx Method (string, string, Integer, TnxAbstractTaskInfo)

25.1.112.3.31.2 ReIndexTableEx Method (string, string, string, TnxAbstractTaskInfo)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.ReIndexTableEx Method (string, string, string, TnxAbstractTaskInfo)

[TnxDatabase Class](#) | Related Topics

Pascal

```
public function ReIndexTableEx(
  const aTableName: string;
  const aPassword: string;
  const aIndexName: string;
  out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; overload;
```

Parameters

Parameters	Description
aIndexName	the name of the index
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aName	table name

Remarks

ReIndexTableEx starts an asynchronous re indexing of a certain index of a table. Returns an error code on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > ReIndexTableEx > ReIndexTableEx Method (string, string, string, TnxAbstractTaskInfo)

25.1.112.3.32 RenameTable Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.RenameTable Method

[TnxDatabase Class](#) | Related Topics

Pascal

```
public procedure RenameTable(
    const aOldName: string;
    const aNewName: string;
    const aPassword: string
);
```

Parameters

Parameters	Description
aOldName	name of existing table
aNewName	new name for the table

Remarks

RenameTable changes the name of a table.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > RenameTable Method

25.1.112.3.33 RestructureTable Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.RestructureTable Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public function RestructureTable(
    const aTableName: string;
    const aPassword: string;
    aDictionary: TnxDataDictionary;
    aMapperDesc: TnxBaseTableMapperDescriptor
) : TnxAbstractTaskInfo;
```

Parameters

Parameters	Description
aDictionary	new dictionary of the table
aName	table name
aFieldName	defines how fields are mapped from the old to the new structure
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.

Remarks

With this method you can start an asynchronous restructure of a given table. Raises an exception on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > RestructureTable Method

25.1.112.3.34 RestructureTableEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.RestructureTableEx Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public function RestructureTableEx(
  const aTableName: string;
  const aPassword: string;
  aDictionary: TnxDataDictionary;
  aMapperDesc: TnxBaseTableMapperDescriptor;
  out aTaskInfo: TnxAbstractTaskInfo
): TnxResult;
```

Parameters

Parameters	Description
aDictionary	new dictionary of the table
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.
aName	table name
aFieldName	defines how fields are mapped from the old to the new structure

Remarks

With this method you can start an asynchronous restructure of a given table. Returns an error code on failure.

Related Topics

[DeleteTable](#), [CreateTable](#), [EmptyTable](#), [RenameTable](#), [PackTable](#), [ReIndexTable](#), [RestructureTable](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > RestructureTableEx Method

25.1.112.3.35 Rollback Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.Rollback Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public procedure Rollback;
```

Remarks

Rollback rolls (as the name says) back a pending transaction (level). All changes since the last StartTransaction (or equivalent) call are lost. For more info see the nexus manual.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > Rollback Method

25.1.112.3.36 StartTransaction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.StartTransaction Method

[TnxDatabase Class](#) | Related Topics

Pascal

```
public procedure StartTransaction(
  aSnapshot: Boolean = False
);
```

Parameters

Parameters	Description
aSnapshot	if true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

StartTransaction tries to start a transaction. If it fails an exception is triggered.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > StartTransaction Method

25.1.112.3.37 StartTransactionWith Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDatabase.StartTransactionWith Method

[TnxDatabase Class](#) | Related Topics

Pascal

```
public procedure StartTransactionWith(
  const aTables: array of TnxTable;
  aSnapshot: Boolean = False
);
```

Parameters

Parameters	Description
aTables	the tables that should be part of the transaction
aSnapShot	if true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

Start a transaction if an exclusive lock can be granted on all specified tables. If the grant fails one or more tables or an error occurs, an exception is raised.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > StartTransactionWith Method

25.1.112.3.38 StartTransactionWithEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.StartTransactionWithEx Method

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public function StartTransactionWithEx(
    const aTables: array of TnxTable;
    aSnapShot: Boolean = False
): TnxResult;
```

Parameters

Parameters	Description
aTables	the tables that should be part of the transaction
aSnapShot	if true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

Start a transaction if an exclusive lock can be granted on all specified tables. If the grant fails one or more tables or an error occurs, an error code is returned.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > StartTransactionWithEx Method

25.1.112.3.39 TableExists Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDatabase.TableExists Method**[TnxDatabase Class](#)**Pascal**

```
public function TableExists(
  const aTableName: string;
  const aPassword: string
): Boolean;
```

Parameters**Parameters**

aTableName

Description

table name

Remarks

This function returns true if a table with the given name exists otherwise false.

See Also[TnxDatabase Class](#)*You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > TableExists Method*

25.1.112.3.40 TransactionCorrupted Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDatabase.TransactionCorrupted Method**[TnxDatabase Class](#) | [Related Topics](#)**Pascal**

```
public procedure TransactionCorrupted;
```

Remarks

This method marks the transaction (level) as corrupted and thus preventing it from ever being committed. For more info see the nexus manual.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also[TnxDatabase Class](#)[Related Topics](#)*You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > TransactionCorrupted Method*

25.1.112.3.41 TryStartTransaction Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDatabase.TryStartTransaction Method**[TnxDatabase Class](#) | [Related Topics](#)**Pascal**

```
public function TryStartTransaction(
  aSnapshot: Boolean = False
```

```
    ): Boolean;
```

Parameters**Parameters**

aSnapShot

Description

if true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

Tries start a transaction (if none was started by the current TnxDatabase), returns true if a new transaction was started, false if a transaction was already started by this database instance or triggers an exception if an error occurs.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Methods > TryStartTransaction Method

25.1.112.Properties

25.1.112.4.1 AliasName Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.AliasName Property

[TnxDatabase Class](#) | Related Topics

Pascal

```
published property AliasName: string;
```

Remarks

The alias name the database is connected to. Either AliasName or AliasPath **must** be set.

Related Topics

[AliasPath](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > AliasName Property

25.1.112.4.2 AliasPath Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.AliasPath Property

[TnxDatabase Class](#) | Related Topics

Pascal

```
published property AliasPath: string;
```

Remarks

The (local server) path the database is connected to. Either AliasName or AliasPath **must** be set.

Related Topics[AliasName](#)**See Also**[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > AliasPath Property

25.1.112.4.3 DataSetCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.DataSetCount Property

[TnxDatabase Class](#) | Related Topics

Pascal

```
public property DataSetCount: Integer;
```

Remarks

Returns the number of dataset instances attached to this database.

Related Topics[DataSetCount](#), [DataSets](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)**See Also**[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > DataSetCount Property

25.1.112.4.4 DataSets Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.DataSets Property

[TnxDatabase Class](#) | Related Topics

Pascal

```
public property DataSets [aInx: Integer]: TnxDataset;
```

Remarks

Returns the dataset with index aInx out of the list of attached dataset instances.

Related Topics[DataSetCount](#), [DataSets](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)**See Also**[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > DataSets Property

25.1.112.4.5 Default Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.Default Property

[TnxDatabase Class](#)**Pascal**

```
published property Default: Boolean;
```

Remarks

If true this Database is the default database for it's session.

See Also[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > Default Property

25.1.112.4.6 Exclusive Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDatabase.Exclusive Property**[TnxDatabase Class](#)**Pascal**

```
published property Exclusive: Boolean;
```

Remarks

If a database is exclusively opened, only ONE client has access at a time.

See Also[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > Exclusive Property

25.1.112.4.7 FailSafe Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDatabase.FailSafe Property**[TnxDatabase Class](#)**Pascal**

```
published property FailSafe: Boolean;
```

Remarks

Sets failsafe mode on or off. For more info see NexusDB manual.

See Also[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > FailSafe Property

25.1.112.4.8 Implicit Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDatabase.Implicit Property**[TnxDatabase Class](#)**Pascal**

```
public property Implicit: Boolean;
```

Remarks

Implicit is true if the database instance was create implicitly, e.g by using tnxTable directly without assigning it to a database.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > Implicit Property

25.1.112.4.9 InTransaction Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.InTransaction Property

[TnxDatabase Class](#) | Related Topics

Pascal

```
public property InTransaction: Boolean;
```

Remarks

Returns true if a transaction for the current database is active otherwise false.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxDatabase Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > InTransaction Property

25.1.112.4.10 Path Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.Path Property

[TnxDatabase Class](#)

Pascal

```
public property Path: string;
```

Remarks

Returns the file system path of this database.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > Path Property

25.1.112.4.11 Queries Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.Queries Property

[TnxDatabase Class](#) | Related Topics

Pascal

```
public property Queries [aIdx: Integer]: TnxStatementDataSet;
```

Remarks

Returns the query with index aIdx out of the list of attached query instances.

Related Topics

[DataSetCount](#), [DataSets](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > Queries Property

25.1.112.4.12 QueryCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.QueryCount Property

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public property QueryCount: Integer;
```

Remarks

Returns the number of query instances attached to the database.

Related Topics

[DataSetCount](#), [DataSets](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)

See Also

[TnxDatabase Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > QueryCount Property

25.1.112.4.13 ReadOnly Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.ReadOnly Property

[TnxDatabase Class](#)

Pascal

```
published property ReadOnly: Boolean;
```

Remarks

With this property a database can be set to read only.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > ReadOnly Property

25.1.112.4.14 TableCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.TableCount Property

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public property TableCount: Integer;
```

Remarks

Returns the number of table instances attached to the database.

Related Topics

[DataSetCount](#), [DataSets](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)

See Also

[TnxDatabase Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > TableCount Property

25.1.112.4.15 Tables Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.Tables Property

[TnxDatabase Class](#) | [Related Topics](#)

Pascal

```
public property Tables [aInx: Integer]: TnxTable;
```

Remarks

Returns the table with index aInx out of the list of attached table instances.

Related Topics

[DataSetCount](#), [DataSets](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)

See Also

[TnxDatabase Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > Tables Property

25.1.112.4.16 TransContext Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabase.TransContext Property

[TnxDatabase Class](#)

Pascal

```
published property TransContext: TnxTransContext;
```

Remarks

The transaction context the database is attached to. This property can be nil.

See Also

[TnxDatabase Class](#)

You are here: Symbol Reference > Classes > TnxDatabase Class > Properties > TransContext Property

25.1.11.TnxDataDictionary Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataDictionary Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxDataDictionary = class(TnxBaseTableDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

The data dictionary is the heart of every NexusDB table. It describes the files, fields, indices, etc and even how to read/write the records.

If you don't need the fine control that the data dictionary gives you, you can just use TIndexDefs, TFieldDefs and the CreateTable method of TDataSet.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class

25.1.113. Constructors

25.1.113.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataDictionary.Create Constructor

[TnxDataDictionary Class](#)

Pascal

```
public constructor Create; reintroduce;
```

Remarks

constructor.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Constructors > Create Constructor

25.1.113. Destructors

25.1.113.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataDictionary.Destroy Destructor

[TnxDataDictionary Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Destructors > Destroy Destructor

25.1.113. Methods

25.1.113.3.1 AddStreamDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataDictionary.AddStreamDescriptor Method

[TnxDataDictionary Class](#) | [Related Topics](#)

Pascal

```
public function AddStreamDescriptor(
  aStreamClass: TnxBaseStreamDescriptorClass = nil
): TnxBaseStreamDescriptor;
```

Remarks

This function sets the Stream Descriptor class to the given aStreamClass, creates and instance, adds it to the data dictionary and returns the instance.

Related Topics

[AddStreamDescriptor](#), [RemoveStreamDescriptor](#), [StreamDescriptor](#)

See Also

[TnxDataDictionary Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > [AddStreamDescriptor Method](#)

25.1.113.3.2 Assign Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataDictionary.Assign Method

[TnxDataDictionary Class](#)

Pascal

```
public procedure Assign(
  Source: TPersistent
); override;
```

Parameters

Parameters	Description
Source	the source Data Dictionary to copy the settings from.

Remarks

This methods copies the settings of the given Source to the current Data Dictionary, if Source is of the same type.

See Also

[TnxDataDictionary Class](#)[You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > Assign Method](#)

25.1.113.3.3 AssignOnlyFields Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataDictionary.AssignOnlyFields Method**[TnxDataDictionary Class](#)**Pascal**

```
public procedure AssignOnlyFields(
    aSource: TnxBaseTableDescriptor
); override;
```

Remarks

This is AssignOnlyFields, a member of class TnxDataDictionary.

See Also[TnxDataDictionary Class](#)[You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > AssignOnlyFields Method](#)

25.1.113.3.4 CheckReadOnly Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataDictionary.CheckReadOnly Method**[TnxDataDictionary Class](#)**Pascal**

```
public procedure CheckReadOnly; override;
```

Remarks

Checks if the Data Dictionary is read-only. Raises an exception if true.

See Also[TnxDataDictionary Class](#)[You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > CheckReadOnly Method](#)

25.1.113.3.5 CheckValid Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataDictionary.CheckValid Method**[TnxDataDictionary Class](#)**Pascal**

```
public procedure CheckValid(
    aForBuild: Boolean
); virtual; reintroduce;
```

Remarks

Checks if the Data Dictionary is valid. Raises an exception if not.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > CheckValid Method

25.1.113.3.6 FindRelatedDescriptorOfType Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxDataDictionary.FindRelatedDescriptorOfType Method

TnxDataDictionary Class

Pascal

```
public function FindRelatedDescriptorOfType(
  aType: TnxDictionaryItemClass;
  out aDescriptor;
  aFrom: TnxDictionaryItem = nil
): Boolean; override;
```

Remarks

This is FindRelatedDescriptorOfType, a member of class TnxDataDictionary.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > FindRelatedDescriptorOfType Method

25.1.113.3.7 GetFileFromExt Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxDataDictionary.GetFileFromExt Method

TnxDataDictionary Class

Pascal

```
public function GetFileFromExt(
  const aFileExt: string
): Integer;
```

Parameters

Parameters	Description
aFileExt	the file extension (attached to the base name) to look for.

Remarks

Return the file number for a given file extension, or -1 if not found.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > GetFileFromExt Method

25.1.113.3.8 IsEqual Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxDataDictionary.AreEqual Method

TnxDataDictionary Class

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Parameters**Parameters**

aDataDictionary

Description

the Data Dictionary to compare the current one to.

Remarks

Compares the current Data Dictionary to another. Returns true if they are equal.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > IsEqual Method

25.1.113.3.9 MakeReadOnly Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataDictionary.MakeReadOnly Method

[TnxDataDictionary Class](#)

Pascal

```
public procedure MakeReadOnly;
```

Remarks

marks the dictionary as readonly

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > MakeReadOnly Method

25.1.113.3.10 ReadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataDictionary.ReadFromReader Method

[TnxDataDictionary Class](#)

Pascal

```
public procedure ReadFromReader(
    aReader: TReader
);
```

Remarks

Reads the whole Data Dictionary settings (data) from aReader.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > ReadFromReader Method

25.1.113.3.11 ReadFromStream Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataDictionary.ReadFromStream Method**[TnxDataDictionary Class](#)**Pascal**

```
public procedure ReadFromStream(
    aStream: TStream
);
```

Remarks

Reads the whole Data Dictionary settings (data) from aStream.

See Also[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > ReadFromStream Method

25.1.113.3.12 RemoveStreamDescriptor Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataDictionary.RemoveStreamDescriptor Method**[TnxDataDictionary Class](#) | [Related Topics](#)**Pascal**

```
public procedure RemoveStreamDescriptor;
```

Remarks

This function frees the Instance of the associated Stream Descriptor.

Related Topics[AddStreamDescriptor](#), [RemoveStreamDescriptor](#), [StreamDescriptor](#)**See Also**[TnxDataDictionary Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > RemoveStreamDescriptor Method

25.1.113.3.13 UsableBlockSize Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataDictionary.UsableBlockSize Method**[TnxDataDictionary Class](#)**Pascal**

```
public function UsableBlockSize(
    aBlockSize: TnxBlockSize
): TnxWord32; virtual;
```

Parameters**Parameters**

aBlockSize

Description

block size the calculation should be based on.

Remarks

This function calculates the number of bytes that can be used for storing data per block by subtracting the header size from the block size in bytes

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > UsableBlockSize Method

25.1.113.3.14 WriteToStream Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxDataDictionary.WriteToStream Method

[TnxDataDictionary Class](#)

Pascal

```
public procedure WriteToStream(
  aStream: TStream;
  aClientVersion: Integer
);
```

Remarks

Writes the whole Data Dictionary settings (data) to aStream.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > WriteToStream Method

25.1.113.3.15 WriteToWriter Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxDataDictionary.WriteToWriter Method

[TnxDataDictionary Class](#)

Pascal

```
public procedure WriteToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
);
```

Remarks

Writes the whole Data Dictionary settings (data) to aWriter.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Methods > WriteToWriter Method

25.1.113.4 Properties

25.1.113.4.1 FilesDescriptor Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataDictionary.FilesDescriptor Property

TnxDataDictionary Class

Pascal

```
public property FilesDescriptor: TnxFileDescriptor;
```

Remarks

Returns the FilesDescriptor for this dictionary.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Properties > FilesDescriptor Property

25.1.113.4.2 StreamDescriptor Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataDictionary.StreamDescriptor Property

[TnxDataDictionary Class](#) | [Related Topics](#)

Pascal

```
public property StreamDescriptor: TnxBaseStreamDescriptor;
```

Remarks

This property returns the associated Stream Descriptor.

Related Topics

[AddStreamDescriptor](#), [RemoveStreamDescriptor](#), [StreamDescriptor](#)

See Also

[TnxDataDictionary Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Properties > StreamDescriptor Property

25.1.113.4.3 Version Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataDictionary.Version Property

[TnxDataDictionary Class](#)

Pascal

```
public property Version: Integer;
```

Remarks

Returns the Version of the Data Dictionary implementation and thus the table.

See Also

[TnxDataDictionary Class](#)

You are here: Symbol Reference > Classes > TnxDataDictionary Class > Properties > Version Property

25.1.11 TnxDataMessageReader Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataMessageReader Class

[nxllTransport](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxDataMessageReader = class(TnxReader);
```

File

[nxllTransport](#)

Remarks

A TReader descendant for easily reading the data of a message.

See Also

[nxllTransport](#)

[Members](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxDataMessageReader Class

25.1.114 Constructors

25.1.114.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataMessageReader.Create Constructor

TnxDataMessageReader Class

Pascal

```
public constructor Create(
  const aDataMessage: TnxDataMessage
);
```

Parameters

Parameters

aDataMessage

Description

the Message used.

Remarks

constructor.

See Also

[TnxDataMessageReader Class](#)

You are here: Symbol Reference > Classes > TnxDataMessageReader Class > Constructors > Create Constructor

25.1.114.Destructors

25.1.114.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataMessageReader.Destroy Destructor

[TnxDataMessageReader Class](#)

```
Pascal
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxDataMessageReader Class](#)

You are here: Symbol Reference > Classes > TnxDataMessageReader Class > Destructors > Destroy Destructor

25.1.114.Properties

25.1.114.3.1 DataMessageStream Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataMessageReader.DataMessageStream Property[TnxDataMessageReader Class](#)

```
Pascal
public property DataMessageStream: TStream;
```

Remarks

A stream for reading/writing message data.

See Also

[TnxDataMessageReader Class](#)

You are here: Symbol Reference > Classes > TnxDataMessageReader Class > Properties > DataMessageStream Property

25.1.115.TnxDataStream Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataStream Class
[nxllTransport | Members](#)

```
Pascal
public TnxDataStream = class(TnxStaticMemoryStream);
```

File

[nxllTransport](#)

Remarks

A stream for reading/writing message data.

See Also

[nxllTransport](#)
[Members](#)

You are here: Symbol Reference > Classes > TnxDataStream Class

25.1.115. Constructors

25.1.115.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataMessageStream.Create Constructor

TnxDataMessageStream Class

Pascal

```
public constructor Create(
    const aDataMessage: TnxDataMessage
);
```

Parameters

Parameters

aDataMessage

Description

the Message used.

Remarks

constructor.

See Also

[TnxDataMessageStream Class](#)

You are here: Symbol Reference > Classes > TnxDataMessageStream Class > Constructors > Create Constructor

25.1.116 TnxDataset Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset Class

[Events](#) | [nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxDataset = class(TDataSet);
```

File

nxdb

Remarks

TnxDataset is the base class for all TDataset compatible Nexus components. Most methods and properties work exactly the same as Borland's implementation.

See Also

[Events](#)

[nxdb](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxDataset Class

25.1.116. Constructors

25.1.116.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.Create Constructor

TnxDataset Class

```
Pascal
public constructor Create(
    AOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Constructors > Create Constructor

25.1.116.:Destructors

25.1.116.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.Destroy Destructor**TnxDataset Class**

```
Pascal
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Destructors > Destroy Destructor

25.1.116.:Methods

25.1.116.3.1 AddFileBlob Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.AddFileBlob Method**TnxDataset Class**

```
Pascal
public procedure AddFileBlob(
    aField: TField;
    const aFileName: string
);
```

Parameters

Parameters	Description
aField	the blob field you want the file associated with
aFileName	the file name

Remarks

AddFileBlob adds a blob linked to **real** file. Please keep in mind that the file and it's name are server based and that NexusDB is not responsible for this file.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > AddFileBlob Method

25.1.116.3.2 AddFileBlobEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.AddFileBlobEx Method

[TnxDataset Class](#)

Pascal

```
public function AddFileBlobEx(
  aField: TField;
  const aFileName: string
): TnxResult;
```

Parameters

Parameters	Description
aField	the blob field you want the file associated with
aFileName	the file name

Remarks

AddFileBlob adds a blob linked to **real** file. Please keep in mind that the file and it's name are server based and that NexusDB is not responsible for this file.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > AddFileBlobEx Method

25.1.116.3.3 AfterConstruction Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.AfterConstruction Method

[TnxDataset Class](#)

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

This is AfterConstruction, a member of class TnxDataset.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > AfterConstruction Method

25.1.116.3.4 BatchPost Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.BatchPost Method

[TnxDataset Class](#) | Related Topics

Pascal

```
public procedure BatchPost;
```

Remarks

Only call BatchPost in batch append mode. It copies the current record buffer into the internal batch append buffer and posts the internal batch append buffer if its size is bigger than aCutoffSize defined in BeginBatchAppend.

Related Topics

[FlushBatchAppend](#), [EndBatchAppend](#), [BeginBatchAppend](#), [BatchPost](#), [InBatchAppend](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > BatchPost Method

25.1.116.3.5 BeginBatchAppend Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.BeginBatchAppend Method

TnxDataset Class

Pascal

```
public procedure BeginBatchAppend(
  aCutoffSize: Integer
);
```

Parameters

Parameters	Description
aCutoffSize	the size in bytes when a batch is send to the server.

Remarks

Use BeginBatchAppend to start batch append mode. In this mode the only allowed actions are: setting field values, BatchPost, EndBatchAppend.

Please note that batch writing to blob fields is fully supported!

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > BeginBatchAppend Method

25.1.116.3.6 BookmarkValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.BookmarkValid Method

TnxDataset Class

Pascal

```
public function BookmarkValid(
  aBookmark: TBookmark
): Boolean; override;
```

Remarks

see [TDataSet.BookmarkValid](#)

See Also[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Methods > BookmarkValid Method

25.1.116.3.7 CompareBookmarks Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.CompareBookmarks Method**[TnxDataset Class](#)**Pascal**

```
public function CompareBookmarks(
  aBookmark1: TBookmark;
  aBookmark2: TBookmark
): Integer; override;
```

Remarkssee [TDataSet.CompareBookmarks](#)**See Also**[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Methods > CompareBookmarks Method

25.1.116.3.8 CopyRecords Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.CopyRecords Method**[TnxDataset Class](#)**Pascal**

```
public procedure CopyRecords(
  aSrcTable: TnxDataset;
  aCopyBLOBS: Boolean;
  aMaxTransSize: Integer = 0
);
```

Parameters

Parameters	Description
aSrcTable	the dataset the records are coming from. apply filters/ranges if you want to restrict the records
aCopyBLOBS	set to true if you want to copy the blob fields too
aMaxTransSize	number of dirty mem blocks to copy in one transaction

Remarks

The method copies all visible records from the source dataset to the current table.

It correctly uses the ranges/filters on the source table.

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > CopyRecords Method

25.1.116.3.9 CopyRecordsEx Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxDataset.CopyRecordsEx Method

TnxDataset Class

Pascal

```
public procedure CopyRecordsEx(
  aSrcTable: TnxDataset;
  aCopyBLOBS: Boolean;
  aMaxTransSize: Integer = 0;
  aCheckValFields: Boolean = True
);
```

Parameters

Parameters	Description
aSrcTable	the dataset the records are coming from. apply filters/ranges if you want to restrict the records
aCopyBLOBS	set to true if you want to copy the blob fields too
aMaxTransSize	number of dirty mem blocks to copy in one transaction
aCheckValFields	Wether or not to enforce field validation checks

Remarks

The method copies all visible records from the source dataset to the current table.

It correctly uses the ranges/filters on the source table.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > CopyRecordsEx Method

25.1.116.3.10 CreateBlobStream Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxDataset.CreateBlobStream Method

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
public function CreateBlobStream(
  aField: TField;
  aMode: TBlobStreamMode
): TStream; override;
```

Remarks

This function creates the right type of blob stream based on the state of the dataset.

The blob stream should be freed again as soon as possible. It must be freed before calling Post or moving to another record.

Related Topics

TDataSet.CreateBlobStream TnxBaseBlobStream TnxBlobStream TnxBlockModeBlobStream
TnxBatchAppendBlobStream

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > CreateBlobStream Method

25.1.116.3.11 DeleteRecords Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.DeleteRecords Method

[TnxDataset Class](#)

Pascal

```
public procedure DeleteRecords;
```

Remarks

This method deletes **all** visible records of the current table. You can use filters/ranges to restrict the functionality to a subset of records.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > DeleteRecords Method

25.1.116.3.12 DeleteStream Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.DeleteStream Method

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
public procedure DeleteStream(
    const aName: string
);
```

Parameters

Parameters	Description
aName	Name of the stream

Remarks

Use this to delete the streams.

NEVER EVER DELETE THE DICT STREAM!

Related Topics

[GetStreamList](#), [WriteStream](#), [ReadStream](#), [DeleteStream](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > DeleteStream Method

25.1.116.3.13 EndBatchAppend Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataset.EndBatchAppend Method**[TnxDataset Class](#) | Related Topics**Pascal**

```
public procedure EndBatchAppend;
```

Remarks

This method terminates the batch append mode. It sends the internal batch append buffer to the server if required.

Related Topics

[FlushBatchAppend](#), [EndBatchAppend](#), [BeginBatchAppend](#), [BatchPost](#), [InBatchAppend](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Methods](#) > [EndBatchAppend Method](#)

25.1.116.3.14 Exists Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataset.Exists Method**[TnxDataset Class](#)**Pascal**

```
public function Exists: Boolean; virtual;
```

Remarks

Exists returns true if this table exists, in the given session/database context.

It can be called without opening the table.

See Also

[TnxDataset Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Methods](#) > [Exists Method](#)

25.1.116.3.15 FlushBatchAppend Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataset.FlushBatchAppend Method**[TnxDataset Class](#) | Related Topics**Pascal**

```
public procedure FlushBatchAppend;
```

Parameters**Parameters**

aCutoffSize

Description

the size in bytes when a batch is send to the server.

Remarks

Only call in batch append mode. It forces the dataset to post the current internal batch append buffer.

Related Topics

[FlushBatchAppend](#), [EndBatchAppend](#), [BeginBatchAppend](#), [BatchPost](#), [InBatchAppend](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > FlushBatchAppend Method

25.1.116.3.16 GetAutoIncValue Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.GetAutoIncValue Method

[TnxDataset Class](#) | Related Topics

Pascal

```
public procedure GetAutoIncValue(
    out aValue: TnxWord32
);
```

Parameters

Parameters	Description
aValue	holds the highest AutoInc value after the method returns

Remarks

GetAutoincValue gets the last used value of the AutoInc field of this dataset.

this function works even if there is no autoinc field declared in the table, because Nexus always tracks a "highest autoinc" value in the header.

Related Topics

[SetAutoIncValue](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > GetAutoIncValue Method

25.1.116.3.17 GetBookmark Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.GetBookmark Method

[TnxDataset Class](#)

Pascal

```
public function GetBookmark: TBookmark; override;
```

Remarks

see [TDataSet.GetBookmark](#)

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > GetBookmark Method

25.1.116.3.18 GetCurrentRecord Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.GetCurrentRecord Method

TnxDataset Class

Pascal

```
public function GetCurrentRecord(
    aBuffer: PChar
): Boolean; override;
```

Remarks

see [TDataSet.GetCurrentRecord](#)

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > GetCurrentRecord Method

25.1.116.3.19 GetFieldData Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.GetFieldData Method

TnxDataset Class

Pascal

```
public function GetFieldData(
    aField: TField;
    aBuffer: Pointer
): Boolean; override;
```

Remarks

see [TDataSet.GetFieldData](#)

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > GetFieldData Method

25.1.116.3.20 GetStreamList Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.GetStreamList Method

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
public procedure GetStreamList(
    aStreams: TStrings
);
```

Parameters

Parameters	Description
------------	-------------

aStreams

string list that will hold the names; must not be nil

Remarks

GetStreamList returns a list of the "table streams" saved in the table. A "table stream" is a delphi stream of system or user defined data that can be saved with a table. All tables contain a stream called DICT which contain the data dictionary.

Related Topics

[GetStreamList](#), [WriteStream](#), [ReadStream](#), [DeleteStream](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > GetStreamList Method

25.1.116.3.21 GotoCurrent Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.GotoCurrent Method

[TnxDataset Class](#)

Pascal

```
public procedure GotoCurrent(
    aDataSet: TnxDataset
) ; dynamic;
```

Parameters**Parameters**

aDataSet

Description

the dataset the current one is synchronized with

Remarks

GotoCurrent synchronizes a dataset with the given one or in other words positions the dataset on the same record as aDataSet. This method will result in an error if that record is currently not visible (range/filter) or if aDataSet is not positioned on a valid record

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > GotoCurrent Method

25.1.116.3.22 InBatchAppend Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.InBatchAppend Method

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
public function InBatchAppend: Boolean;
```

Remarks

This function returns true if a Batch Append is open.

Related Topics

[FlushBatchAppend](#), [EndBatchAppend](#), [BeginBatchAppend](#), [BatchPost](#), [InBatchAppend](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > InBatchAppend Method

25.1.116.3.23 IsRecordLocked Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.IsRecordLocked Method

[TnxDataset Class](#)

Pascal

```
public function IsRecordLocked: TnxLockPresent;
```

Remarks

Returns the current Record's lock status.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > IsRecordLocked Method

25.1.116.3.24 IsSequenced Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.IsSequenced Method

[TnxDataset Class](#)

Pascal

```
public function IsSequenced: Boolean; override;
```

Remarks

see TDataSet.IsSequenced

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > IsSequenced Method

25.1.116.3.25 IsTableLocked Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.IsTableLocked Method

[TnxDataset Class](#)

Pascal

```
public function IsTableLocked(
  aLockType: TnxLockType
): TnxLockPresent;
```

Remarks

Returns the state of a lock of the certain type is active on the current table

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > IsTableLocked Method

25.1.116.3.26 KeyAsVariant Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.KeyAsVariant Method

TnxDataset Class

Pascal

```
public function KeyAsVariant: Variant; virtual;
```

Remarks

Returns a key for the current record.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > KeyAsVariant Method

25.1.116.3.27 Locate Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.Locate Method

TnxDataset Class

Pascal

```
public function Locate(
  const aKeyFields: string;
  const aKeyValue: Variant;
  aOptions: TLocateOptions
): Boolean; override;
```

Remarks

see [TDataSet.Locate](#)

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > Locate Method

25.1.116.3.28 LockTable Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.LockTable Method

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
public procedure LockTable(
  aLockType: TnxLockType
);
```

Parameters

Parameters	Description
aLockType	the type of lock you want to acquire

Remarks

LockTable locks the dataset for the given operation. Make sure to keep these locks as short as possible, because it might otherwise interfere with the functionality/access of other instances to the table.

These locks interact directly with record locks, e.g. you can't acquire a table lock while anyone else has a record lock.

Related Topics

[UnlockTable](#), [UnlockTableAll](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > LockTable Method

25.1.116.3.29 Lookup Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.Lookup Method

[TnxDataset Class](#)

Pascal

```
public function Lookup(
    const aKeyFields: string;
    const aKeyValue: Variant;
    const aResultFields: string
): Variant; override;
```

Remarks

see [TDataSet.Lookup](#)

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > Lookup Method

25.1.116.3.30 Post Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.Post Method

[TnxDataset Class](#)

Pascal

```
public procedure Post; override;
```

Remarks

== virtual [TDataSet](#) methods == -- navigation

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > Post Method

25.1.116.3.31 PSEndTransaction Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataset.PSEndTransaction Method**[TnxDataset Class](#)**Pascal**

```
public procedure PSEndTransaction(
    Commit: Boolean
); override;
```

Remarkssee [TDataset.PSEndTransaction](#)**See Also**[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Methods > PSEndTransaction Method

25.1.116.3.32 PSEExecuteStatement Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataset.PSEExecuteStatement Method**[TnxDataset Class](#)**Pascal**

```
public function PSEExecuteStatement(
    const ASql: string;
    AParams: TParams;
    ResultSet: Pointer = nil
): Integer; override;
```

Remarkssee [TDataset.PSEExecuteStatement](#)**See Also**[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Methods > PSEExecuteStatement Method

25.1.116.3.33 PSGetQuoteChar Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxDataset.PSGetQuoteChar Method**[TnxDataset Class](#)**Pascal**

```
public function PSGetQuoteChar: string; override;
```

Remarkssee [TDataset.PSGetQuoteChar](#)**See Also**[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Methods > PSGetQuoteChar Method

25.1.116.3.34 PSInTransaction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.PSInTransaction Method

TnxDataset Class

Pascal

```
public function PSInTransaction: Boolean; override;
```

Remarks

see TDataset.PSInTransaction

See Also

[TnxDataset Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Methods](#) > [PSInTransaction Method](#)

25.1.116.3.35 PSIsSqlBased Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.PSIsSqlBased Method

TnxDataset Class

Pascal

```
public function PSIsSqlBased: Boolean; override;
```

Remarks

see TDataset.PSIsSqlBased

See Also

[TnxDataset Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Methods](#) > [PSIsSqlBased Method](#)

25.1.116.3.36 PSIsSqlSupported Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.PSIsSqlSupported Method

TnxDataset Class

Pascal

```
public function PSIsSqlSupported: Boolean; override;
```

Remarks

see TDataset.PSIsSqlSupported

See Also

[TnxDataset Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Methods](#) > [PSIsSqlSupported Method](#)

25.1.116.3.37 PSReset Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.PSReset Method

TnxDataset Class

Pascal

```
public procedure PSReset; override;
```

Remarks

see TDataset.PSReset

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > PSReset Method

25.1.116.3.38 PSStartTransaction Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.PSStartTransaction Method

[TnxDataset Class](#)

Pascal

```
public procedure PSStartTransaction; override;
```

Remarks

see TDataset.PSStartTransaction

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > PSStartTransaction Method

25.1.116.3.39 ReadStream Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.ReadStream Method

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
public procedure ReadStream(
    const aName: string;
    aStream: TStream
);
```

Parameters

Parameters	Description
aName	Name of the Stream
aStream	the stream to read into; must not be nil

Remarks

Use this method to read a certain stream. It raises exception if not the stream is not found.

Related Topics

[GetStreamList](#), [WriteStream](#), [ReadStream](#), [DeleteStream](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > ReadStream Method

25.1.116.3.40 ReadStreamEx Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.ReadStreamEx Method

[TnxDataset Class](#)

Pascal

```
public function ReadStreamEx(
  const aName: string;
  aStream: TStream
): TnxResult;
```

Remarks

This is ReadStreamEx, a member of class TnxDataset.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > ReadStreamEx Method

25.1.116.3.41 RecordCountAsync Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.RecordCountAsync Method

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
public procedure RecordCountAsync(
  out aTaskInfo: TnxAbstractTaskInfo
);
```

Parameters

Parameters	Description
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.

Remarks

This method starts an asynchronous RecordCount. Very useful if there's for example a filter applied and the server needs to iterate through all records to find out the count. Use TaskInfo.GetStatus to find out the status of the operation.

Related Topics

[TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > RecordCountAsync Method

25.1.116.3.42 SetAutoIncValue Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDataset.SetAutoIncValue Method**[TnxDataset Class](#) | [Related Topics](#)**Pascal**

```
public procedure SetAutoIncValue(
  aValue: TnxWord32
);
```

Parameters**Parameters**

aValue

Description

the value you want to the AutoInc value to

Remarks

Use this method to set the AutoInc field's value; be very careful with this because it might break referential integrity based on these values and also might lead to posting errors due to unique field violations.

Related Topics[GetAutoIncValue](#)**See Also**
[TnxDataset Class](#)
[Related Topics](#)
You are here: Symbol Reference > Classes > TnxDataset Class > Methods > SetAutoIncValue Method

25.1.116.3.43 UnlockTable Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDataset.UnlockTable Method**[TnxDataset Class](#) | [Related Topics](#)**Pascal**

```
public procedure UnlockTable(
  aLockType: TnxLockType
);
```

Parameters**Parameters**

aLockType

Description

the type of lock you want to release

Remarks

Remove the given lock from the dataset with this method.

Related Topics[LockTable](#), [UnlockTableAll](#)**See Also**
[TnxDataset Class](#)
[Related Topics](#)
You are here: Symbol Reference > Classes > TnxDataset Class > Methods > UnlockTable Method

25.1.116.3.44 UnlockTableAll Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDataset.UnlockTableAll Method**[TnxDataset Class](#) | [Related Topics](#)**Pascal**

```
public procedure UnlockTableAll;
```

Remarks

UnlockTableAll releases **all** cursor based locks of this dataset, but this does **not** include record locks.

Related Topics

[LockTable](#), [UnlockTable](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > UnlockTableAll Method

25.1.116.3.45 WriteStream Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxDataset.WriteStream Method**[TnxDataset Class](#) | [Related Topics](#)**Pascal**

```
public procedure WriteStream(
  const aName: string;
  aStream: TStream
);
```

Parameters

Parameters	Description
aName	Name of the Stream
aStream	the stream to write

Remarks

This method stores aStream in the table and overwrites any existing stream of that name.

NEVER EVER OVERWRITE THE DICT STREAM!

Related Topics

[GetStreamList](#), [WriteStream](#), [ReadStream](#), [DeleteStream](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Methods > WriteStream Method

25.1.116.4Events

25.1.116.4.1 OnServerFilterTimeout Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.OnServerFilterTimeout Property

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
published property OnServerFilterTimeout: TnxServerFilterTimeoutEvent;
```

Remarks

The event is triggered if the FilterTimeout time has passed since activating a filter. Set cancel to false to give the server another FilterTimeout period of time to finish the filtering.

Related Topics

[Filter](#), [Filtered](#), [FilterTimeout](#), [OnServerFilterTimeout](#), [FilterOptions](#)

See Also

[TnxDataset Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Events > OnServerFilterTimeout Property

25.1.116.4Properties

25.1.116.5.1 _Dictionary Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset._Dictionary Property

[TnxDataset Class](#)

Pascal

```
public property _Dictionary: TnxDataDictionary;
```

Remarks

The Dictionary holds all field and index definitions as well as full information on how to access and read the data file.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > _Dictionary Property

25.1.116.5.2 AbstractCursor Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.AbstractCursor Property

[TnxDataset Class](#)

Pascal

```
public property AbstractCursor: TnxAbstractCursor;
```

Remarks

This is a pointer to the current cursor of this dataset.

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AbstractCursor Property

25.1.116.5.3 ActiveDesigntime Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.ActiveDesigntime Property**[TnxDataset Class](#) | Related Topics**Pascal**

```
published property ActiveDesigntime: Boolean;
```

Remarks

Set ActiveDesigntime to true if the dataset should be automatically opened at design time. Very handy to use in combination with ActiveRuntime if you need the dataset open at design time, but sometimes forget to close it before creating a shipping version.

Related Topics[ActiveRuntime](#)**See Also**[TnxDataset Class](#)[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > ActiveDesigntime Property

25.1.116.5.4 ActiveRuntime Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.ActiveRuntime Property**[TnxDataset Class](#) | Related Topics**Pascal**

```
published property ActiveRuntime: Boolean;
```

Remarks

Set ActiveRuntime to true if the dataset should be automatically opened at runtime. Take care that this needs a fully set up chain of components otherwise you very likely get an "could not connect to server" error on application startup.

Related Topics[ActiveDesigntime](#)**See Also**[TnxDataset Class](#)[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > ActiveRuntime Property

25.1.116.5.5 AfterCancel Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.AfterCancel Property**

[TnxDataset Class](#)**Pascal**

```
published property AfterCancel;
```

Remarks

see TDataSet.AfterCancel.

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterCancel Property

25.1.116.5.6 AfterClose Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.AfterClose Property**[TnxDataset Class](#)**Pascal**

```
published property AfterClose;
```

Remarks

see TDataSet.AfterClose.

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterClose Property

25.1.116.5.7 AfterDelete Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.AfterDelete Property**[TnxDataset Class](#)**Pascal**

```
published property AfterDelete;
```

Remarks

see TDataSet.AfterDelete

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterDelete Property

25.1.116.5.8 AfterEdit Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.AfterEdit Property**[TnxDataset Class](#)**Pascal**

```
published property AfterEdit;
```

Remarks

see TDataSet.AfterEdit

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterEdit Property

25.1.116.5.9 AfterInsert Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.AfterInsert Property

[TnxDataset Class](#)

Pascal

```
published property AfterInsert;
```

Remarks

see TDataSet.AfterInsert

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterInsert Property

25.1.116.5.10 AfterOpen Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.AfterOpen Property

[TnxDataset Class](#)

Pascal

```
published property AfterOpen;
```

Remarks

see TDataSet.AfterOpen

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterOpen Property

25.1.116.5.11 AfterPost Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.AfterPost Property

[TnxDataset Class](#)

Pascal

```
published property AfterPost;
```

Remarks

see TDataSet.AfterPost

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterPost Property

25.1.116.5.12 AfterRefresh Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.AfterRefresh Property

TnxDataset Class

Pascal

```
published property AfterRefresh;
```

Remarks

see TDataSet.AfterRefresh

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterRefresh Property

25.1.116.5.13 AfterScroll Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.AfterScroll Property

TnxDataset Class

Pascal

```
published property AfterScroll;
```

Remarks

see TDataSet.AfterScroll

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AfterScroll Property

25.1.116.5.14 AliasName Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.AliasName Property

TnxDataset Class

Pascal

```
published property AliasName: string;
```

Remarks

AliasName is a quick way to access an alias without a connected tnxDatabase. AliasName does **NOT** support direct paths!

Please note that Session must be assigned for this to work.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AliasName Property

25.1.116.5.15 AutoCalcFields Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.AutoCalcFields Property

TnxDataset Class

Pascal

```
published property AutoCalcFields;
```

Remarks

see TDataSet.AutoCalcFields

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > AutoCalcFields Property

25.1.116.5.16 BeforeCancel Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.BeforeCancel Property

TnxDataset Class

Pascal

```
published property BeforeCancel;
```

Remarks

see TDataSet.BeforeCancel

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeCancel Property

25.1.116.5.17 BeforeClose Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.BeforeClose Property

TnxDataset Class

Pascal

```
published property BeforeClose;
```

Remarks

see TDataSet.BeforeClose

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeClose Property

25.1.116.5.18 BeforeDelete Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.BeforeDelete Property

TnxDataset Class

Pascal

```
published property BeforeDelete;
```

Remarks

see TDataSet.BeforeDelete

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeDelete Property

25.1.116.5.19 BeforeEdit Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.BeforeEdit Property

[TnxDataset Class](#)

Pascal

```
published property BeforeEdit;
```

Remarks

see TDataSet.BeforeEdit

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeEdit Property

25.1.116.5.20 BeforeInsert Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.BeforeInsert Property

[TnxDataset Class](#)

Pascal

```
published property BeforeInsert;
```

Remarks

see TDataSet.BeforeInsert

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeInsert Property

25.1.116.5.21 BeforeOpen Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.BeforeOpen Property

[TnxDataset Class](#)

Pascal

```
published property BeforeOpen;
```

Remarks

see TDataSet.BeforeOpen

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeOpen Property

25.1.116.5.22 BeforePost Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.BeforePost Property**[TnxDataset Class](#)**Pascal**

```
published property BeforePost;
```

Remarks

see TDataSet.BeforePost

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforePost Property

25.1.116.5.23 BeforeRefresh Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.BeforeRefresh Property**[TnxDataset Class](#)**Pascal**

```
published property BeforeRefresh;
```

Remarks

see TDataSet.BeforeRefresh

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeRefresh Property

25.1.116.5.24 BeforeScroll Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.BeforeScroll Property**[TnxDataset Class](#)**Pascal**

```
published property BeforeScroll;
```

Remarks

see TDataSet.BeforeScroll

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BeforeScroll Property

25.1.116.5.25 BlockReadOptions Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.BlockReadOptions Property

TnxDataset Class

Pascal

```
published property BlockReadOptions: TnxRecordGetBatchExOptions;
```

Remarks

Use BlockReadOptions to set how block/batch operations work.

If gboBookmarks is true, bookmarks are received from the server as well as records. The Borland's implementation in TTable doesn't support bookmarks during batch reads.

If gboBlob is true, blobs are processed in the batch modes.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > BlockReadOptions Property

25.1.116.5.26 Database Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.Database Property

TnxDataset Class

Pascal

```
published property Database: TnxDatabase;
```

Remarks

This is database the dataset is linked to. Either AliasName or Database must be set.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > Database Property

25.1.116.5.27 FieldsDescriptor Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.FieldsDescriptor Property

TnxDataset Class

Pascal

```
public property FieldsDescriptor: TnxFieldsDescriptor;
```

Remarks

Returns the Fields Descriptor of the current table.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > FieldsDescriptor Property

25.1.116.5.28 Filter Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.Filter Property

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
published property Filter;
```

Remarks

see TDataset.Filter

Related Topics

[Filter](#), [Filtered](#), [FilterTimeout](#), [OnServerFilterTimeout](#), [FilterOptions](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Properties](#) > [Filter Property](#)

25.1.116.5.29 Filtered Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.Filtered Property

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
published property Filtered;
```

Remarks

see TDataset.Filtered

Related Topics

[Filter](#), [Filtered](#), [FilterTimeout](#), [OnServerFilterTimeout](#), [FilterOptions](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Properties](#) > [Filtered Property](#)

25.1.116.5.30 FilterOptions Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.FilterOptions Property

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
published property FilterOptions;
```

Remarks

see TDataset.FilterOptions

Related Topics

[Filter](#), [Filtered](#), [FilterTimeout](#), [OnServerFilterTimeout](#), [FilterOptions](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > FilterOptions Property

25.1.116.5.31 FilterResync Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.FilterResync Property

[TnxDataset Class](#)

Pascal

```
published property FilterResync: boolean;
```

Remarks

When this property is set to True, changing the Filter property causes the server to refresh the dataset. Set this property to False when you don't want the server to refresh the dataset.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > FilterResync Property

25.1.116.5.32 FilterTimeout Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.FilterTimeout Property

[TnxDataset Class](#) | [Related Topics](#)

Pascal

```
published property FilterTimeout: TnxWord32;
```

Remarks

This is the timeout in ms for Filter operations.

Related Topics

[Filter](#), [Filtered](#), [FilterTimeout](#), [OnServerFilterTimeout](#), [FilterOptions](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > FilterTimeout Property

25.1.116.5.33 FilterType Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.FilterType Property

[TnxDataset Class](#)

Pascal

```
published property FilterType: TnxFilterType;
```

Remarks

The type of filter to use.

See Also[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Properties > FilterType Property

25.1.116.5.34 FlipOrder Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.FlipOrder Property**[TnxDataset Class](#)**Pascal**

```
published property FlipOrder: Boolean;
```

Remarks

Flips the navigation order for normal navigation as well as calls to key lookups.

See Also[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Properties > FlipOrder Property

25.1.116.5.35 OnCalcFields Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.OnCalcFields Property**[TnxDataset Class](#)**Pascal**

```
published property OnCalcFields;
```

Remarks

see TDataSet.OnCalcFields

See Also[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Properties > OnCalcFields Property

25.1.116.5.36 OnDeleteError Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.OnDeleteError Property**[TnxDataset Class](#)**Pascal**

```
published property OnDeleteError;
```

Remarks

see TDataSet.OnDeleteError

See Also[TnxDataset Class](#)*You are here:* Symbol Reference > Classes > TnxDataset Class > Properties > OnDeleteError Property

25.1.116.5.37 OnEditError Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.OnEditError Property

TnxDataset Class

Pascal

```
published property OnEditError;
```

Remarks

see TDataSet.OnEditError

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > OnEditError Property

25.1.116.5.38 OnFilterRecord Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.OnFilterRecord Property

TnxDataset Class

Pascal

```
published property OnFilterRecord;
```

Remarks

see TDataSet.OnFilterRecord

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > OnFilterRecord Property

25.1.116.5.39 OnNew Record Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.OnNewRecord Property

TnxDataset Class

Pascal

```
published property OnNewRecord;
```

Remarks

see TDataSet.OnNewRecord

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > OnNewRecord Property

25.1.116.5.40 OnPostError Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDataset.OnPostError Property

TnxDataset Class

Pascal

```
published property OnPostError;
```

Remarks

see [TDataSet.OnPostError](#)

See Also

[TnxDataset Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Properties](#) > [OnPostError Property](#)

25.1.116.5.41 Options Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.Options Property

[TnxDataset Class](#)

Pascal

```
published property Options: TnxDataSetOptions;
```

Remarks

The options of the current dataset

See Also

[TnxDataset Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Properties](#) > [Options Property](#)

25.1.116.5.42 Session Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.Session Property

[TnxDataset Class](#)

Pascal

```
published property Session: TnxBaseSession;
```

Remarks

This is the session the dataset is linked to. You only need to set this if you want to use AliasName, otherwise (when using an explicit database component) it is automatically set internally.

See Also

[TnxDataset Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxDataset Class](#) > [Properties](#) > [Session Property](#)

25.1.116.5.43 SimpleExpressionFilterClass Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.SimpleExpressionFilterClass Property

[TnxDataset Class](#)

Pascal

```
public property SimpleExpressionFilterClass: TnxSimpleExpressionFilterDescriptorClass;
```

Remarks

The Filter class the dataset creates an instance of for each Filter evaluation set

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > SimpleExpressionFilterClass Property

25.1.116.5.44 SqlFilterClass Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.SqlFilterClass Property**[TnxDataset Class](#)**Pascal**

```
public property SqlFilterClass: TnxSqlFilterDescriptorClass;
```

Remarks

the Filter class the dataset creates an instance of for each Filter evaluation set

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > SqlFilterClass Property

25.1.116.5.45 TableDescriptor Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.TableDescriptor Property**[TnxDataset Class](#)**Pascal**

```
public property TableDescriptor: TnxBaseTableDescriptor;
```

Remarks

Returns the Table Descriptor of the data dictionary.

See Also[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > TableDescriptor Property

25.1.116.5.46 Timeout Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDataset.Timeout Property**[TnxDataset Class](#) | [Related Topics](#)**Pascal**

```
published property Timeout: Integer;
```

Remarks

The is the timeout in ms used for communication with the server. If it -1 the timeout is inherited from Database.

Related Topics[FilterTimeout](#)

See Also

[TnxDataset Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > Timeout Property

25.1.116.5.47 Version Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataset.Version Property

[TnxDataset Class](#)

Pascal

```
published property Version: string;
```

Remarks

The version information for the current dataset.

See Also

[TnxDataset Class](#)

You are here: Symbol Reference > Classes > TnxDataset Class > Properties > Version Property

25.1.117 TnxDictionaryItem Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxDictionaryItem = class(TnxBaseRegisterableDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

The dictionary item class is the base class for all sub classes owned by a data dictionary. This includes all the different descriptor classes.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class

25.1.117.1.1.1 Destroy Destructor

25.1.117.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.Destroy Destructor

[TnxDictionaryItem Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Destructors > Destroy Destructor

25.1.117. Methods

25.1.117.2.1 Assign Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.Assign Method

[TnxDictionaryItem Class](#)

Pascal

```
public procedure Assign(
    Source: TPersistent
); override;
```

Remarks

Assigns the settings of a Source item to the current Item

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > Assign Method

25.1.117.2.2 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.CheckValid Method

[TnxDictionaryItem Class](#)

Pascal

```
public procedure CheckValid; virtual;
```

Remarks

This method is called to check the validity of the item. If a descendant needs to validate settings, this method should be overridden and on error raise a an exception.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > CheckValid Method

25.1.117.2.3 FindDescriptorClass Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.FindDescriptorClass Method

[TnxDictionaryItem Class](#)

Pascal

```
public class function FindDescriptorClass(
    const aName: string
): TnxDictionaryItemClass;
```

Remarks

Returns the class for the given named Descriptor class

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > [FindDescriptorClass Method](#)

25.1.117.2.4 FindParentOfType Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.FindParentOfType Method

[TnxDictionaryItem Class](#)

Pascal

```
public function FindParentOfType(
    aType: TPersistentClass;
    out aParent
): Boolean;
```

Remarks

Iterates the parents of the current item and returns the first found instance of a certain class type. Returns true if one is found.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > [FindParentOfType Method](#)

25.1.117.2.5 FindRelatedDescriptorOfType Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.FindRelatedDescriptorOfType Method

[TnxDictionaryItem Class](#)

Pascal

```
public function FindRelatedDescriptorOfType(
    aType: TnxDictionaryItemClass;
    out aDescriptor;
    aFrom: TnxDictionaryItem = nil
): Boolean; virtual;
```

Remarks

Iterates the parents of the given item aFrom and returns the first found instance of a certain class type. Returns true if one is found.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > FindRelatedDescriptorOfType Method

25.1.117.2.6 GetParentOfType Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDictionaryItem.GetParentOfType Method

TnxDictionaryItem Class

Pascal

```
public function GetParentOfType(
  aType: TPersistentClass
): TPersistent;
```

Remarks

Iterates the parents of the item and returns the first found instance of a certain class type. Raises an exception if none is found.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > GetParentOfType Method

25.1.117.2.7 GetRelatedDescriptorOfType Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDictionaryItem.GetRelatedDescriptorOfType Method

TnxDictionaryItem Class

Pascal

```
public function GetRelatedDescriptorOfType(
  aType: TnxDictionaryItemClass
): TnxDictionaryItem;
```

Remarks

Iterates the parents of the given item aFrom and returns the first found instance of a certain class type. Raises an exception if none is found.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > GetRelatedDescriptorOfType Method

25.1.117.2.8 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDictionaryItem.AreEqual Method

TnxDictionaryItem Class

Pascal

```
public function IsEqual(
  aDictionaryItem: TnxDictionaryItem
): Boolean; virtual;
```

Remarks

The IsEqual function should return true if the given aDictionaryItem is of the same type and has *exactly* the same settings. Should be overridden by all descendants.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > IsEqual Method

25.1.117.2.9 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.LoadFromReader Method

[TnxDictionaryItem Class](#)

Pascal

```
public procedure LoadFromReader(
  aReader: TReader
); virtual;
```

Parameters

Parameters

aReader

Description

the TReader to read the information from; must be able to cope with different versions!

Remarks

Loads the item settings from a Reader/Stream

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > LoadFromReader Method

25.1.117.2.10 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDictionaryItem.SaveToWriter Method

[TnxDictionaryItem Class](#)

Pascal

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
); virtual;
```

Parameters

Parameters

aWriter

Description

the TWriter to write to

aClientVersion

the Version of the information to write

Remarks

Saves the item settings to a Writer/Stream

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Methods > SaveToWriter Method

25.1.117.Properties

25.1.117.3.1 ConstraintName Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDictionaryItem.ConstraintName Property

TnxDictionaryItem Class

Pascal

```
public property ConstraintName: string;
```

Remarks

Set to the name of the constraint this dictionary item represents.

See Also

[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Properties > ConstraintName Property

25.1.117.3.2 CustomStrings Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDictionaryItem.CustomStrings Property

[TnxDictionaryItem Class](#) | [Related Topics](#)

Pascal

```
public property CustomStrings: TStrings;
```

Remarks

A storage container for developers to store extra data with the descriptor.

Related Topics

[HasCustomStrings](#)

See Also

[TnxDictionaryItem Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Properties > CustomStrings Property

25.1.117.3.3 HasCustomStrings Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxDictionaryItem.HasCustomStrings Property

[TnxDictionaryItem Class](#) | [Related Topics](#)

Pascal

```
public property HasCustomStrings: Boolean;
```

Remarks

This property indicates if the item has CustomStrings assigned.

Related Topics[HasCustomStrings](#)**See Also**

[TnxDictionaryItem Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Properties >
[HasCustomStrings Property](#)

25.1.117.3.4 ID Property[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxDictionaryItem.ID Property**[TnxDictionaryItem Class](#)**Pascal**

```
public property ID: Integer;
```

Remarks

Unique ID of the dictionary item

See Also[TnxDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxDictionaryItem Class > Properties > ID Property

25.1.11 TnxEmptyDefaultValueDescriptor Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxEmptyDefaultValueDescriptor Class**[nxsdDataDictionary](#)**Pascal**

```
public TnxEmptyDefaultValueDescriptor = class(TnxBaseDefaultValueDescriptor);
```

File[nxsdDataDictionary](#)**Remarks**

This descriptor should be used if you want to set a fields default value to not null. The practical result: strings become " (empty string), integers 0, etc.

By default all fields are NULL if not otherwise set.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Classes > TnxEmptyDefaultValueDescriptor Class

25.1.11 TnxExtendableServerObject Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxExtendableServerObject Class**[nxsdServerEngine](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxExtendableServerObject = class(TnxAbstractServerObject);
```

File

[nxsdServerEngine](#)

Remarks

The base class for Server objects that can be extended with an Extender.

See Also

[nxsdServerEngine](#)

[Members](#)

[Methods](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class

25.1.119. Constructors

25.1.119.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxExtendableServerObject.Create Constructor

[TnxExtendableServerObject Class](#)

Pascal

```
public constructor Create(
    aParent: TnxExtendableServerObject
);
```

Remarks

constructor.

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Constructors > Create Constructor

25.1.119. Destructors

25.1.119.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxExtendableServerObject.Destroy Destructor

[TnxExtendableServerObject Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Destructors > Destroy Destructor

25.1.119.Methods

25.1.119.3.1 AfterConstruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxExtendableServerObject.AfterConstruction Method

TnxExtendableServerObject Class

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

see TObject.AfterConstruction

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > AfterConstruction Method

25.1.119.3.2 BeforeDestruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxExtendableServerObject.BeforeDestruction Method

TnxExtendableServerObject Class

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

see TObject.BeforeConstruction

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > BeforeDestruction Method

25.1.119.3.3 esoOptionClear Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxExtendableServerObject.esoOptionClear Method

TnxExtendableServerObject Class

Pascal

```
public function esoOptionClear(
  const aName: string
): TnxResult; virtual;
```

Remarks

This is esoOptionClear, a member of class TnxExtendableServerObject.

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > esoOptionClear Method

25.1.119.3.4 esoOptionGetEffective Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxExtendableServerObject.esoOptionGetEffective Method**[TnxExtendableServerObject Class](#)**Pascal**

```
public function esoOptionGetEffective(
    const aName: string;
    out aValue: string
): TnxResult; virtual;
```

Remarks

This is esoOptionGetEffective, a member of class TnxExtendableServerObject.

See Also[TnxExtendableServerObject Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxExtendableServerObject Class](#) > [Methods](#) > [esoOptionGetEffective Method](#)

25.1.119.3.5 esoOptionSet Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxExtendableServerObject.esoOptionSet Method**[TnxExtendableServerObject Class](#)**Pascal**

```
public function esoOptionSet(
    const aName: string;
    const aValue: string
): TnxResult; virtual;
```

Remarks

This is esoOptionSet, a member of class TnxExtendableServerObject.

See Also[TnxExtendableServerObject Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxExtendableServerObject Class](#) > [Methods](#) > [esoOptionSet Method](#)

25.1.119.3.6 GetExtenderOfType Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxExtendableServerObject.GetExtenderOfType Method**[TnxExtendableServerObject Class](#)**Pascal**

```
public function GetExtenderOfType(
    aExtenderClass: TnxBaseEngineExtenderClass
): TnxBaseEngineExtender;
```

Remarks

returns the first Extender of a certain class type of the registered Extender list

See Also

TnxExtendableServerObject Class

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > GetExtenderOfType Method

25.1.119.3.7 NotifyExtenders Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxExtendableServerObject.NotifyExtenders Method**[TnxExtendableServerObject Class](#)**Pascal**

```
public function NotifyExtenders(
    aAction: TnxEngineAction;
    aBefore: Boolean;
    const aArgs: array of const
) : TnxResult; virtual;
```

Parameters

Parameters	Description
aAction	The action that is going to be executed or was executed.
aBefore	This Parameter is true if the Action is going to be executed.
aArgs	a number of Arguments passed in and/or out of the function

Remarks

This function is called to notify all Monitors attached to the owning Server Engine of an Action.

See Also[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > [NotifyExtenders Method](#)

25.1.119.3.8 OptionClear Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxExtendableServerObject.OptionClear Method**[TnxExtendableServerObject Class](#)**Pascal**

```
public function OptionClear(
    const aName: string
) : TnxResult; virtual;
```

Remarks

options

See Also[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > [OptionClear Method](#)

25.1.119.3.9 OptionGet Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxExtendableServerObject.OptionGet Method

TnxExtendableServerObject Class

Pascal

```
public function OptionGet(
    const aName: string;
    out aValue: string
): TnxResult; virtual;
```

Remarks

This is OptionGet, a member of class TnxExtendableServerObject.

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > OptionGet Method

25.1.119.3.10 OptionGetEffective Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxExtendableServerObject.OptionGetEffective Method

TnxExtendableServerObject Class

Pascal

```
public function OptionGetEffective(
    const aName: string;
    out aValue: string
): TnxResult; virtual;
```

Remarks

This is OptionGetEffective, a member of class TnxExtendableServerObject.

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > OptionGetEffective Method

25.1.119.3.11 OptionList Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxExtendableServerObject.OptionList Method

TnxExtendableServerObject Class

Pascal

```
public function OptionList(
    aList: TStrings
): TnxResult; virtual;
```

Remarks

This is OptionList, a member of class TnxExtendableServerObject.

See Also

TnxExtendableServerObject Class

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > OptionList Method

25.1.119.3.12 OptionListEffective Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxExtendableServerObject.OptionListEffective Method

TnxExtendableServerObject Class

Pascal

```
public function OptionListEffective(
    aList: TStrings
) : TnxResult; virtual;
```

Remarks

This is OptionListEffective, a member of class TnxExtendableServerObject.

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > OptionListEffective Method

25.1.119.3.13 OptionSet Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxExtendableServerObject.OptionSet Method

TnxExtendableServerObject Class

Pascal

```
public function OptionSet(
    const aName: string;
    const aValue: string
) : TnxResult; virtual;
```

Remarks

This is OptionSet, a member of class TnxExtendableServerObject.

See Also

[TnxExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxExtendableServerObject Class > Methods > OptionSet Method

25.1.12(TnxExtTextKeyFieldDescriptor Class)

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxExtTextKeyFieldDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxExtTextKeyFieldDescriptor = class(TnxTextKeyFieldDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This is an extended version of the Text Key Field descriptor that can also handle Multi bytes characters, locale support, etc.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxExtTextKeyFieldDescriptor Class

25.1.120. Methods

25.1.120.1.1 LoadFromReader Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxExtTextKeyFieldDescriptor.LoadFromReader Method

TnxExtTextKeyFieldDescriptor Class

Pascal

```
public procedure LoadFromReader(
  aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxExtTextKeyFieldDescriptor.

See Also

[TnxExtTextKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxExtTextKeyFieldDescriptor Class > Methods > LoadFromReader Method

25.1.120.1.2 SaveToWriter Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxExtTextKeyFieldDescriptor.SaveToWriter Method

TnxExtTextKeyFieldDescriptor Class

Pascal

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxExtTextKeyFieldDescriptor.

See Also

[TnxExtTextKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxExtTextKeyFieldDescriptor Class > Methods > SaveToWriter Method

25.1.12 TnxFieldDescriptor Class

NexusDB V2 VCL Reference

TnxFieldDescriptor Class

[Contents](#) | [Index](#)

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxFieldDescriptor = class(TnxLocalizedDictionaryItem, InxFieldDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This descriptor class describes fields. For each field added to the Data Dictionary one instance is created.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxFieldDescriptor Class](#)

25.1.121.1. Destructors

25.1.121.1.1 Destroy Destructor

NexusDB V2 VCL Reference

TnxFieldDescriptor.Destroy Destructor

[Contents](#) | [Index](#)

[TnxFieldDescriptor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxFieldDescriptor Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxFieldDescriptor Class](#) > [Destructors](#) > [Destroy Destructor](#)

25.1.121.2. Methods

25.1.121.2.1 AddDefaultValue Method

NexusDB V2 VCL Reference

TnxFieldDescriptor.AddDefaultValue Method

[Contents](#) | [Index](#)

[TnxFieldDescriptor Class](#)

Pascal

```
public function AddDefaultValue(
  aClass: TnxBaseDefaultValueDescriptorClass = nil
): TnxBaseDefaultValueDescriptor;
```

Remarks

Sets the Default Value Descriptor class of the field. And instance of this descriptor will be used to generate the Default value of the field for every new record.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > AddDefaultValue Method

25.1.121.2.2 AddValidations Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.AddValidations Method

[TnxFieldDescriptor Class](#)

Pascal

```
public function AddValidations(
    aClass: TnxFieldValidationsDescriptorClass = nil
): TnxFieldValidationsDescriptor;
```

Remarks

Sets the Validations Descriptor class of the field.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > AddValidations Method

25.1.121.2.3 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.CheckValid Method

[TnxFieldDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxFieldDescriptor.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > CheckValid Method

25.1.121.2.4 EnsureValidations Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.EnsureValidations Method

[TnxFieldDescriptor Class](#)

Pascal

```
public function EnsureValidations: TnxFieldValidationsDescriptor;
```

Remarks

Sets the Validations Descriptor class of the field.

■ **See Also**

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > EnsureValidations Method

25.1.121.2.5 FindRelatedDescriptorOfType Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.FindRelatedDescriptorOfType Method

[TnxFieldDescriptor Class](#)

Pascal

```
public function FindRelatedDescriptorOfType(
    aType: TnxDictionaryItemClass;
    out aDescriptor;
    aFrom: TnxDictionaryItem = nil
) : Boolean; override;
```

■ **Remarks**

This is FindRelatedDescriptorOfType, a member of class TnxFieldDescriptor.

■ **See Also**

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > FindRelatedDescriptorOfType Method

25.1.121.2.6 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.AreEqual Method

[TnxFieldDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
) : Boolean; override;
```

■ **Remarks**

This is IsEqual, a member of class TnxFieldDescriptor.

■ **See Also**

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > IsEqual Method

25.1.121.2.7 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.LoadFromReader Method

[TnxFieldDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
```

```
aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxFieldDescriptor.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > LoadFromReader Method

25.1.121.2.8 RemoveDefaultValue Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.RemoveDefaultValue Method

[TnxFieldDescriptor Class](#)

Pascal

```
public procedure RemoveDefaultValue;
```

Remarks

This method sets the Default Value Descriptor class to nil thus disabling default values for this field.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > RemoveDefaultValue Method

25.1.121.2.9 RemoveValidations Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.RemoveValidations Method

[TnxFieldDescriptor Class](#)

Pascal

```
public procedure RemoveValidations;
```

Remarks

This method sets the Validations Descriptor class to nil thus disabling validation for this field.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > RemoveValidations Method

25.1.121.2.10 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.SaveToWriter Method

[TnxFieldDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
```

```
aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxFieldDescriptor.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > SaveToWriter Method

25.1.121.2.11 SetupField Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.SetupField Method

[TnxFieldDescriptor Class](#)

Pascal

```
public class procedure SetupField(
  aType: TnxFieldType;
  var aUnits: Integer;
  var aDecPl: Integer;
  var aRequired: Boolean;
  out aLength: Integer
); virtual;
```

Parameters

Parameters	Description
aType	the Field type you need the values for
aUnits	the units wanted/needed for the given field type. Will be changed if invalid!
aDecPl	the number of decimal places wanted. Will be changed if invalid!
aRequired	set to true if you want the field to be required. Will be changed if invalid!
aLength	this will be the length in bytes needed for this field.

Remarks

This function returns the correct values for the various field types.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > SetupField Method

25.1.121.2.12 UpdateSetup Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.UpdateSetup Method

[TnxFieldDescriptor Class](#)

Pascal

```
public procedure UpdateSetup;
```

Remarks

This is UpdateSetup, a member of class TnxFieldDescriptor.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Methods > UpdateSetup Method

25.1.121.Properties

25.1.121.3.1 BufferAsVariant Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.BufferAsVariant Property

[TnxFieldDescriptor Class](#)

Pascal

```
public property BufferAsVariant [aFieldBuffer: PnxByteArray]: OleVariant;
```

Remarks

Gets/Sets a FieldBuffer as OleVariant.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Properties > BufferAsVariant Property

25.1.121.3.2 Name Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.Name Property

[TnxFieldDescriptor Class](#)

Pascal

```
public property Name: string;
```

Remarks

This is the name of the field.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Properties > Name Property

25.1.121.3.3 Number Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptor.Number Property

[TnxFieldDescriptor Class](#)

Pascal

```
public property Number: Integer;
```

Remarks

The index in the list of fields of the ParentDataDictionary.

See Also

[TnxFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldDescriptor Class > Properties > Number Property

25.1.12 TnxFieldsDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxFieldsDescriptor = class(TnxDictionaryItem, InxFieldsSource);
```

File

[nxsdDataDictionary](#)

Remarks

The Fields Descriptor holds a list of FieldDescriptors for the associated Data Dictionary.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class

25.1.122 Constructors

25.1.122.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.Create Constructor

[TnxFieldsDescriptor Class](#)

Pascal

```
public constructor Create(
  aParent: TPersistent
); virtual;
```

Remarks

constructor.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Constructors > Create Constructor

25.1.122.:Destructors

25.1.122.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldsDescriptor.Destroy Destructor

TnxFieldsDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Destructors > Destroy Destructor

25.1.122.:Methods

25.1.122.3.1 AddField Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldsDescriptor.AddField Method

TnxFieldsDescriptor Class

Pascal

```
public function AddField(
  const aName: string;
  const aDesc: string;
  aType: TnxFieldType;
  aUnits: Integer;
  aDecPl: Integer;
  aRequired: Boolean;
  aClass: TnxFieldDescriptorClass = nil
): TnxFieldDescriptor;
```

Parameters

Parameters	Description
aName	the field name
aDesc	a human readable description of the field
aType	the field type
aUnits	Units needed/wanted for the field
aDecPl	the number of decimal places for the field (if applicable)
aRequired	set to True if the field needs a value.
aClass	the class of the instance to be created.

Remarks

This function adds a new instance of a Field Descriptor to the data dictionary and returns the newly created Instance.

See Also

[TnxFieldsDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > AddField Method

25.1.122.3.2 AddValidations Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.AddValidations Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public function AddValidations(
    aClass: TnxFieldsValidationsDescriptorClass = nil
): TnxFieldsValidationsDescriptor;
```

Remarks

Sets the Validations Descriptor class of the field.

See Also[TnxFieldsDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > AddValidations Method

25.1.122.3.3 CheckValid Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.CheckValid Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxFieldsDescriptor.

See Also[TnxFieldsDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > CheckValid Method

25.1.122.3.4 Clear Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.Clear Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public procedure Clear(
    aNotify: Boolean
); virtual;
```

Remarks

This function clears the Data Dictionary.

See Also

[TnxFieldsDescriptor Class](#)[You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > Clear Method](#)

25.1.122.3.5 EnsureValidations Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.EsureValidations Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public function EnsureValidations: TnxFieldsValidationsDescriptor;
```

Remarks

Sets the Validations Descriptor class of the field.

See Also[TnxFieldsDescriptor Class](#)[You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > EnsureValidations Method](#)

25.1.122.3.6 fsdFieldChangeNotification Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.fsdFieldChangeNotification Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public procedure fsdFieldChangeNotification(
  aBefore: Boolean;
  aOldField: Integer;
  aNewField: Integer
);
```

Remarks

This is fsdFieldChangeNotification, a member of class TnxFieldsDescriptor.

See Also[TnxFieldsDescriptor Class](#)[You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > fsdFieldChangeNotification Method](#)

25.1.122.3.7 GetFieldForFilter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.GetFieldForFilter Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public function GetFieldForFilter(
  aField: Integer;
  aName: PChar;
  aData: PnxByteArray;
  out aType: TnxWord16;
  out aSize: TnxWord16;
```

```

    out aIsNull: Boolean;
    out aValue: Pointer
): Boolean;

```

Parameters

Parameters	Description
aField	the index of the field in the list of Field Descriptors.
aName	the name of the field.
aData	a pointer to the record buffer.
aType	returns the field type.
aSize	returns the data size of the field.
aIsNull	is set to true if the field is null. no data is written to aValue in this case
aValue	a pointer to a buffer to receive the field data.

Remarks

Given a record buffer, returns the pointer to the field data. Be careful because aValue is NOT a copy of the data but a pointer into the actual record buffer. This is a VERY fast method to access the raw field data.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > GetFieldForFilter Method

25.1.122.3.8 GetFieldFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.GetFieldFromName Method

[TnxFieldsDescriptor Class](#)

Pascal

```

public function GetFieldFromName(
  const aFieldName: string
): Integer;

```

Remarks

Return the field number for a given field name, or -1 if not found.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > GetFieldFromName Method

25.1.122.3.9 GetFields Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.GetFields Method

[TnxFieldsDescriptor Class](#)

Pascal

```

public procedure GetFields(

```

```
aStrings: TStrings
);
```

Parameters

Parameters	Description
aStrings	non nil reference to a TStrings object which will be filled with a list of all fieldnames.

Remarks

Fills a TStrings object with a list of all fields and references to the TnxFieldDescriptor objects.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > GetFields Method

25.1.122.3.10 GetRecordField Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.GetRecordField Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public procedure GetRecordField(
    aField: Integer;
    aData: PnxByteArray;
    var aIsNull: Boolean;
    aValue: Pointer
);
```

Parameters

Parameters	Description
aField	the index of the field in the list of Field Descriptors.
aData	a pointer to the record buffer
aIsNull	is set to true if the field is null. no data is written to aValue in this case
aValue	a pointer to a buffer to receive the field data.

Remarks

Given a record buffer, reads the field data.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > GetRecordField Method

25.1.122.3.11 GetRecordFieldForFilter

25.1.122.3.11.1 GetRecordFieldForFilter Method (Integer, PnxByteArray)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.GetRecordFieldForFilter Method (Integer, PnxByteArray)

[TnxFieldsDescriptor Class](#)

Pascal

```
public function GetRecordFieldForFilter(
    aField: Integer;
    aData: PnxByteArray
): Pointer; overload;
```

Remarks

Given a record buffer, returns the pointer to the field data. Be careful because aValue is NOT a copy of the data but a pointer into the actual record buffer. This is a VERY fast method to access the raw field data.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > [GetRecordFieldForFilter](#) > GetRecordFieldForFilter Method (Integer, PnxByteArray)

25.1.122.3.11.2 GetRecordFieldForFilter Method (Integer, PnxByteArray, TnxFieldType, TnxWord16, Pointer)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.GetRecordFieldForFilter Method (Integer, PnxByteArray, TnxFieldType, TnxWord16, Pointer)

[TnxFieldsDescriptor Class](#)

Pascal

```
public function GetRecordFieldForFilter(
    aField: Integer;
    aData: PnxByteArray;
    out aType: TnxFieldType;
    out aSize: TnxWord16;
    out aValue: Pointer
): Boolean; overload;
```

Remarks

Given a record buffer, returns the pointer to the field data. Be careful because aValue is NOT a copy of the data but a pointer into the actual record buffer. This is a VERY fast method to access the raw field data.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > [GetRecordFieldForFilter](#) > GetRecordFieldForFilter Method (Integer, PnxByteArray, TnxFieldType, TnxWord16, Pointer)

25.1.122.3.12 HasAutoIncField Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.HasAutoIncField Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public function HasAutoIncField(
    var aField: Integer
): Boolean;
```

Remarks

Return True and the index of the first autoinc field in the dictionary.

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > HasAutoIncField Method

25.1.122.3.13 HasBlobFields Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.HasBlobFields Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public function HasBlobFields(
    aBlobFields: TnxList = nil
): Boolean;
```

Remarks

Returns True if the table contains any Blob fields.

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > HasBlobFields Method

25.1.122.3.14 HasRecRevField Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.HasRecRevField Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public function HasRecRevField(
    var aField: Integer
): Boolean;
```

Remarks

Return True and the index of the first recrev field in the dictionary.

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > HasRecRevField Method

25.1.122.3.15 HasSameFields Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.HasSameFields Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public function HasSameFields(
    aSource: TnxFieldsDescriptor;
    aBlobFields: TnxList
```

```
) : Boolean;
```

Remarks

Use this method to verify a dictionary has the same field types, sizes, and ordering as a source dictionary. Returns True if the field information matches otherwise returns False. Note that the fields may have different names. If the record contains any Blob fields, the number of each Blob field is stored in output parameter aBlobFields.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > HasSameFields Method

25.1.122.3.16 InitRecord Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.InitRecord Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public procedure InitRecord(
  aData: PnxByteArray
);
```

Remarks

Given a record buffer, initialize it so that all fields are null

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > InitRecord Method

25.1.122.3.17 InsertFieldAt Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.InsertFieldAt Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public function InsertFieldAt(
  aInx: Integer;
  const aName: string;
  const aDesc: string;
  aType: TnxFieldType;
  aUnits: Integer;
  aDecPl: Integer;
  aRequired: Boolean;
  aClass: TnxFieldDescriptorClass = nil
) : TnxFieldDescriptor;
```

Parameters

Parameters

aInx

aName

aDesc

Description

the index of the new field

the field name

a human readable description of the field

aType	the field type
aUnits	Units needed/wanted for the field
aDecPl	the number of decimal places for the field (if applicable)
aRequired	set to True if the field needs a value.
aClass	the class of the instance to be created.

Remarks

This function adds a new instance of a Field Descriptor to the data dictionary and returns the newly created Instance. The field is inserted into the array at position aInx and the rest of the field array indexes are adjusted.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > InsertFieldAt Method

25.1.122.3.18 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.IsEqual Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxFieldsDescriptor.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > IsEqual Method

25.1.122.3.19 IsRecordFieldNull Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.IsRecordFieldNull Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public function IsRecordFieldNull(
    aField: Integer;
    aData: PnxByteArray
): Boolean;
```

Remarks

Given a record buffer, return True if the field is null.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > IsRecordFieldNull Method

25.1.122.3.20 LoadFromReader Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldsDescriptor.LoadFromReader Method

TnxFieldsDescriptor Class

Pascal

```
public procedure LoadFromReader(
  aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxFieldsDescriptor.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > LoadFromReader Method

25.1.122.3.21 MoveField Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldsDescriptor.MoveField Method

TnxFieldsDescriptor Class

Pascal

```
public procedure MoveField(
  aFrom: Integer;
  aTo: Integer
);
```

Remarks

Moves a FieldDescriptor from position aFrom to aTo in the list of Field Descriptors.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > MoveField Method

25.1.122.3.22 RemoveField Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldsDescriptor.RemoveField Method

TnxFieldsDescriptor Class

Pascal

```
public procedure RemoveField(
  aInx: Integer
);
```

Parameters

Parameters	Description
------------	-------------

aIdx

the index of the descriptor to be removed in the list of Field Descriptors.

Remarks

Remove a Field Descriptor from the Data Dictionary.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > RemoveField Method

25.1.122.3.23 RemoveValidations Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.RemoveValidations Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public procedure RemoveValidations;
```

Remarks

This method sets the Validations Descriptor class to nil thus disabling validation for this field.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > RemoveValidations Method

25.1.122.3.24 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.SaveToWriter Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxFieldsDescriptor.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > SaveToWriter Method

25.1.122.3.25 SetRecordField Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.SetRecordField Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public procedure SetRecordField(
    aField: Integer;
    aData: PnxByteArray;
    aValue: Pointer
);
```

Remarks

Given a record buffer, write the required field from the buffer pointed to by aValue; if aValue is nil, the field is set to null.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > SetRecordField Method

25.1.122.3.26 SetRecordFieldNull Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.SetRecordFieldNull Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public procedure SetRecordFieldNull(
    aField: Integer;
    aData: PnxByteArray;
    aIsNull: Boolean
);
```

Remarks

Given a record buffer, set the required field to null or non-null. Set the field in the record to binary zeros.

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > SetRecordFieldNull Method

25.1.122.3.27 UpdateLogRecLenAlign Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptor.UpdateLogRecLenAlign Method

[TnxFieldsDescriptor Class](#)

Pascal

```
public procedure UpdateLogRecLenAlign;
```

Remarks

Updates the Record Alignment

See Also

[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > UpdateLogRecLenAlign Method

25.1.122.3.28 UpdateSetupAndOffsets Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.UpdateSetupAndOffsets Method**[TnxFieldsDescriptor Class](#)**Pascal**

```
public procedure UpdateSetupAndOffsets;
```

Remarks

calls UpdateSetup on all fields and recalculates field offsets then calls UpdateLogRecLenAlign

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Methods > [UpdateSetupAndOffsets Method](#)

25.1.122.4.Properties

25.1.122.4.1 FieldByIndexAsVariant Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.FieldByIndexAsVariant Property**[TnxFieldsDescriptor Class](#)**Pascal**

```
public property FieldByIndexAsVariant [aRecordBuffer: PnxByteArray; aIndex: Integer]: OleVariant;
```

Remarks

Gets/Sets the value of the field on position aIndex as OleVariant in the given aRecordBuffer.

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Properties > [FieldByIndexAsVariant Property](#)

25.1.122.4.2 FieldByNameAsVariant Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxFieldsDescriptor.FieldNameAsVariant Property**[TnxFieldsDescriptor Class](#)**Pascal**

```
public property FieldByNameAsVariant [aRecordBuffer: PnxByteArray; const aName: string]: OleVariant;
```

Remarks

Gets/Sets the value of a named field as OleVariant in the given aRecordBuffer.

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Properties > [FieldNameAsVariant Property](#)

25.1.122.4.3 LogicalRecordLength Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxFieldsDescriptor.LogicalRecordLength Property**[TnxFieldsDescriptor Class](#)**Pascal**

```
public property LogicalRecordLength: Integer;
```

Remarks

The length of the logical record for the data dictionary (ie just the total size of the fields).

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Properties > LogicalRecordLength Property

25.1.122.4.4 RecordLength Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxFieldsDescriptor.RecordLength Property**[TnxFieldsDescriptor Class](#)**Pascal**

```
public property RecordLength: Integer;
```

Remarks

The length of the physical record for the data dictionary. Includes trailing byte array to identify null fields.

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Properties > RecordLength Property

25.1.122.4.5 Validations Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxFieldsDescriptor.Validations Property**[TnxFieldsDescriptor Class](#)**Pascal**

```
public property Validations: TnxFieldsValidationsDescriptor;
```

Remarks

This is Validations, a member of class TnxFieldsDescriptor.

See Also[TnxFieldsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsDescriptor Class > Properties > Validations Property

25.1.12 TnxFieldsValidationsDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsValidationsDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxFieldsValidationsDescriptor = class(TnxFieldValidationsDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

Holds a list of Field validation Descriptors

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

You are here: Symbol Reference > Classes > TnxFieldsValidationsDescriptor Class

25.1.123 Methods

25.1.123.1.1 AddValidation Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsValidationsDescriptor.AddValidation Method

TnxFieldsValidationsDescriptor Class

Pascal

```
public function AddValidation(
  const aName: string;
  aClass: TnxBaseFieldsValidationDescriptorClass = nil
): TnxBaseFieldValidationDescriptor;
```

Parameters

Parameters	Description
aName	the name of the Validation Descriptor
aClass	the class of the instance to be created.

Remarks

This function adds a new instance of a Validation Descriptor to the data dictionary and returns the newly created Instance.

See Also

[TnxFieldsValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldsValidationsDescriptor Class > Methods > AddValidation Method

25.1.12 TnxFieldValidationsDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldValidationsDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxFieldValidationsDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

Holds a list of Field validation Descriptors

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class

25.1.124. Constructors

25.1.124.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxFieldValidationsDescriptor.Create Constructor

[TnxFieldValidationsDescriptor Class](#)

Pascal

```
public constructor Create(
    aParent: TPersistent
); virtual;
```

Remarks

constructor.

See Also

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Constructors > Create Constructor

25.1.124. Destructors

25.1.124.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxFieldValidationsDescriptor.Destroy Destructor

[TnxFieldValidationsDescriptor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Destructors > Destroy Destructor

25.1.124. Methods

25.1.124.3.1 AddValidation Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldValidationsDescriptor.AddValidation Method

TnxFieldValidationsDescriptor Class

Pascal

```
public function AddValidation(
  const aName: string;
  aClass: TnxBaseFieldValidationDescriptorClass = nil
): TnxBaseFieldValidationDescriptor;
```

Parameters

Parameters	Description
aName	the name of the Validation Descriptor
aClass	the class of the instance to be created.

Remarks

This function adds a new instance of a Validation Descriptor to the data dictionary and returns the newly created Instance.

See Also

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > AddValidation Method

25.1.124.3.2 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldValidationsDescriptor.CheckValid Method

TnxFieldValidationsDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxFieldValidationsDescriptor.

See Also

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > CheckValid Method

25.1.124.3.3 Clear Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldValidationsDescriptor.Clear Method

TnxFieldValidationsDescriptor Class

Pascal

```
public procedure Clear;
```

Remarks

This function clears the Validation Descriptors Descriptor.

■ **See Also**

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > Clear Method

25.1.124.3.4 GetValidationDescriptorFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldValidationsDescriptor.GetValidationDescriptorFromName Method

[TnxFieldValidationsDescriptor Class](#)

Pascal

```
public function GetValidationDescriptorFromName(
    const aValidationDescriptorName: string
): Integer;
```

■ **Remarks**

Return the Validation number for a given Validation name, or -1 if not found.

■ **See Also**

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > GetValidationDescriptorFromName Method

25.1.124.3.5 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldValidationsDescriptor.AreEqual Method

[TnxFieldValidationsDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

■ **Remarks**

This is IsEqual, a member of class TnxFieldValidationsDescriptor.

■ **See Also**

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > IsEqual Method

25.1.124.3.6 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldValidationsDescriptor.LoadFromReader Method

[TnxFieldValidationsDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
```

```
); override;
```

Remarks

This is LoadFromReader, a member of class TnxFieldValidationsDescriptor.

See Also

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > LoadFromReader Method

25.1.124.3.7 RemoveValidation

25.1.124.3.7.1 RemoveValidation Method (Integer)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldValidationsDescriptor.RemoveValidation Method (Integer)

[TnxFieldValidationsDescriptor Class](#)

Pascal

```
public procedure RemoveValidation(
    aInx: Integer
); overload;
```

Parameters

Parameters	Description
aInx	the index of the descriptor to be removed in the list of Validation Descriptors.

Remarks

Remove a Validation Descriptor from the Data Dictionary.

See Also

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > RemoveValidation > RemoveValidation Method (Integer)

25.1.124.3.7.2 RemoveValidation Method (string)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldValidationsDescriptor.RemoveValidation Method (string)

[TnxFieldValidationsDescriptor Class](#)

Pascal

```
public procedure RemoveValidation(
    const aName: string
); overload;
```

Parameters

Parameters	Description
aName	the name of the descriptor to be removed.

Remarks

Remove a Validation Descriptor from the Data Dictionary.

See Also

TnxFieldValidationsDescriptor Class

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > RemoveValidation > RemoveValidation Method (string)

25.1.124.3.8 SaveToWriter Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFieldValidationsDescriptor.SaveToWriter Method

TnxFieldValidationsDescriptor Class

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxFieldValidationsDescriptor.

See Also

[TnxFieldValidationsDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFieldValidationsDescriptor Class > Methods > SaveToWriter Method

25.1.12.TnxFileDescriptor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxFileDescriptor = class(TnxDictionaryItem);
```

File

[nxsdDataDictionary](#)

Remarks

A file descriptor describes the attributes of a physical file.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class

25.1.125.1Methods

25.1.125.1.1 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor.CheckValid Method

[TnxFileDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxFileDescriptor.

See Also

[TnxFileDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Methods > CheckValid Method

25.1.125.1.2 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFileDescriptor.IsEqual Method

[TnxFileDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxFileDescriptor.

See Also

[TnxFileDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Methods > IsEqual Method

25.1.125.1.3 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFileDescriptor.LoadFromReader Method

[TnxFileDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxFileDescriptor.

See Also

[TnxFileDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Methods > LoadFromReader Method

25.1.125.1.4 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFileDescriptor.SaveToWriter Method

[TnxFileDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxFileDescriptor.

See Also

[TnxFileDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Methods > SaveToWriter Method

25.1.125.Properties

25.1.125.2.1 BlockSize Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFileDescriptor.BlockSize Property

[TnxFileDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property BlockSize: TnxBlockSize;
```

Remarks

The block size of the file. Please note that for table files (with the NexusDB default engines) each record has to fit into 1 block.

Related Topics

[BlockSize](#), [BlockSizeBytes](#), [UsableBlockSize](#)

See Also

[TnxFileDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Properties > BlockSize Property

25.1.125.2.2 BlockSizeBytes Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFileDescriptor.BlockSizeBytes Property

[TnxFileDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property BlockSizeBytes: TnxWord32;
```

Remarks

The block size of the file in bytes. Please note that for table files (with the NexusDB default engines) each record has to fit into 1 block.

Related Topics

[BlockSize](#), [BlockSizeBytes](#), [UsableBlockSize](#)

See Also

[TnxFileDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Properties > BlockSizeBytes Property

25.1.125.2.3 Desc Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor.Desc Property

TnxFileDescriptor Class

Pascal

```
public property Desc: string;
```

Remarks

This can be filled with a human readable description of the file.

See Also

[TnxFileDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Properties > Desc Property

25.1.125.2.4 Extension Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor.Extension Property

TnxFileDescriptor Class

Pascal

```
public property Extension: string;
```

Remarks

The file extension used for the file.

See Also

[TnxFileDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Properties > Extension Property

25.1.125.2.5 Grow Size Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor.GrowSize Property

[TnxFileDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property GrowSize: TnxGrowSize;
```

Remarks

This is the size, in numbers of blocks, the file will grow when needed.

Related Topics

[InitialSize](#)

See Also

[TnxFileDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Properties > GrowSize Property

25.1.125.2.6 InitialSize Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor.InitialSize Property

[TnxFileDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property InitialSize: TnxGrowSize;
```

Remarks

This is the initial size, in numbers of blocks, of the file described.

Related Topics

[GrowSize](#)

See Also

[TnxFileDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Properties > InitialSize Property

25.1.125.2.7 Number Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor.Number Property

[TnxFileDescriptor Class](#)

Pascal

```
public property Number: Integer;
```

Remarks

This number holds the index of this descriptor in the list of File descriptors in the ParentDataDictionary.

See Also

[TnxFileDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFileDescriptor Class > Properties > Number Property

25.1.125.2.8 UsableBlockSize Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFileDescriptor.UsableBlockSize Property

[TnxFileDescriptor Class](#)

Pascal

```
public property UsableBlockSize: TnxWord32;
```

Remarks

The number of bytes that can be used per Block. That is calculated by subtracting the header size from the total BlockSize.

See Also[TnxFileDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxFileDescriptor Class > Properties > UsableBlockSize Property

25.1.12 TnxFilesDescriptor Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxFilesDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#)**Pascal**

```
public TnxFilesDescriptor = class(TnxDictionaryItem);
```

File[nxsdDataDictionary](#)**Remarks**

A Files Descriptors is a list of associated FileDescriptors

See Also[nxsdDataDictionary](#)[Members](#)[Methods](#)*You are here:* Symbol Reference > Classes > TnxFilesDescriptor Class

25.1.126 Constructors

25.1.126.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxFilesDescriptor.Create Constructor

[TnxFilesDescriptor Class](#)**Pascal**

```
public constructor Create(
  aParent: TPersistent
); virtual;
```

Remarks

constructor.

See Also[TnxFilesDescriptor Class](#)*You are here:* Symbol Reference > Classes > TnxFilesDescriptor Class > Constructors > Create Constructor

25.1.126.Destructors

25.1.126.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxFilesDescriptor.Destroy Destructor

TnxFilesDescriptor Class

```
Pascal
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Destructors > Destroy Destructor

25.1.126.Methods

25.1.126.3.1 AddFile Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.AddFile Method

TnxFilesDescriptor Class

```
Pascal
public function AddFile(
  const aExtension: string;
  aBlockSize: TnxBlockSize;
  const aDesc: string = '';
  aClass: TnxFileDescriptorClass = nil
): TnxFileDescriptor;
```

Parameters

Parameters	Description
aExtension	the file extension to use.
aBlockSize	the block size for the new File Descriptor
aDesc	a human readable description
aClass	the class of the instance to be created

Remarks

This function adds a new instance of a File Descriptor to the data dictionary and returns the newly created Instance.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > AddFile Method

25.1.126.3.2 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.CheckValid Method

TnxFilesDescriptor Class

```
Pascal
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxFilesDescriptor.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > CheckValid Method

25.1.126.3.3 Clear Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.Clear Method

[TnxFilesDescriptor Class](#)

Pascal

```
public procedure Clear(
    aNotify: Boolean
); virtual;
```

Remarks

This function clears the Files Descriptor.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > Clear Method

25.1.126.3.4 GetFileFromExt Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.GetFileFromExt Method

[TnxFilesDescriptor Class](#)

Pascal

```
public function GetFileFromExt(
    const aFileExt: string
): Integer;
```

Parameters

Parameters

aFileExt

Description

the file extension (attached to the base name) to look for.

Remarks

Return the file number for a given file extension, or -1 if not found.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > GetFileFromExt Method

25.1.126.3.5 InsertFileAt Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.InsertFileAt Method

TnxFilesDescriptor Class

```
Pascal
public function InsertFileAt(
    aInx: Integer;
    const aExtension: string;
    aBlockSize: TnxBlockSize;
    const aDesc: string = '';
    aClass: TnxFileDescriptorClass = nil
): TnxFileDescriptor;
```

Parameters

Parameters	Description
aInx	the index of the new field
aExtension	the file extension to use.
aBlockSize	the block size for the new File Descriptor
aDesc	a human readable description
aClass	the class of the instance to be created

Remarks

This function adds a new instance of a File Descriptor to the data dictionary and returns the newly created Instance. The field is inserted into the array at position aInx and the rest of the field array indexes are adjusted.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > InsertFileAt Method

25.1.126.3.6 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.IsEqual Method

[TnxFilesDescriptor Class](#)

```
Pascal
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxFilesDescriptor.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > IsEqual Method

25.1.126.3.7 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.LoadFromReader Method

[TnxFilesDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxFilesDescriptor.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > LoadFromReader Method

25.1.126.3.8 MoveFile Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.MoveFile Method

[TnxFilesDescriptor Class](#)

Pascal

```
public procedure MoveFile(
    aFrom: Integer;
    aTo: Integer
);
```

Remarks

Moves a FileDescriptor in the list of descriptors from aFrom to the aTo position

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > MoveFile Method

25.1.126.3.9 RemoveFile Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptor.RemoveFile Method

[TnxFilesDescriptor Class](#)

Pascal

```
public procedure RemoveFile(
    aInx: Integer
);
```

Parameters**Parameters**

aInx

Description

the index of the descriptor to be removed in the list of File Descriptors.

Remarks

Remove a File Descriptor from the Data Dictionary.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > RemoveFile Method

25.1.126.3.10 SaveToWriter Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxFilesDescriptor.SaveToWriter Method

[TnxFilesDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxFilesDescriptor.

See Also

[TnxFilesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxFilesDescriptor Class > Methods > SaveToWriter Method

25.1.12 TnxGetServerNamesHandler Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxGetServerNamesHandler Class

[nxIITransport](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxGetServerNamesHandler = class(TnxInterfacedObject, InxBroadcastReplyHandler);
```

File

[nxIITransport](#)

Remarks

This is class TnxGetServerNamesHandler.

See Also

[nxIITransport](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxGetServerNamesHandler Class

25.1.127 Constructors

25.1.127.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxGetServerNamesHandler.Create Constructor

[TnxGetServerNamesHandler Class](#)

Pascal

```
public constructor Create(
```

```
aTransport: TnxBaseTransport
);
```

Remarks

This is Create, a member of class TnxGetServerNamesHandler.

See Also

[TnxGetServerNamesHandler Class](#)

You are here: Symbol Reference > Classes > TnxGetServerNamesHandler Class > Constructors > Create Constructor

25.1.127.:Destructors

25.1.127.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxGetServerNamesHandler.Destroy Destructor

[TnxGetServerNamesHandler Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxGetServerNamesHandler.

See Also

[TnxGetServerNamesHandler Class](#)

You are here: Symbol Reference > Classes > TnxGetServerNamesHandler Class > Destructors > Destroy Destructor

25.1.127.:Methods

25.1.127.3.1 GetServerNames Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxGetServerNamesHandler.GetServerNames Method

[TnxGetServerNamesHandler Class](#)

Pascal

```
public procedure GetServerNames(
  aList: TStrings;
  aTimeOut: TnxWord32
);
```

Remarks

This is GetServerNames, a member of class TnxGetServerNamesHandler.

See Also

[TnxGetServerNamesHandler Class](#)

You are here: Symbol Reference > Classes > TnxGetServerNamesHandler Class > Methods > GetServerNames Method

25.1.12{TnxHeapBlobDescriptor Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxHeapBlobDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxHeapBlobDescriptor = class(TnxBaseBlobDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

A Heap Blob Descriptor adds support to Blob Descriptors for specifying a Heap Descriptor to read/store blobs on a heap.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class

25.1.128. Constructors

25.1.128.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxHeapBlobDescriptor.Create Constructor

[TnxHeapBlobDescriptor Class](#)

Pascal

```
public constructor Create(
    aParent: TPersistent;
    const aHeaderSection: string
); override;
```

Remarks

constructor.

See Also

[TnxHeapBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Constructors > Create Constructor

25.1.128. Destructors

25.1.128.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxHeapBlobDescriptor.Destroy Destructor

[TnxHeapBlobDescriptor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxHeapBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Destructors > Destroy Destructor

25.1.128. Methods

25.1.128.3.1 AddHeapDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxHeapBlobDescriptor.AddHeapDescriptor Method

[TnxHeapBlobDescriptor Class](#) | [Related Topics](#)

Pascal

```
public function AddHeapDescriptor(
    aHeapClass: TnxBaseHeapDescriptorClass = nil
): TnxBaseHeapDescriptor;
```

Remarks

Sets the associated Heap Descriptor.

Related Topics

[AddHeapDescriptor](#), [RemoveHeapDescriptor](#), [HeapDescriptor](#)

See Also

[TnxHeapBlobDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Methods > [AddHeapDescriptor Method](#)

25.1.128.3.2 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxHeapBlobDescriptor.CheckValid Method

[TnxHeapBlobDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxHeapBlobDescriptor.

See Also

[TnxHeapBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Methods > [CheckValid Method](#)

25.1.128.3.3 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxHeapBlobDescriptor.AreEqual Method

[TnxHeapBlobDescriptor Class](#)

```
Pascal
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxHeapBlobDescriptor.

See Also[TnxHeapBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Methods > IsEqual Method

[25.1.128.3.4 LoadFromReader Method](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapBlobDescriptor.LoadFromReader Method**[TnxHeapBlobDescriptor Class](#)

```
Pascal
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxHeapBlobDescriptor.

See Also[TnxHeapBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Methods > LoadFromReader Method

[25.1.128.3.5 RemoveHeapDescriptor Method](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapBlobDescriptor.RemoveHeapDescriptor Method**[TnxHeapBlobDescriptor Class](#) | [Related Topics](#)

```
Pascal
public procedure RemoveHeapDescriptor;
```

Remarks

Sets the associated Heap Descriptor back to nil.

Related Topics[AddHeapDescriptor](#), [RemoveHeapDescriptor](#), [HeapDescriptor](#)**See Also**[TnxHeapBlobDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Methods > RemoveHeapDescriptor Method

25.1.128.3.6 SaveToWriter Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxHeapBlobDescriptor.SaveToWriter Method**[TnxHeapBlobDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxHeapBlobDescriptor.

See Also

[TnxHeapBlobDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Methods > SaveToWriter Method

25.1.128.4 Properties

25.1.128.4.1 HeapDescriptor Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxHeapBlobDescriptor.HeapDescriptor Property**[TnxHeapBlobDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public property HeapDescriptor: TnxBaseHeapDescriptor;
```

Remarks

Returns the associated Heap Descriptor.

Related Topics

[AddHeapDescriptor](#), [RemoveHeapDescriptor](#), [HeapDescriptor](#)

See Also

[TnxHeapBlobDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapBlobDescriptor Class > Properties > HeapDescriptor Property

25.1.129 TnxHeapRecordDescriptor Class*NexusDB V2 VCL Reference*[Contents](#) | [Index](#)**TnxHeapRecordDescriptor Class**[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxHeapRecordDescriptor = class(TnxBaseRecordDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This is a descendant of Record Descriptor that allows to specify a Heap Descriptor that is used by the specified engine.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class

25.1.129.1.Destructors

25.1.129.1.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxHeapRecordDescriptor.Destroy Destructor

[TnxHeapRecordDescriptor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxHeapRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Destructors > Destroy Destructor

25.1.129.2.Methods

25.1.129.2.1 AddHeapDescriptor Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxHeapRecordDescriptor.AddHeapDescriptor Method

[TnxHeapRecordDescriptor Class](#) | [Related Topics](#)

Pascal

```
public function AddHeapDescriptor(
  aHeapClass: TnxBaseHeapDescriptorClass = nil
): TnxBaseHeapDescriptor;
```

Remarks

Set the associated Heap Descriptor.

Related Topics

[AddHeapDescriptor](#), [RemoveHeapDescriptor](#), [HeapDescriptor](#)

See Also

[TnxHeapRecordDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > [AddHeapDescriptor Method](#)

25.1.129.2.2 AddRecordCompressionDescriptor Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapRecordDescriptor.AddRecordCompressionDescriptor Method**[TnxHeapRecordDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public function AddRecordCompressionDescriptor(
    aRecordCompressionClass: TnxBaseRecordCompressionDescriptorClass = nil
): TnxBaseRecordCompressionDescriptor;
```

Remarks

Set the associated RecordCompression Descriptor.

Related Topics

[AddRecordCompressionDescriptor](#), [RemoveRecordCompressionDescriptor](#), [RecordCompressionDescriptor](#)

See Also

[TnxHeapRecordDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > [AddRecordCompressionDescriptor Method](#)

25.1.129.2.3 CheckValid Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapRecordDescriptor.CheckValid Method**[TnxHeapRecordDescriptor Class](#)**Pascal**

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxHeapRecordDescriptor.

See Also

[TnxHeapRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > [CheckValid Method](#)

25.1.129.2.4 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapRecordDescriptor.AreEqual Method**[TnxHeapRecordDescriptor Class](#)**Pascal**

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxHeapRecordDescriptor.

See Also[TnxHeapRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > IsEqual Method

25.1.129.2.5 LoadFromReader Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapRecordDescriptor.LoadFromReader Method**[TnxHeapRecordDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxHeapRecordDescriptor.

See Also[TnxHeapRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > LoadFromReader Method

25.1.129.2.6 RemoveHeapDescriptor Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapRecordDescriptor.RemoveHeapDescriptor Method**[TnxHeapRecordDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public procedure RemoveHeapDescriptor;
```

Remarks

Sets the associated Heap Descriptor to nil.

Related Topics[AddHeapDescriptor](#), [RemoveHeapDescriptor](#), [HeapDescriptor](#)**See Also**[TnxHeapRecordDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > RemoveHeapDescriptor Method

25.1.129.2.7 RemoveRecordCompressionDescriptor Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxHeapRecordDescriptor.RemoveRecordCompressionDescriptor Method**[TnxHeapRecordDescriptor Class](#) | [Related Topics](#)**Pascal**

```
public procedure RemoveRecordCompressionDescriptor;
```

Remarks

Sets the associated RecordCompression Descriptor to nil.

Related Topics

[AddRecordCompressionDescriptor](#), [RemoveRecordCompressionDescriptor](#), [RecordCompressionDescriptor](#)

See Also

[TnxHeapRecordDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > RemoveRecordCompressionDescriptor Method

25.1.129.2.8 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxHeapRecordDescriptor.SaveToWriter Method

[TnxHeapRecordDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxHeapRecordDescriptor.

See Also

[TnxHeapRecordDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Methods > SaveToWriter Method

25.1.129.Properties

25.1.129.3.1 HeapDescriptor Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxHeapRecordDescriptor.HeapDescriptor Property

[TnxHeapRecordDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property HeapDescriptor: TnxBaseHeapDescriptor;
```

Remarks

Returns the associated Heap Descriptor.

Related Topics

[AddHeapDescriptor](#), [RemoveHeapDescriptor](#), [HeapDescriptor](#)

See Also

[TnxHeapRecordDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Properties > HeapDescriptor Property

25.1.129.3.2 RecordCompressionDescriptor Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxHeapRecordDescriptor.RecordCompressionDescriptor Property

[TnxHeapRecordDescriptor Class](#) | [Related Topics](#)

Pascal

```
public property RecordCompressionDescriptor: TnxBaseRecordCompressionDescriptor;
```

Remarks

Returns the associated RecordCompression Descriptor.

Related Topics

[AddRecordCompressionDescriptor](#), [RemoveRecordCompressionDescriptor](#),
[RecordCompressionDescriptor](#)

See Also

[TnxHeapRecordDescriptor Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxHeapRecordDescriptor Class > Properties > RecordCompressionDescriptor Property

25.1.13(TnxIndexDataSet Class)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet Class

[nxdb](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxIndexDataSet = class(TnxDataset);
```

File

nxdb

Remarks

TnxIndexDataSet adds support for indexing to TnxDataset.

See Also

[nxdb](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class

25.1.130.'Constructors

25.1.130.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet.Create Constructor

[TnxIndexDataSet Class](#)

Pascal

```
public constructor Create(
    AOwner: TComponent
) override;
```

Remarks

constructor.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Constructors > Create Constructor

25.1.130.:Destructors

25.1.130.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.Destroy Destructor

[TnxIndexDataSet Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Destructors > Destroy Destructor

25.1.130.:Methods

25.1.130.3.1 ApplyRange Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.ApplyRange Method

[TnxIndexDataSet Class](#)

Pascal

```
public procedure ApplyRange;
```

Remarks

see tTable.ApplyRange.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > ApplyRange Method

25.1.130.3.2 Cancel Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.Cancel Method

[TnxIndexDataSet Class](#)

Pascal
public procedure Cancel; **override**;

Remarks

see tDataSet.Cancel.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > Cancel Method

25.1.130.3.3 CancelRange Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDataSet.CancelRange Method**[TnxIndexDataSet Class](#)

Pascal
public procedure CancelRange;

Remarks

see tTable.CancelRange.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > CancelRange Method

25.1.130.3.4 EditKey Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDataSet.EditKey Method**[TnxIndexDataSet Class](#)

Pascal
public procedure EditKey;

Remarks

see tTable.EditKey.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > EditKey Method

25.1.130.3.5 EditRangeEnd Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDataSet.EditRangeEnd Method**[TnxIndexDataSet Class](#)

Pascal
public procedure EditRangeEnd;

Remarks

see tTable.EditRangeEnd.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > EditRangeEnd Method

25.1.130.3.6 EditRangeStart Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.EditRangeStart Method

[TnxIndexDataSet Class](#)

Pascal

```
public procedure EditRangeStart;
```

Remarks

see tTable.EditRangeStart.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > EditRangeStart Method

25.1.130.3.7 FindKey

25.1.130.3.7.1 FindKey Method (array of const)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.FindKey Method (array of const)

[TnxIndexDataSet Class](#)

Pascal

```
public function FindKey(
    const aKeyValues: array of const
): Boolean; overload;
```

Remarks

see tTable.FindKey.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > FindKey > FindKey Method (array of const)

25.1.130.3.7.2 FindKey Method (array of const, TnxSetKeyOptions)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.FindKey Method (array of const, TnxSetKeyOptions)

[TnxIndexDataSet Class](#)

Pascal

```
public function FindKey(
    const aKeyValues: array of const;
    aOptions: TnxSetKeyOptions
) : Boolean; overload;
```

Remarks

see tTable.FindKey.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > FindKey > FindKey Method (array of const, TnxSetKeyOptions)

25.1.130.3.8 FindNearest

25.1.130.3.8.1 FindNearest Method (array of const)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.FindNearest Method (array of const)

[TnxIndexDataSet Class](#)

Pascal

```
public procedure FindNearest(
    const aKeyValues: array of const
); overload;
```

Remarks

see tTable.FindNearest.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > FindNearest > FindNearest Method (array of const)

25.1.130.3.8.2 FindNearest Method (array of const, TnxSetKeyOptions)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.FindNearest Method (array of const, TnxSetKeyOptions)

[TnxIndexDataSet Class](#)

Pascal

```
public procedure FindNearest(
    const aKeyValues: array of const;
    aOptions: TnxSetKeyOptions
); overload;
```

Remarks

see tTable.FindNearest.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > FindNearest > FindNearest Method (array of const, TnxSetKeyOptions)

25.1.130.3.9 GetIndexNames Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxIndexDataSet.GetIndexNames Method**[TnxIndexDataSet Class](#)**Pascal**

```
public procedure GetIndexNames(
    aList: TStrings
);
```

Remarks

see `tTable.GetIndexNames`.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > GetIndexNames Method

25.1.130.3.10 GotoKey Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxIndexDataSet.GotoKey Method**[TnxIndexDataSet Class](#)**Pascal**

```
public function GotoKey: Boolean;
```

Remarks

see `tTable.GotoKey`.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > GotoKey Method

25.1.130.3.11 GotoNearest Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxIndexDataSet.GotoNearest Method**[TnxIndexDataSet Class](#)**Pascal**

```
public procedure GotoNearest;
```

Remarks

see `tTable.GotoNearest`

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > GotoNearest Method

25.1.130.3.12 GotoNearestBackward Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDataSet.GotoNearestBackward Method**[TnxIndexDataSet Class](#)**Pascal**

```
public procedure GotoNearestBackward;
```

Remarks

see tTable.GotoNearest

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > GotoNearestBackward Method

25.1.130.3.13 Post Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDataSet.Post Method**[TnxIndexDataSet Class](#)**Pascal**

```
public procedure Post; override;
```

Remarks

see tDataSet.Post.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > Post Method

25.1.130.3.14 SetKey Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDataSet.SetKey Method**[TnxIndexDataSet Class](#)**Pascal**

```
public procedure SetKey;
```

Remarks

see tTable.FindNearest.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetKey Method

25.1.130.3.15 SetRange

25.1.130.3.15.1 SetRange Method (array of const, array of const)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet.SetRange Method (array of const, array of const)

TnxIndexDataSet Class

Pascal

```
public procedure SetRange(
    const aStartValues: array of const;
    const aEndValues: array of const
); overload;
```

Remarks

see tTable.SetRange.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRange > SetRange Method (array of const, array of const)

25.1.130.3.15.2 SetRange Method (array of const, array of const, TnxSetKeyOptions)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet.SetRange Method (array of const, array of const, TnxSetKeyOptions)

TnxIndexDataSet Class

Pascal

```
public procedure SetRange(
    const aStartValues: array of const;
    const aEndValues: array of const;
    aOptions: TnxSetKeyOptions
); overload;
```

Remarks

see tTable.SetRange.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRange > SetRange Method (array of const, array of const, TnxSetKeyOptions)

25.1.130.3.15.3 SetRange Method (array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet.SetRange Method (array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions)

TnxIndexDataSet Class

Pascal

```
public procedure SetRange(
    const aStartValues: array of const;
    const aEndValues: array of const;
```

```
aStartOptions: TnxSetKeyOptions;
aEndOptions: TnxSetKeyOptions
); overload;
```

Remarks

see tTable.SetRange.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRange > [SetRange Method \(array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions\)](#)

25.1.130.3.16 SetRangeEnd Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.SetRangeEnd Method

[TnxIndexDataSet Class](#)

Pascal

```
public procedure SetRangeEnd;
```

Remarks

see tTable.SetRangeEnd.

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRangeEnd Method

25.1.130.3.17 SetRangeShared

25.1.130.3.17.1 SetRangeShared Method (array of const, array of const, array of const, TnxSetKeyOptions)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDataSet.SetRangeShared Method (array of const, array of const, array of const, TnxSetKeyOptions)

[TnxIndexDataSet Class](#)

Pascal

```
public procedure SetRangeShared(
  const aSharedValue: array of const;
  const aStartValues: array of const;
  const aEndValues: array of const;
  aOptions: TnxSetKeyOptions = []
); overload;
```

Remarks

???

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRangeShared > [SetRangeShared Method \(array of const, array of const, array of const, TnxSetKeyOptions\)](#)

25.1.130.3.17.2 SetRangeShared Method (array of const, array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet.SetRangeShared Method (array of const, array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions)

TnxIndexDataSet Class

Pascal

```
public procedure SetRangeShared(
  const aSharedValue: array of const;
  const aStartValues: array of const;
  const aEndValues: array of const;
  aStartOptions: TnxSetKeyOptions;
  aEndOptions: TnxSetKeyOptions
); overload;
```

Remarks

???

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRangeShared > SetRangeShared Method (array of const, array of const, array of const, TnxSetKeyOptions, TnxSetKeyOptions)

25.1.130.3.18 SetRangeSimple Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet.SetRangeSimple Method

TnxIndexDataSet Class

Pascal

```
public procedure SetRangeSimple(
  const aValue: array of const;
  aOptions: TnxSetKeyOptions = []
); overload;
```

Remarks

Sets a normal simple Range (as opposed to SetRangeShared).

See Also

[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRangeSimple Method

25.1.130.3.19 SetRangeStart Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDataSet.SetRangeStart Method

TnxIndexDataSet Class

Pascal

```
public procedure SetRangeStart;
```

Remarks

see tTable.SetRangeStart.

See Also[TnxIndexDataSet Class](#)

You are here: Symbol Reference > Classes > TnxIndexDataSet Class > Methods > SetRangeStart Method

25.1.13 TnxIndexDescriptor Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxIndexDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxIndexDescriptor = class(TnxDictionaryItem);
```

File[nxsdDataDictionary](#)**Remarks**

An Index Descriptor describes how NexusDB creates an Index for a table. It uses Key Descriptor for creating the record keys and optionally a File Descriptor to describe a sperate file used for storing the index.

See Also[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class

25.1.131 Constructors

25.1.131.1.1 CreateStandalone Constructor

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxIndexDescriptor.CreateStandalone Constructor

[TnxIndexDescriptor Class](#)**Pascal**

```
public constructor CreateStandalone(
  aNumber: Integer;
  const aName: string;
  aFileNumber: Integer;
  aKeyClass: TnxKeyDescriptorClass = nil
); overload;
```

Remarks

With this constructor an Index descriptor can be created that has NO Data Dictionary as parent.

See Also[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Constructors > CreateStandalone Constructor

25.1.131.:Destructors

25.1.131.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.Destroy Destructor

TnxIndexDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Destructors > Destroy Destructor

25.1.131.:Methods

25.1.131.3.1 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.CheckValid Method

TnxIndexDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxIndexDescriptor.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Methods > CheckValid Method

25.1.131.3.2 CreateStandaloneFromReader Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.CreateStandaloneFromReader Method

TnxIndexDescriptor Class

Pascal

```
public class function CreateStandaloneFromReader(
  aReader: TReader
) : TnxIndexDescriptor; overload;
```

Remarks

With this constructor an Index descriptor can be created that has NO Data Dictionary as parent.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Methods > CreateStandaloneFromReader Method

25.1.131.3.3 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.IsEqual Method

TnxIndexDescriptor Class

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxIndexDescriptor.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Methods > IsEqual Method

25.1.131.3.4 LoadFromReader Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.LoadFromReader Method

TnxIndexDescriptor Class

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxIndexDescriptor.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Methods > LoadFromReader Method

25.1.131.3.5 SaveStandalone Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.SaveStandalone Method

TnxIndexDescriptor Class

Pascal

```
public procedure SaveStandalone(
    aWriter: TnxWriter
);
```

Remarks

With this constructor an Index that has NO Data Dictionary as parent can be saved to a separate Writer.

See Also[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Methods > SaveStandalone Method

25.1.131.3.6 SaveToWriter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDescriptor.SaveToWriter Method**[TnxIndexDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxIndexDescriptor.

See Also[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Methods > SaveToWriter Method

25.1.131.Properties

25.1.131.4.1 Desc Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDescriptor.Desc Property**[TnxIndexDescriptor Class](#)**Pascal**

```
public property Desc: string;
```

Remarks

A description for the index.

See Also[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > Desc Property

25.1.131.4.2 Dups Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIndexDescriptor.Dups Property**[TnxIndexDescriptor Class](#)**Pascal**

```
public property Dups: Boolean;
```

Remarks

Set to true if the index allows duplicates.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > Dups Property

25.1.131.4.3 FileNumber Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDescriptor.FileNumber Property

[TnxIndexDescriptor Class](#)

Pascal

```
public property FileNumber: Integer;
```

Remarks

The file number where the index is stored.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > FileNumber Property

25.1.131.4.4 IndexEngine Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDescriptor.IndexEngine Property

[TnxIndexDescriptor Class](#)

Pascal

```
public property IndexEngine: string;
```

Remarks

The Name of the Index Engine used to create access the index.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > IndexEngine Property

25.1.131.4.5 IndexFile Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDescriptor.IndexFile Property

[TnxIndexDescriptor Class](#)

Pascal

```
public property IndexFile: TnxFileDescriptor;
```

Remarks

The File Descriptor if the Index is stored in an extra file.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > IndexFile Property

25.1.131.4.6 KeyDescriptor Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.KeyDescriptor Property

TnxIndexDescriptor Class

Pascal

```
public property KeyDescriptor: TnxBaseKeyDescriptor;
```

Remarks

The Key Descriptor used to create the record keys.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > KeyDescriptor Property

25.1.131.4.7 Name Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.Name Property

TnxIndexDescriptor Class

Pascal

```
public property Name: string;
```

Remarks

The name of the Index.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > Name Property

25.1.131.4.8 Number Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIndexDescriptor.Number Property

TnxIndexDescriptor Class

Pascal

```
public property Number: Integer;
```

Remarks

The number of the Index in the list of indices of the ParentDataDictionary.

See Also

[TnxIndexDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxIndexDescriptor Class > Properties > Number Property

25.1.13 TnxINIComponentConfiguration Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxINIComponentConfiguration Class

[nxConfigSettings](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxINIComponentConfiguration = class(TnxBaseComponentConfiguration);
```

File

[nxConfigSettings](#)

Remarks

This is class TnxINIComponentConfiguration.

See Also

[nxConfigSettings](#)

[Members](#)

[Methods](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxINIComponentConfiguration Class](#)

25.1.132 Constructors

25.1.132.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxINIComponentConfiguration.Create Constructor

[TnxINIComponentConfiguration Class](#)

Pascal

```
public constructor Create(
  aServerConfiguration: TnxINIServerConfiguration;
  aComponent: String
);
```

Remarks

This is Create, a member of class TnxINIComponentConfiguration.

See Also

[TnxINIComponentConfiguration Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxINIComponentConfiguration Class](#) > [Constructors](#) > [Create Constructor](#)

25.1.132.Destructors

25.1.132.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxINIComponentConfiguration.Destroy Destructor

[TnxINIComponentConfiguration Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxINIComponentConfiguration.

■ **See Also**

[TnxINIComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIComponentConfiguration Class > Destructors > Destroy Destructor

25.1.132.:Methods

25.1.132.3.1 GetValue Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxINIComponentConfiguration.GetValue Method

[TnxINIComponentConfiguration Class](#)

Pascal

```
public function GetValue(
    aName: String;
    aDefault: Variant
): Variant; override;
```

■ **Remarks**

This is GetValue, a member of class TnxINIComponentConfiguration.

■ **See Also**

[TnxINIComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIComponentConfiguration Class > Methods > GetValue Method

25.1.132.3.2 GetValueStream Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxINIComponentConfiguration.GetValueStream Method

[TnxINIComponentConfiguration Class](#)

Pascal

```
public procedure GetValueStream(
    aName: String;
    aStream: TStream
); override;
```

■ **Remarks**

This is GetValueStream, a member of class TnxINIComponentConfiguration.

■ **See Also**

[TnxINIComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIComponentConfiguration Class > Methods > GetValueStream Method

25.1.132.3.3 SetValue Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxINIComponentConfiguration.SetValue Method

[TnxINIComponentConfiguration Class](#)

Pascal

```
public procedure SetValue(
    aName: String;
    aValue: Variant
); override;
```

Remarks

This is SetValue, a member of class TnxINIComponentConfiguration.

See Also

[TnxINIComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIComponentConfiguration Class > Methods > SetValue Method

25.1.132.3.4 SetValueStream Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxINIComponentConfiguration.SetValueStream Method

[TnxINIComponentConfiguration Class](#)

Pascal

```
public procedure SetValueStream(
    aName: String;
    aStream: TStream
); override;
```

Remarks

This is SetValueStream, a member of class TnxINIComponentConfiguration.

See Also

[TnxINIComponentConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIComponentConfiguration Class > Methods > SetValueStream Method

25.1.13 TnxINIServerConfiguration Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxINIServerConfiguration Class

[nxConfigSettings](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxINIServerConfiguration = class(TnxBaseServerConfiguration);
```

File

[nxConfigSettings](#)

Remarks

This is class TnxINIServerConfiguration.

See Also

[nxConfigSettings](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxINIServerConfiguration Class

25.1.133. Constructors

25.1.133.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxINIConfiguration.Create Constructor

TnxINIConfiguration Class

Pascal

```
public constructor Create(
    aFileName: String
); override;
```

Remarks

This is Create, a member of class TnxINIConfiguration.

See Also

[TnxINIConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIConfiguration Class > Constructors > Create Constructor

25.1.133.2. Destructors

25.1.133.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxINIConfiguration.Destroy Destructor

TnxINIConfiguration Class

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxINIConfiguration.

See Also

[TnxINIConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIConfiguration Class > Destructors > Destroy Destructor

25.1.133.3. Methods

25.1.133.3.1 EnsureComponentConfiguration Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxINIConfiguration.EnsureComponentConfiguration Method

TnxINIConfiguration Class

Pascal

```
public procedure EnsureComponentConfiguration(
    aComponent: string;
    aSettings: TnxBaseSettings
); override;
```

Remarks

This is EnsureComponentConfiguration, a member of class TnxINIConfiguration.

See Also[TnxINIConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIConfiguration Class > Methods > EnsureComponentConfiguration Method

25.1.133.3.2 Flush Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxINIConfiguration.Flush Method**[TnxINIConfiguration Class](#)**Pascal**

```
public procedure Flush;
```

Remarks

This is Flush, a member of class TnxINIConfiguration.

See Also[TnxINIConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIConfiguration Class > Methods > Flush Method

25.1.133.3.3 GetComponentConfiguration Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxINIConfiguration.GetComponentConfiguration Method**[TnxINIConfiguration Class](#)**Pascal**

```
public function GetComponentConfiguration(
    aComponent: string
): TnxBaseComponentConfiguration; override;
```

Remarks

This is GetComponentConfiguration, a member of class TnxINIConfiguration.

See Also[TnxINIConfiguration Class](#)

You are here: Symbol Reference > Classes > TnxINIConfiguration Class > Methods > GetComponentConfiguration Method

25.1.13 TnxIPv4Transport Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxIPv4Transport Class**[nxtwWinsockTransport | Members | Methods](#)**Pascal**

```
public TnxIPv4Transport = class(TnxWinsockTransport);
```

File[nxtwWinsockTransport](#)

Remarks

IPv4 Transport

See Also

[nxtwWinsockTransport](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxIPv4Transport Class

25.1.134. Methods

25.1.134.1.1 ProtocolName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIPv4Transport.ProtocolName Method

TnxIPv4Transport Class

Pascal

```
public class function ProtocolName: string; override;
```

Remarks

This is ProtocolName, a member of class TnxIPv4Transport.

See Also

[TnxIPv4Transport Class](#)

You are here: Symbol Reference > Classes > TnxIPv4Transport Class > Methods > ProtocolName Method

25.1.135 TnxIterationList Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIterationList Class

[nxllComponent](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxIterationList = class(TnxSortedList);
```

File

[nxllComponent](#)

Remarks

A special list class used by Iteration methods.

See Also

[nxllComponent](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxIterationList Class

25.1.135. Methods

25.1.135.1.1 Add Method

NexusDB V2 VCL Reference

TnxIterationList.Add Method

[Contents](#) | [Index](#)

TnxIterationList Class

Pascal

```
public function Add(
    aItem: Pointer
): Integer; override;
```

Remarks

Adds a new pointer to the list

See Also

[TnxIterationList Class](#)

You are here: Symbol Reference > Classes > TnxIterationList Class > Methods > Add Method

25.1.135.1.2 LoadSettingsFromStream Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIterationList.LoadSettingsFromStream Method

TnxIterationList Class

Pascal

```
public procedure LoadSettingsFromStream(
    aReader: TnxReader
); virtual;
```

Remarks

Loads items from a stream into the list

See Also

[TnxIterationList Class](#)

You are here: Symbol Reference > Classes > TnxIterationList Class > Methods > LoadSettingsFromStream Method

25.1.135.1.3 SaveSettingsToStream Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxIterationList.SaveSettingsToStream Method

TnxIterationList Class

Pascal

```
public procedure SaveSettingsToStream(
    aWriter: TnxWriter
); virtual;
```

Remarks

Saves the items in the list to a stream

See Also

[TnxIterationList Class](#)

You are here: Symbol Reference > Classes > TnxIterationList Class > Methods > SaveSettingsToStream Method

25.1.13 TnxKeyAsVariantField Class

[NexusDB V2 VCL Reference](#)

TnxKeyAsVariantField Class

[Contents | Index](#)

[nxdb](#) | [Members](#)

Pascal

```
public TnxKeyAsVariantField = class(TVariantField);
```

File

[nxdb](#)

Remarks

Field class that can correctly get a nx internal keyvalue as variant.

See Also

[nxdb](#)
[Members](#)

You are here: Symbol Reference > Classes > TnxKeyAsVariantField Class

25.1.136 Constructors

25.1.136.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxKeyAsVariantField.Create Constructor

TnxKeyAsVariantField Class

Pascal

```
public constructor Create(
  AOwner: TComponent
); override;
```

Remarks

constructor

See Also

[TnxKeyAsVariantField Class](#)

You are here: Symbol Reference > Classes > TnxKeyAsVariantField Class > Constructors > Create Constructor

25.1.13 TnxKeyFieldDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxKeyFieldDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxKeyFieldDescriptor = class(TnxLocalizedDictionaryItem);
```

File

[nxsdDataDictionary](#)

■ Remarks

A Key Field Descriptor is used by a Key Descriptor to describe how field values processed when creating the key.

■ See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class

25.1.137.1.Destructors

25.1.137.1.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxKeyFieldDescriptor.Destroy Destructor

TnxKeyFieldDescriptor Class

Pascal

```
public destructor Destroy; override;
```

■ Remarks

This is Destroy, a member of class TnxKeyFieldDescriptor.

■ See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Destructors > Destroy Destructor

25.1.137.1.Methods

25.1.137.2.1 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxKeyFieldDescriptor.CheckValid Method

TnxKeyFieldDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

■ Remarks

This is CheckValid, a member of class TnxKeyFieldDescriptor.

■ See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Methods > CheckValid Method

25.1.137.2.2 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxKeyFieldDescriptor.AreEqual Method

TnxKeyFieldDescriptor Class

```
Pascal
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxKeyFieldDescriptor.

See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Methods > IsEqual Method

25.1.137.2.3 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxKeyFieldDescriptor.LoadFromReader Method

TnxKeyFieldDescriptor Class

```
Pascal
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxKeyFieldDescriptor.

See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Methods > LoadFromReader Method

25.1.137.2.4 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxKeyFieldDescriptor.SaveToWriter Method

TnxKeyFieldDescriptor Class

```
Pascal
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxKeyFieldDescriptor.

See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Methods > SaveToWriter Method

25.1.137.Properties

25.1.137.3.1 Ascend Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxKeyFieldDescriptor.Ascend Property

TnxKeyFieldDescriptor Class

Pascal

```
public property Ascend: Boolean;
```

Remarks

If true the keys should be sorted ascending otherwise descending.

See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > Ascend Property

25.1.137.3.2 Desc Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxKeyFieldDescriptor.Desc Property

TnxKeyFieldDescriptor Class

Pascal

```
public property Desc: string;
```

Remarks

Used to store the description for this descriptor.

See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > Desc Property

25.1.137.3.3 Field Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxKeyFieldDescriptor.Field Property

TnxKeyFieldDescriptor Class

Pascal

```
public property Field: TnxFieldDescriptor;
```

Remarks

This is a pointer to the associated Field descriptor.

See Also

[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > Field Property

25.1.137.3.4 FieldNumber Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxKeyFieldDescriptor.FieldNumber Property**[TnxKeyFieldDescriptor Class](#)**Pascal**

```
public property FieldNumber: Integer;
```

Remarks

This property returns the index of the associated field in the field list of the Data Dictionary.

See Also[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > FieldNumber Property

25.1.137.3.5 KeyChars Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxKeyFieldDescriptor.KeyChars Property**[TnxKeyFieldDescriptor Class](#)**Pascal**

```
public property KeyChars: Integer;
```

Remarks

Returns the number of chars in the key

See Also[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > KeyChars Property

25.1.137.3.6 KeyFieldEngine Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxKeyFieldDescriptor.KeyFieldEngine Property**[TnxKeyFieldDescriptor Class](#)**Pascal**

```
public property KeyFieldEngine: string;
```

Remarks

The name of the Key Field Engine to create the Keys for each field.

See Also[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > KeyFieldEngine Property

25.1.137.3.7 KeyLength Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxKeyFieldDescriptor.KeyLength Property**[TnxKeyFieldDescriptor Class](#)**Pascal**

```
public property KeyLength: Integer;
```

Remarks

Returns the length of the key

See Also[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > KeyLength Property

25.1.137.3.8 NullBehaviour Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxKeyFieldDescriptor.NullBehaviour Property**[TnxKeyFieldDescriptor Class](#)**Pascal**

```
public property NullBehaviour: TnxNullBehaviour;
```

Remarks

defines what happens with NULLs in the key.

See Also[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > NullBehaviour Property

25.1.137.3.9 OverrideLengthBytes Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxKeyFieldDescriptor.OverrideLengthBytes Property**[TnxKeyFieldDescriptor Class](#)**Pascal**

```
public property OverrideLengthBytes: Integer;
```

Remarks

???

See Also[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > OverrideLengthBytes Property

25.1.137.3.10 OverrideLengthChars Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxKeyFieldDescriptor.OverrideLengthChars Property**[TnxKeyFieldDescriptor Class](#)**Pascal**

```
public property OverrideLengthChars: Integer;
```

Remarks

???

See Also[TnxKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxKeyFieldDescriptor Class > Properties > [OverrideLengthChars Property](#)

25.1.13{TnxLocaleDescriptor Class[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor Class**[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxLocaleDescriptor = class(TnxDictionaryItem);
```

File[nxsdDataDictionary](#)**Remarks**

This is the basic locale descriptor. It can be added to the data dictionary, field descriptors and key field descriptors. If a key field descriptor has no locale descriptor attached falls back to the field descriptor. If the field descriptor has no locale descriptor attached it falls back to the data dictionary. If there is no locale descriptor available at all or if the Locale of the used locale descriptor is 0 all comparisons and sorting is performed by direct byte value comparison.

See Also
[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class

25.1.138.1Methods

25.1.138.1.1 CheckValid Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.CheckValid Method**[TnxLocaleDescriptor Class](#)**Pascal**

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxLocaleDescriptor.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Methods > CheckValid Method

25.1.138.1.2 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.IsEqual Method**[TnxLocaleDescriptor Class](#)**Pascal**

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxLocaleDescriptor.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Methods > IsEqual Method

25.1.138.1.3 LoadFromReader Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.LoadFromReader Method**[TnxLocaleDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxLocaleDescriptor.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Methods > LoadFromReader Method

25.1.138.1.4 SaveToWriter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.SaveToWriter Method**[TnxLocaleDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxLocaleDescriptor.

See Also

[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Methods > SaveToWriter Method

25.1.138.Properties

25.1.138.2.1 CompareCodepage Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocaleDescriptor.CompareCodepage Property

[TnxLocaleDescriptor Class](#)

Pascal

```
public property CompareCodepage: Word;
```

Remarks

Default Codepage derived from the Locale, used to compare strings. If this is different from StorageCodepage the strings need to be transformed before comparison.

See Also

[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > CompareCodepage Property

25.1.138.2.2 Flags Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocaleDescriptor.Flags Property

[TnxLocaleDescriptor Class](#)

Pascal

```
public property Flags: Cardinal;
```

Remarks

Returns the string flags as defined by the API (NORM_*)�.

See Also

[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > Flags Property

25.1.138.2.3 IgnoreKanaType Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocaleDescriptor.IgnoreKanaType Property

[TnxLocaleDescriptor Class](#)

Pascal

```
public property IgnoreKanaType: Boolean;
```

Remarks

Set to true, to not differentiate between Hiragana and Katakana characters. Corresponding Hiragana and Katakana characters compare as equal.

See Also

[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > [IgnoreKanaType Property](#)

25.1.138.2.4 IgnoreNonSpace Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocaleDescriptor.IgnoreNonSpace Property

[TnxLocaleDescriptor Class](#)

Pascal

```
public property IgnoreNonSpace: Boolean;
```

Remarks

Set to true to ignore non spacing characters.

See Also

[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > [IgnoreNonSpace Property](#)

25.1.138.2.5 IgnoreSymbols Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocaleDescriptor.IgnoreSymbols Property

[TnxLocaleDescriptor Class](#)

Pascal

```
public property IgnoreSymbols: Boolean;
```

Remarks

Set to true to ignore symbols.

See Also

[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > [IgnoreSymbols Property](#)

25.1.138.2.6 IgnoreWidth Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocaleDescriptor.IgnoreWidth Property

[TnxLocaleDescriptor Class](#)

Pascal

```
public property IgnoreWidth: Boolean;
```

Remarks

Set to true to differentiate between a single-byte character and the same character as a double-byte character.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > IgnoreWidth Property

25.1.138.2.7 Locale Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.Locale Property**[TnxLocaleDescriptor Class](#)**Pascal**

```
public property Locale: LCID;
```

Remarks

This specifies the Locale ID to use. Use LOCALE_SYSTEM_DEFAULT or LOCALE_USER_DEFAULT for the defaults or create an id by using MAKELCID. The value is converted into a specific locale via ConvertDefaultLocale on assignment.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > Locale Property

25.1.138.2.8 OverrideStorageCodepage Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.OverrideStorageCodepage Property**[TnxLocaleDescriptor Class](#)**Pascal**

```
public property OverrideStorageCodepage: Word;
```

Remarks

Set to a valid codepage to store strings in a non default codepage. e.g. CP_UTF8. To store strings in the default codepage for the locale use 0.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > OverrideStorageCodepage Property

25.1.138.2.9 StorageCodepage Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.StorageCodepage Property**[TnxLocaleDescriptor Class](#)**Pascal**

```
public property StorageCodepage: Word;
```

Remarks

Codepage for storing strings in record buffers.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > StorageCodepage Property

25.1.138.2.10 UseStringSort Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocaleDescriptor.UseStringSort Property**[TnxLocaleDescriptor Class](#)**Pascal**

```
public property UseStringSort: Boolean;
```

Remarks

Set to true to treat punctuation the same as symbols.

See Also[TnxLocaleDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxLocaleDescriptor Class > Properties > UseStringSort Property

25.1.13 TnxLocalizedDictionaryItem Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocalizedDictionaryItem Class**[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxLocalizedDictionaryItem = class(TnxDictionaryItem);
```

File[nxsdDataDictionary](#)**Remarks**

A dictionary item with Locale Descriptors for localization of strings.

See Also[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class

25.1.139.1.1.Destructors

25.1.139.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocalizedDictionaryItem.Destroy Destructor**[TnxLocalizedDictionaryItem Class](#)**Pascal**

```
public destructor Destroy; override;
```

■ Remarks

This is Destroy, a member of class TnxLocalizedDictionaryItem.

■ See Also

[TnxLocalizedDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Destructors > [Destroy Destructor](#)

25.1.139. Methods

25.1.139.2.1 AddLocaleDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocalizedDictionaryItem.AddLocaleDescriptor Method

[TnxLocalizedDictionaryItem Class](#) | [Related Topics](#)

Pascal

```
public function AddLocaleDescriptor(
  aLocaleClass: TnxLocaleDescriptorClass = nil
): TnxLocaleDescriptor;
```

■ Remarks

This function sets the Locale Descriptor class to the given aLocaleClass, creates and instance, adds it to the data dictionary and returns the instance.

■ Related Topics

[AddLocaleDescriptor](#), [RemoveLocaleDescriptor](#), [LocaleDescriptor](#)

■ See Also

[TnxLocalizedDictionaryItem Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Methods > [AddLocaleDescriptor Method](#)

25.1.139.2.2 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLocalizedDictionaryItem.CheckValid Method

[TnxLocalizedDictionaryItem Class](#)

Pascal

```
public procedure CheckValid; override;
```

■ Remarks

This is CheckValid, a member of class TnxLocalizedDictionaryItem.

■ See Also

[TnxLocalizedDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Methods > [CheckValid Method](#)

25.1.139.2.3 FindRelatedDescriptorOfType Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocalizedDictionaryItem.FindRelatedDescriptorOfType Method**[TnxLocalizedDictionaryItem Class](#)**Pascal**

```
public function FindRelatedDescriptorOfType(
    aType: TnxDictionaryItemClass;
    out aDescriptor;
    aFrom: TnxDictionaryItem = nil
): Boolean; override;
```

Remarks

This is FindRelatedDescriptorOfType, a member of class TnxLocalizedDictionaryItem.

See Also[TnxLocalizedDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Methods > [FindRelatedDescriptorOfType Method](#)

25.1.139.2.4 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocalizedDictionaryItem.IsEqual Method**[TnxLocalizedDictionaryItem Class](#)**Pascal**

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxLocalizedDictionaryItem.

See Also[TnxLocalizedDictionaryItem Class](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Methods > [IsEqual Method](#)

25.1.139.2.5 RemoveLocaleDescriptor Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxLocalizedDictionaryItem.RemoveLocaleDescriptor Method**[TnxLocalizedDictionaryItem Class](#) | [Related Topics](#)**Pascal**

```
public procedure RemoveLocaleDescriptor;
```

Remarks

This function frees the Instance of the associated Locale Descriptor.

Related Topics[AddLocaleDescriptor](#), [RemoveLocaleDescriptor](#), [LocaleDescriptor](#)

See Also

[TnxLocalizedDictionaryItem Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Methods > RemoveLocaleDescriptor Method

25.1.139.Properties

25.1.139.3.1 LocaleDescriptor Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxLocalizedDictionaryItem.LocaleDescriptor Property

[TnxLocalizedDictionaryItem Class](#) | [Related Topics](#)

Pascal

```
public property LocaleDescriptor: TnxLocaleDescriptor;
```

Remarks

This property returns the associated Locale Descriptor.

Related Topics

[AddLocaleDescriptor](#), [RemoveLocaleDescriptor](#), [LocaleDescriptor](#)

See Also

[TnxLocalizedDictionaryItem Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Properties > [LocaleDescriptor Property](#)

25.1.139.3.2 UsedLocaleDescriptor Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxLocalizedDictionaryItem.UsedLocaleDescriptor Property

[TnxLocalizedDictionaryItem Class](#) | [Related Topics](#)

Pascal

```
public property UsedLocaleDescriptor: TnxLocaleDescriptor;
```

Remarks

This property returns the Locale Descriptor currently in use.

Related Topics

[AddLocaleDescriptor](#), [RemoveLocaleDescriptor](#), [LocaleDescriptor](#)

See Also

[TnxLocalizedDictionaryItem Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Properties > [UsedLocaleDescriptor Property](#)

25.1.139.3.3 UsedStorageCodePage Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxLocalizedDictionaryItem.UsedStorageCodePage Property**[TnxLocalizedDictionaryItem Class](#) | [Related Topics](#)**Pascal**

```
public property UsedStorageCodePage: Word;
```

Remarks

This property returns the StorageCodePage of the Locale Descriptor currently in use or 0.

Related Topics

[AddLocaleDescriptor](#), [RemoveLocaleDescriptor](#), [LocaleDescriptor](#)

See Also

[TnxLocalizedDictionaryItem Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxLocalizedDictionaryItem Class > Properties > UsedStorageCodePage Property

25.1.14(TnxLockedOptionsExtendableServerObject Class)*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxLockedOptionsExtendableServerObject Class**[nxsdServerEngine](#) | [Members](#) | [Methods](#)**Pascal**

```
public TnxLockedOptionsExtendableServerObject = class(TnxExtendableServerObject);
```

File

[nxsdServerEngine](#)

Remarks

This is class TnxLockedOptionsExtendableServerObject.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxLockedOptionsExtendableServerObject Class

25.1.140.'Destructors

25.1.140.1.1 Destroy Destructor

NexusDB V2 VCL Reference[Contents | Index](#)**TnxLockedOptionsExtendableServerObject.Destroy Destructor**[TnxLockedOptionsExtendableServerObject Class](#)**Pascal**

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxLockedOptionsExtendableServerObject.

See Also

[TnxLockedOptionsExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxLockedOptionsExtendableServerObject Class > Constructors > Destroy Destructor

25.1.140. Methods

25.1.140.2.1 [esoOptionClear Method](#)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLockedOptionsExtendableServerObject.esoOptionClear Method

[TnxLockedOptionsExtendableServerObject Class](#)

Pascal

```
public function esoOptionClear(
  const aName: string
): TnxResult; override;
```

Remarks

This is [esoOptionClear](#), a member of class [TnxLockedOptionsExtendableServerObject](#).

See Also

[TnxLockedOptionsExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxLockedOptionsExtendableServerObject Class > Methods > [esoOptionClear Method](#)

25.1.140.2.2 [esoOptionSet Method](#)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLockedOptionsExtendableServerObject.esoOptionSet Method

[TnxLockedOptionsExtendableServerObject Class](#)

Pascal

```
public function esoOptionSet(
  const aName: string;
  const aValue: string
): TnxResult; override;
```

Remarks

This is [esoOptionSet](#), a member of class [TnxLockedOptionsExtendableServerObject](#).

See Also

[TnxLockedOptionsExtendableServerObject Class](#)

You are here: Symbol Reference > Classes > TnxLockedOptionsExtendableServerObject Class > Methods > [esoOptionSet Method](#)

25.1.14 TnxLoggableComponent Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLoggableComponent Class

[nxIComponent](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxLoggableComponent = class(TnxRegisterableComponent);
```

File[nxllComponent](#)**Remarks**

A component class that has integrated loggin support.

See Also[nxllComponent](#)[Members](#)[Methods](#)[Properties](#)

You are here: Symbol Reference > Classes > TnxLoggableComponent Class

25.1.141. Methods

25.1.141.1.1 IcLog

25.1.141.1.1.1 IcLog Method (array of string)

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxLoggableComponent.IcLog Method (array of string)

TnxLoggableComponent Class

Pascal

```
public procedure IcLog(
  const aMsgs: array of string
); virtual; overload;
```

Remarks

Log a number of strings.

See Also[TnxLoggableComponent Class](#)

You are here: Symbol Reference > Classes > TnxLoggableComponent Class > Methods > IcLog > IcLog Method (array of string)

25.1.141.1.1.2 IcLog Method (string)

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxLoggableComponent.IcLog Method (string)

TnxLoggableComponent Class

Pascal

```
public procedure IcLog(
  const aMsg: string
); virtual; overload;
```

Remarks

Log a string.

See Also[TnxLoggableComponent Class](#)

You are here: Symbol Reference > Classes > TnxLoggableComponent Class > Methods > IcLog > IcLog Method (string)

25.1.141.1.1.3 IcLog Method (string, , TnxMemSize)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxLoggableComponent.IcLog Method (string, , TnxMemSize)

TnxLoggableComponent Class

Pascal

```
public procedure IcLog(
  const aMsg: string;
  const aData;
  aDataLen: TnxMemSize
); virtual; overload;
```

Remarks

Log a string and Memory Block.

See Also

[TnxLoggableComponent Class](#)

You are here: Symbol Reference > Classes > TnxLoggableComponent Class > Methods > IcLog > IcLog Method (string, , TnxMemSize)

25.1.141.1.1.4 IcLog Method (string, array of const)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxLoggableComponent.IcLog Method (string, array of const)

TnxLoggableComponent Class

Pascal

```
public procedure IcLog(
  const aMsg: string;
  const aArgs: array of const
); virtual; overload;
```

Remarks

Log a string.

See Also

[TnxLoggableComponent Class](#)

You are here: Symbol Reference > Classes > TnxLoggableComponent Class > Methods > IcLog > IcLog Method (string, array of const)

25.1.141.Properties

25.1.141.2.1 EventLog Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxLoggableComponent.EventLog Property

TnxLoggableComponent Class

Pascal

```
published property EventLog: TnxBaseLog;
```

Remarks

The Event log associated with this component. If none set the IcLog calls will do nothing.

See Also

[TnxLoggableComponent Class](#)

You are here: Symbol Reference > Classes > TnxLoggableComponent Class > Properties > EventLog Property

25.1.141.2.2 EventLogEnabled Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLoggableComponent.EventLogEnabled Property

[TnxLoggableComponent Class](#)

Pascal

```
published property EventLogEnabled: Boolean;
```

Remarks

True if logging is enabled for this component.

See Also

[TnxLoggableComponent Class](#)

You are here: Symbol Reference > Classes > TnxLoggableComponent Class > Properties > EventLogEnabled Property

25.1.14:TnxMainIndicesDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMainIndicesDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxMainIndicesDescriptor = class(TnxBaseIndicesDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

Holds a list of IndexDescriptors that are main indices. Main indices are user selectable indices (e.g. IndexName property) as opposed to internally used indices.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxMainIndicesDescriptor Class

25.1.142.Methods

25.1.142.1.1 CheckValid Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMainIndicesDescriptor.CheckValid Method

TnxMainIndicesDescriptor Class

```
Pascal
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxMainIndicesDescriptor.

See Also

[TnxMainIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMainIndicesDescriptor Class > Methods > CheckValid Method

25.1.142.1.2 Clear Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMainIndicesDescriptor.Clear Method**TnxMainIndicesDescriptor Class**

```
Pascal
public procedure Clear(
    aNotify: Boolean
); override;
```

Remarks

Clears the list of descriptors.

See Also

[TnxMainIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMainIndicesDescriptor Class > Methods > Clear Method

25.1.142.1.3 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMainIndicesDescriptor.AreEqual Method**TnxMainIndicesDescriptor Class**

```
Pascal
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxMainIndicesDescriptor.

See Also

[TnxMainIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMainIndicesDescriptor Class > Methods > IsEqual Method

25.1.142.1.4 LoadFromReader Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxMainIndicesDescriptor.LoadFromReader Method**[TnxMainIndicesDescriptor Class](#)**Pascal**

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxMainIndicesDescriptor.

See Also[TnxMainIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMainIndicesDescriptor Class > Methods > [LoadFromReader Method](#)

25.1.142.1.5 SaveToWriter Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxMainIndicesDescriptor.SaveToWriter Method**[TnxMainIndicesDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxMainIndicesDescriptor.

See Also[TnxMainIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMainIndicesDescriptor Class > Methods > [SaveToWriter Method](#)

25.1.142.Properties

25.1.142.2.1 DefaultIndex Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxMainIndicesDescriptor.DefaultIndex Property**[TnxMainIndicesDescriptor Class](#)**Pascal**

```
public property DefaultIndex: Integer;
```

Remarks

The number of the Default index used for the table.

See Also[TnxMainIndicesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMainIndicesDescriptor Class > Properties > DefaultIndex Property

25.1.14: TnxMemoBlobStream Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemoBlobStream Class

[nxdb](#) | [Members](#) | [Methods](#) | [Related Topics](#)

Pascal

```
public TnxMemoBlobStream = class(TnxBaseBlobStream);
```

File

[nxdb](#)

Remarks

This class is used to access memo blobs if Transliterate is true.

Never create instances directly! Always use TnxDataSet.CreateBlobStream, which will create the right type of blob stream based on the state of the dataset.

The blob stream should be freed again as soon as possible. It must be freed before calling Post or moving to another record.

Related Topics

[TnxDataSet.CreateBlobStream](#) [TnxBaseBlobStream](#) [TnxBlobStream](#) [TnxBlockModeBlobStream](#)

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxMemoBlobStream Class

25.1.143: Destructors

25.1.143.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemoBlobStream.Destroy Destructor

[TnxMemoBlobStream Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxMemoBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxMemoBlobStream Class > Destructors > Destroy Destructor

25.1.143. Methods

25.1.143.2.1 Read Method

NexusDB V2 VCL Reference

TnxMemoBlobStream.Read Method

[Contents](#) | [Index](#)

TnxMemoBlobStream Class

Pascal

```
public function Read(
  var aBuffer;
  aCount: Integer
): Integer; override;
```

Parameters

Parameters	Description
aBuffer	variable to store the read bytes
aCount	number of bytes to read

Remarks

With this method you can read a number of bytes of the blob data.

See Also

[TnxMemoBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxMemoBlobStream Class > Methods > Read Method

25.1.143.2.2 Seek Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMemoBlobStream.Seek Method

TnxMemoBlobStream Class

Pascal

```
public function Seek(
  aOffset: Integer;
  aOrigin: Word
): Integer; override;
```

Parameters

Parameters	Description
aOffset	the offset in bytes to move the pointer
aOrigin	the origin of the aOffset

Remarks

Seek sets the current position in the blob stream.

See Also

[TnxMemoBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxMemoBlobStream Class > Methods > Seek Method

25.1.143.2.3 Truncate Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMemoBlobStream.Truncate Method

TnxMemoBlobStream Class

```
Pascal
public procedure Truncate; override;
```

Remarks

This method truncates the blob data at the current position.

See Also

[TnxMemoBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxMemoBlobStream Class > Methods > Truncate Method

25.1.143.2.4 Write Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemoBlobStream.Write Method**TnxMemoBlobStream Class**

```
Pascal
public function Write(
    const aBuffer;
    aCount: Integer
): Integer; override;
```

Parameters

Parameters	Description
aBuffer	data to be written
aCount	number of bytes to write

Remarks

Use Write to write a number of bytes to the blob at the current position. Take care to not accidentally overwrite data by not positioning with Seek first.

See Also

[TnxMemoBlobStream Class](#)

You are here: Symbol Reference > Classes > TnxMemoBlobStream Class > Methods > Write Method

25.1.14 TnxMemoField Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemoField Class

[nxdb](#) | [Members](#) | [Properties](#)

```
Pascal
public TnxMemoField = class(TMemoField);
```

File

[nxdb](#)

Remarks

Extended field class for memo fields.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxMemoField Class

25.1.144. Properties

25.1.144.1.1 AsWideString Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMemoField.AsWideString Property

[TnxMemoField Class](#)

Pascal

```
public property AsWideString: WideString;
```

Remarks

This is AsWideString, a member of class TnxMemoField.

See Also

[TnxMemoField Class](#)

You are here: Symbol Reference > Classes > TnxMemoField Class > Properties > AsWideString Property

25.1.145!TnxMemTable Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMemTable Class

[Events](#) | [nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxMemTable = class(TnxIndexDataSet);
```

File

[nxdb](#)

Remarks

This is class TnxMemTable.

See Also

[Events](#)
[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxMemTable Class

25.1.145. Constructors

25.1.145.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMemTable.Create Constructor

[TnxMemTable Class](#)

Pascal

```
public constructor Create(
    AOwner: TComponent
) ; override;
```

Remarks

constructor.

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Constructors > Create Constructor

25.1.145.1.Destructors

25.1.145.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.Destroy Destructor

[TnxMemTable Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Destructors > Destroy Destructor

25.1.145.1.Events

25.1.145.3.1 OnCreateTable Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.OnCreateTable Property

[TnxMemTable Class](#)

Pascal

```
published property OnCreateTable: TnxCreateTableEvent;
```

Remarks

This event is fired before the table is created. It allows modification of the data dictionary that will be used to create the table.

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Events > OnCreateTable Property

25.1.145.4 Properties

25.1.145.4.1 FieldDefs Property

NexusDB V2 VCL Reference
TnxMemTable.FieldDefs Property

TnxMemTable Class

Pascal

```
published property FieldDefs;
```

Remarks

see tTable.FieldDefs

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > FieldDefs Property

25.1.145.4.2 IndexDefs Property

NexusDB V2 VCL Reference
TnxMemTable.IndexDefs Property

TnxMemTable Class

Pascal

```
published property IndexDefs;
```

Remarks

see tTable.IndexDefs.

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > IndexDefs Property

25.1.145.4.3 IndexFieldCount Property

NexusDB V2 VCL Reference
TnxMemTable.IndexFieldCount Property

[Contents | Index](#)

TnxMemTable Class

Pascal

```
public property IndexFieldCount;
```

Remarks

see tTable.IndexFieldCount.

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > IndexFieldCount Property

25.1.145.4.4 IndexFieldNames Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxMemTable.IndexFieldNames Property**[TnxMemTable Class](#)**Pascal**

```
published property IndexFieldNames;
```

Remarks

see tTable.IndexFieldNames

See Also[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > IndexFieldNames Property

25.1.145.4.5 IndexFields Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxMemTable.IndexFields Property**[TnxMemTable Class](#)**Pascal**

```
public property IndexFields;
```

Remarks

see tTable.IndexFields

See Also[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > IndexFields Property

25.1.145.4.6 IndexName Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxMemTable.IndexName Property**[TnxMemTable Class](#)**Pascal**

```
published property IndexName;
```

Remarks

see tTable.IndexName

See Also[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > IndexName Property

25.1.145.4.7 KeyExclusive Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxMemTable.KeyExclusive Property**

[TnxMemTable Class](#)

```
Pascal
public property KeyExclusive;
```

Remarks

see tTable.KeyExclusive

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > KeyExclusive Property

25.1.145.4.8 KeyFieldCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.KeyFieldCount Property[TnxMemTable Class](#)

```
Pascal
public property KeyFieldCount;
```

Remarks

see tTable.KeyFieldCount

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > KeyFieldCount Property

25.1.145.4.9 KeyPartialLen Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.KeyPartialLen Property[TnxMemTable Class](#)

```
Pascal
public property KeyPartialLen;
```

Remarks

see tTable.KeyPartialLen

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > KeyPartialLen Property

25.1.145.4.10 MasterFields Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.MasterFields Property[TnxMemTable Class](#)

```
Pascal
```

```
published property MasterFields;
```

Remarks

see tTable.MasterFields

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > MasterFields Property

25.1.145.4.11 MasterSource Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.MasterSource Property

[TnxMemTable Class](#)

Pascal

```
published property MasterSource;
```

Remarks

see tTable.MasterSource

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > MasterSource Property

25.1.145.4.12 StoreDefs Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.StoreDefs Property

[TnxMemTable Class](#)

Pascal

```
published property StoreDefs: Boolean;
```

Remarks

see tTable.StoreDefs

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > StoreDefs Property

25.1.145.4.13 TableName Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.TableName Property

[TnxMemTable Class](#)

Pascal

```
published property TableName: string;
```

Remarks

Optional. When specified the name must be unique in the database. If a name is specified it is possible to access the mem table using other TnxTable and TnxQuery components connected to the same database under this name while the mem table is active.

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > TableName Property

25.1.145.4.14 UsedTableName Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMemTable.UsedTableName Property

[TnxMemTable Class](#)

Pascal

```
published property UsedTableName: string;
```

Remarks

Only valid while the table is active. If a TableName was specified UsedTableName will be equal to TableName. Otherwise it will contain the unique name that was created for this mem table. Other TnxTable and TnxQuery components connected to the same database will be able to access the mem table under this name while it is active.

See Also

[TnxMemTable Class](#)

You are here: Symbol Reference > Classes > TnxMemTable Class > Properties > UsedTableName Property

25.1.14(TnxMinMaxValidationDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMinMaxValidationDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxMinMaxValidationDescriptor = class(TnxBaseFieldValidationDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

A validation descriptor to check for Minimum/Maximum values

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class

25.1.146.1.Destructors

25.1.146.1.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMinMaxValidationDescriptor.Destroy Destructor

TnxMinMaxValidationDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxMinMaxValidationDescriptor.

See Also

[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Destructors > Destroy Destructor

25.1.146.2.Methods

25.1.146.2.1 AddKeyField Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMinMaxValidationDescriptor.AddKeyField Method

TnxMinMaxValidationDescriptor Class

Pascal

```
public function AddKeyField(
  aClass: TnxKeyFieldDescriptorClass = nil
): TnxKeyFieldDescriptor;
```

Remarks

Associates the descriptor with a Field.

See Also

[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Methods > AddKeyField Method

25.1.146.2.2 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMinMaxValidationDescriptor.CheckValid Method

TnxMinMaxValidationDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxMinMaxValidationDescriptor.

See Also

[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Methods > CheckValid Method

25.1.146.2.3 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMinMaxValidationDescriptor.IsEqual Method

TnxMinMaxValidationDescriptor Class

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxMinMaxValidationDescriptor.

See Also

[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Methods > IsEqual Method

25.1.146.2.4 LoadFromReader Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMinMaxValidationDescriptor.LoadFromReader Method

TnxMinMaxValidationDescriptor Class

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxMinMaxValidationDescriptor.

See Also

[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Methods > LoadFromReader Method

25.1.146.2.5 RemoveKeyField Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxMinMaxValidationDescriptor.RemoveKeyField Method

TnxMinMaxValidationDescriptor Class

Pascal

```
public procedure RemoveKeyField;
```

Remarks

Removes the field association iow disables the validation.

See Also

[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Methods > RemoveKeyField Method

25.1.146.2.6 SaveToWriter Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxMinMaxValidationDescriptor.SaveToWriter Method**[TnxMinMaxValidationDescriptor Class](#)**Pascal**

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxMinMaxValidationDescriptor.

See Also[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Methods > SaveToWriter Method

25.1.146.Properties

25.1.146.3.1 KeyField Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxMinMaxValidationDescriptor.KeyField Property**[TnxMinMaxValidationDescriptor Class](#)**Pascal**

```
public property KeyField: TnxKeyFieldDescriptor;
```

Remarks

Returns the descriptor of the associated Key field.

See Also[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Properties > KeyField Property

25.1.146.3.2 Max Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxMinMaxValidationDescriptor.Max Property**[TnxMinMaxValidationDescriptor Class](#)**Pascal**

```
public property Max: TnxDynByteArray;
```

Remarks

Gets the maximum value as raw data.

See Also[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Properties > Max Property

25.1.146.3.3 MaxAsVariant Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxMinMaxValidationDescriptor.MaxAsVariant Property**[TnxMinMaxValidationDescriptor Class](#)**Pascal**

```
public property MaxAsVariant: OleVariant;
```

Remarks

Gets/Sets the maximum value as OleVariant.

See Also[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Properties > MaxAsVariant Property

25.1.146.3.4 Min Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxMinMaxValidationDescriptor.Min Property**[TnxMinMaxValidationDescriptor Class](#)**Pascal**

```
public property Min: TnxDynByteArray;
```

Remarks

Gets the minimum value as raw data.

See Also[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Properties > Min Property

25.1.146.3.5 MinAsVariant Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxMinMaxValidationDescriptor.MinAsVariant Property**[TnxMinMaxValidationDescriptor Class](#)**Pascal**

```
public property MinAsVariant: OleVariant;
```

Remarks

Gets/Sets the minimum value as OleVariant.

See Also[TnxMinMaxValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxMinMaxValidationDescriptor Class > Properties > MinAsVariant Property

25.1.14 TnxNestedTableDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNestedTableDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxNestedTableDescriptor = class(TnxBaseTableDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

A nested (sub) Table Descriptor is a special descriptor that is used for defining the sub tables of a certain main table.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxNestedTableDescriptor Class

25.1.147 Constructors

25.1.147.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNestedTableDescriptor.Create Constructor

[TnxNestedTableDescriptor Class](#)

Pascal

```
public constructor Create(
  aParent: TPersistent;
  const aName: string
); virtual; reintroduce;
```

Remarks

constructor - needs the main table as parent and it's Name

See Also

[TnxNestedTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNestedTableDescriptor Class > Constructors > Create Constructor

25.1.147 Properties

25.1.147.2.1 Name Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNestedTableDescriptor.Name Property

[TnxNestedTableDescriptor Class](#)

Pascal

```
public property Name: string;
```

Remarks

Returns the Name of the table.

See Also

[TnxNestedTableDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNestedTableDescriptor Class > Properties > Name Property

25.1.14{TnxNoChangeValidationDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNoChangeValidationDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxNoChangeValidationDescriptor = class(TnxBaseFieldValidationDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

A validation descriptor to disallow field changes once it was initially set.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class

25.1.148.Destructors

25.1.148.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNoChangeValidationDescriptor.Destroy Destructor

[TnxNoChangeValidationDescriptor Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxNoChangeValidationDescriptor.

See Also

[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Destructors > Destroy Destructor

25.1.148. Methods

25.1.148.2.1 AddKeyField Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxNoChangeValidationDescriptor.AddKeyField Method

TnxNoChangeValidationDescriptor Class

Pascal

```
public function AddKeyField(
    aClass: TnxKeyFieldDescriptorClass = nil
): TnxKeyFieldDescriptor;
```

Remarks

Associates the descriptor with a Field.

See Also

[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Methods > AddKeyField Method

25.1.148.2.2 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxNoChangeValidationDescriptor.CheckValid Method

TnxNoChangeValidationDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxNoChangeValidationDescriptor.

See Also

[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Methods > CheckValid Method

25.1.148.2.3 IsEqual Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxNoChangeValidationDescriptor.AreEqual Method

TnxNoChangeValidationDescriptor Class

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxNoChangeValidationDescriptor.

See Also

[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Methods > IsEqual Method

25.1.148.2.4 LoadFromReader Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxNoChangeValidationDescriptor.LoadFromReader Method

TnxNoChangeValidationDescriptor Class

Pascal

```
public procedure LoadFromReader(
  aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxNoChangeValidationDescriptor.

See Also

[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Methods > LoadFromReader Method

25.1.148.2.5 RemoveKeyField Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxNoChangeValidationDescriptor.RemoveKeyField Method

TnxNoChangeValidationDescriptor Class

Pascal

```
public procedure RemoveKeyField;
```

Remarks

Removes the field association iow disables the validation.

See Also

[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Methods > RemoveKeyField Method

25.1.148.2.6 SaveToWriter Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxNoChangeValidationDescriptor.SaveToWriter Method

TnxNoChangeValidationDescriptor Class

Pascal

```
public procedure SaveToWriter(
  aWriter: TnxWriter;
  aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxNoChangeValidationDescriptor.

See Also[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Methods > [SaveToWriter Method](#)

25.1.148.Properties

25.1.148.3.1 KeyField Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxNoChangeValidationDescriptor.KeyField Property

[TnxNoChangeValidationDescriptor Class](#)**Pascal**

```
public property KeyField: TnxKeyFieldDescriptor;
```

Remarks

Returns the descriptor of the associated Key field.

See Also[TnxNoChangeValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoChangeValidationDescriptor Class > Properties > [KeyField Property](#)

25.1.149.TnxNotNullCompKeyDescriptor Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxNotNullCompKeyDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#)**Pascal**

```
public TnxNotNullCompKeyDescriptor = class(TnxCompKeyDescriptor);
```

File[nxsdDataDictionary](#)**Remarks**

Key descriptor used to ensure a field is NOT null.

See Also[nxsdDataDictionary](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxNotNullCompKeyDescriptor Class

25.1.149.Methods

25.1.149.1.1 CheckValid Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxNotNullCompKeyDescriptor.CheckValid Method

[TnxNotNullCompKeyDescriptor Class](#)

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxNoNullCompKeyDescriptor.

See Also

[TnxNoNullCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoNullCompKeyDescriptor Class > Methods > CheckValid Method

25.1.149.1.2 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNoNullCompKeyDescriptor.LoadFromReader Method

[TnxNoNullCompKeyDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxNoNullCompKeyDescriptor.

See Also

[TnxNoNullCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoNullCompKeyDescriptor Class > Methods > LoadFromReader Method

25.1.149.1.3 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNoNullCompKeyDescriptor.SaveToWriter Method

[TnxNoNullCompKeyDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxNoNullCompKeyDescriptor.

See Also

[TnxNoNullCompKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxNoNullCompKeyDescriptor Class > Methods > SaveToWriter Method

25.1.15(TnxNullActiveBroadcast Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNullActiveBroadcast Class

[nxIITransport](#)**Pascal**

```
public TnxNullActiveBroadcast = class(TnxInterfacedObject, InxActiveBroadcast);
```

File[nxIITransport](#)**Remarks**

This is class TnxNullActiveBroadcast.

See Also[nxIITransport](#)

You are here: Symbol Reference > Classes > TnxNullActiveBroadcast Class

25.1.15 TnxPersistent Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxPersistent Class

[nxIComponent](#) | [Members](#) | [Methods](#)**Pascal**

```
public TnxPersistent = class(TPersistent, InxInterface);
```

File[nxIComponent](#)**Remarks**

Base class for all persistent NexusDB objects.

See Also[nxIComponent](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxPersistent Class

25.1.151 Methods

25.1.151.1 FreeInstance Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxPersistent.FreeInstance Method

[TnxPersistent Class](#)**Pascal**

```
public procedure FreeInstance; override;
```

Remarks

Overidden to take advantage of the NexusDB Memory Manager.

See Also[TnxPersistent Class](#)

You are here: Symbol Reference > Classes > TnxPersistent Class > Methods > FreeInstance Method

25.1.151.1.2 New Instance Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxPersistent.NewInstance Method

TnxPersistent Class

Pascal

```
public class function NewInstance: TObject; override;
```

Remarks

Overidden to take advatange of the NexusDB Memory Manager.

See Also

[TnxPersistent Class](#)

You are here: Symbol Reference > Classes > TnxPersistent Class > Methods > NewInstance Method

25.1.15 TnxPooledSession Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxPooledSession Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxPooledSession = class(TnxBaseSession);
```

File

[nxdb](#)

Remarks

This is class TnxPooledSession.

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxPooledSession Class

25.1.152.1.1 Destroy Destructor

25.1.152.1.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxPooledSession.Destroy Destructor

TnxPooledSession Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxPooledSession Class](#)

You are here: Symbol Reference > Classes > TnxPooledSession Class > Destructors > Destroy Destructor

25.1.152.:Methods

25.1.152.2.1 AddRef Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxPooledSession.AddRef Method

TnxPooledSession Class

Pascal

```
public procedure AddRef;
```

Remarks

This is AddRef, a member of class TnxPooledSession.

See Also

[TnxPooledSession Class](#)

You are here: Symbol Reference > Classes > TnxPooledSession Class > Methods > AddRef Method

25.1.152.2.2 Release Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxPooledSession.Release Method

TnxPooledSession Class

Pascal

```
public procedure Release;
```

Remarks

This is Release, a member of class TnxPooledSession.

See Also

[TnxPooledSession Class](#)

You are here: Symbol Reference > Classes > TnxPooledSession Class > Methods > Release Method

25.1.152.:Properties

25.1.152.3.1 SessionPool Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxPooledSession.SessionPool Property

TnxPooledSession Class

Pascal

```
published property SessionPool: TnxBaseSessionPool;
```

Remarks

This is SessionPool, a member of class TnxPooledSession.

See Also

[TnxPooledSession Class](#)

You are here: Symbol Reference > Classes > TnxPooledSession Class > Properties > SessionPool Property

25.1.15:TnxQuery Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxQuery Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxQuery = class(TnxStatementDataSet);
```

File

[nxdb](#)

Remarks

This is the client side representation of a SQL query. It's very similar to the VCL TQuery class.

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxQuery Class

25.1.153.:Constructors

25.1.153.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxQuery.Create Constructor

TnxQuery Class

Pascal

```
public constructor Create(
    aOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxQuery Class](#)

You are here: Symbol Reference > Classes > TnxQuery Class > Constructors > Create Constructor

25.1.153.:Destructors

25.1.153.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxQuery.Destroy Destructor

TnxQuery Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

TnxQuery Class

You are here: Symbol Reference > Classes > TnxQuery Class > Destructors > Destroy Destructor

25.1.153.3 Methods

25.1.153.3.1 ExecSQL Method

NexusDB V2 VCL Reference

TnxQuery.ExecSQL Method

[Contents](#) | [Index](#)

TnxQuery Class

Pascal

```
public procedure ExecSQL;
```

Remarks

see tQuery.ExecSql

See Also

TnxQuery Class

You are here: Symbol Reference > Classes > TnxQuery Class > Methods > ExecSQL Method

25.1.153.4 Properties

25.1.153.4.1 ParamCheck Property

NexusDB V2 VCL Reference

TnxQuery.ParamCheck Property

[Contents](#) | [Index](#)

TnxQuery Class

Pascal

```
published property ParamCheck: boolean;
```

Remarks

see tQuery.ParamCheck

See Also

TnxQuery Class

You are here: Symbol Reference > Classes > TnxQuery Class > Properties > ParamCheck Property

25.1.153.4.2 SQL Property

NexusDB V2 VCL Reference

TnxQuery.SQL Property

[Contents](#) | [Index](#)

TnxQuery Class

Pascal

```
published property SQL: TStrings;
```

Remarks

see tQuery.SQL

See Also

TnxQuery Class

You are here: Symbol Reference > Classes > TnxQuery Class > Properties > SQL Property

25.1.153.4.3 Text Property

NexusDB V2 VCL Reference

TnxQuery.Text Property

[Contents](#) | [Index](#)

TnxQuery Class

Pascal

```
public property Text: string;
```

Remarks

see tQuery.Text

See Also

TnxQuery Class

You are here: Symbol Reference > Classes > TnxQuery Class > Properties > Text Property

25.1.15 TnxRefKeyDescriptor Class

NexusDB V2 VCL Reference

TnxRefKeyDescriptor Class

[Contents](#) | [Index](#)

[nxsdDataDictionary](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxRefKeyDescriptor = class(TnxBaseKeyDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

This Key Descriptor returns the internal id of the record as Key. It is used for the Sequential Access Index.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxRefKeyDescriptor Class

25.1.154. Methods

25.1.154.1.1 LoadFromReader Method

NexusDB V2 VCL Reference

TnxRefKeyDescriptor.LoadFromReader Method

[Contents](#) | [Index](#)

[TnxRefKeyDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxRefKeyDescriptor.

See Also

[TnxRefKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxRefKeyDescriptor Class > Methods > LoadFromReader Method

25.1.154.1.2 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxRefKeyDescriptor.SaveToWriter Method

[TnxRefKeyDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxRefKeyDescriptor.

See Also

[TnxRefKeyDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxRefKeyDescriptor Class > Methods > SaveToWriter Method

25.1.15:TnxRegisterableComponent Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxRegisterableComponent Class

[nxIIComponent](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxRegisterableComponent = class(TnxComponent);
```

File

[nxIIComponent](#)

Remarks

An component that can be registered to the system and thus created from class name.

See Also

[nxIIComponent](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxRegisterableComponent Class

25.1.155.1 Methods

25.1.155.1.1 FindRegisteredClass Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxRegisterableComponent.FindRegisteredClass Method

TnxRegisterableComponent Class

Pascal

```
public class function FindRegisteredClass(
    const aClassName: string
) : TnxRegisterableComponentClass;
```

Remarks

Returns it's class definition if the classname matches.

See Also

[TnxRegisterableComponent Class](#)

You are here: Symbol Reference > Classes > TnxRegisterableComponent Class > Methods > FindRegisteredClass Method

25.1.155.1.2 GetRegisteredClasses Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxRegisterableComponent.GetRegisteredClasses Method

TnxRegisterableComponent Class

Pascal

```
public class procedure GetRegisteredClasses(
    aList: TStrings
);
```

Remarks

Adds itself to the list of registered classes.

See Also

[TnxRegisterableComponent Class](#)

You are here: Symbol Reference > Classes > TnxRegisterableComponent Class > Methods > GetRegisteredClasses Method

25.1.156 TnxServer Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxServer Class

[nxServerComp](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxServer = class(TnxBaseServer);
```

File

[nxServerComp](#)

Remarks

This component implements a full server-in-a-component. Its not used anywhere in NexusDB binaries but is a very good example of how easy it is to create a complete server with the things provided.

See Also

[nxServerComp](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxServer Class

25.1.156. Properties

25.1.156.1.1 CommandHandler Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxServer.CommandHandler Property

TnxServer Class

Pascal

```
public property CommandHandler: TnxServerCommandHandler;
```

Remarks

Gives access to the internally created Command Handler.

See Also

[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > CommandHandler Property

25.1.156.1.2 EnabledModules Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxServer.EnabledModules Property

TnxServer Class

Pascal

```
published property EnabledModules: TnxServerModules;
```

Remarks

Selects the activated modules on startup.

See Also

[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > EnabledModules Property

25.1.156.1.3 EventLog Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxServer.EventLog Property

TnxServer Class

Pascal

```
public property EventLog: TnxEventLog;
```

Remarks

Gives access to the internally created Event Log.

See Also[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > EventLog Property

25.1.156.1.4 EventLogEnabled Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxServer.EventLogEnabled Property**[TnxServer Class](#)**Pascal**

```
published property EventLogEnabled: Boolean;
```

Remarks

If EventLogEnabled is true the event log is active.

See Also[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > EventLogEnabled Property

25.1.156.1.5 NamedPipeTransport Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxServer.NamedPipeTransport Property**[TnxServer Class](#)**Pascal**

```
public property NamedPipeTransport: TnxNamedPipeTransport;
```

Remarks

Gives access to the internally created Named Pipe Transport

See Also[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > NamedPipeTransport Property

25.1.156.1.6 SecurityMonitor Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxServer.SecurityMonitor Property**[TnxServer Class](#)**Pascal**

```
public property SecurityMonitor: TnxSecurityMonitor;
```

Remarks

Gives access to the internally created Security Monitor.

See Also[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > SecurityMonitor Property

25.1.156.1.7 ServerEngine Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxServer.ServerEngine Property

TnxServer Class

Pascal

```
public property ServerEngine: TnxBaseServerEngine;
```

Remarks

Gives access to the internally created Server.

See Also

[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > ServerEngine Property

25.1.156.1.8 ServerName Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxServer.ServerName Property

TnxServer Class

Pascal

```
published property ServerName: string;
```

Remarks

Set this property to set the Server Name instead of using the ServerEngine.ServerName property. It will also set the transports' ServerName properties correctly.

See Also

[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > ServerName Property

25.1.156.1.9 SQLEngine Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxServer.SQLEngine Property

TnxServer Class

Pascal

```
public property SQLEngine: TnxSqlEngine;
```

Remarks

Gives access to the internally created SQL Engine.

See Also

[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > SQLEngine Property

25.1.156.1.10 WinsockTransport Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxServer.WinsockTransport Property

TnxServer Class

Pascal

```
public property WinsockTransport: TnxWinsockTransport;
```

Remarks

Gives access to the internally created Winsock Transport.

See Also

[TnxServer Class](#)

You are here: Symbol Reference > Classes > TnxServer Class > Properties > WinsockTransport Property

25.1.157 TnxServerManager Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxServerManager Class

[nxServerManager](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxServerManager = class(TnxBaseServer);
```

File

[nxServerManager](#)

Remarks

This descendant of TnxBaseServer adds a published property to set its internal Server Engine.

See Also

[nxServerManager](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxServerManager Class

25.1.157.1 Properties

25.1.157.1.1 ServerEngine Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxServerManager.ServerEngine Property

TnxServerManager Class

Pascal

```
published property ServerEngine: TnxBaseServerEngine;
```

Remarks

The server engine used by the component's methods. Simply a server engine and you can use all the methods to manage (de)activating modules, loading/saving settings, ...

See Also

[TnxServerManager Class](#)

You are here: Symbol Reference > Classes > TnxServerManager Class > Properties > ServerEngine Property

25.1.15{TnxSession Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSession Class

[nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxSession = class(TnxBaseSession);
```

File

[nxdb](#)

Remarks

A BDE compatible implementation of a NexusDB session.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxSession Class

25.1.158.1:Destructors

25.1.158.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSession.Destroy Destructor

TnxSession Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxSession Class](#)

You are here: Symbol Reference > Classes > TnxSession Class > Destructors > Destroy Destructor

25.1.158.2:Properties

25.1.158.2.1 ServerEngine Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSession.ServerEngine Property

TnxSession Class

Pascal

```
public property ServerEngine: TnxBaseServerEngine;
```

Remarks

The connected Server Engine. Please note that every Session must(!) be connected to a Server engine before it can be opened

See Also

[TnxSession Class](#)

You are here: Symbol Reference > Classes > TnxSession Class > Properties > ServerEngine Property

25.1.15 TnxSessionOwnedDataAccessStateComponent Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSessionOwnedDataAccessStateComponent Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxSessionOwnedDataAccessStateComponent = class(TnxDataAccessStateComponent);
```

File

[nxdb](#)

Remarks

This is class TnxSessionOwnedDataAccessStateComponent.

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxSessionOwnedDataAccessStateComponent Class

25.1.159 Constructors

25.1.159.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSessionOwnedDataAccessStateComponent.Create Constructor

[TnxSessionOwnedDataAccessStateComponent Class](#)

Pascal

```
public constructor Create(
  aOwner: TComponent
); override;
```

Remarks

This is Create, a member of class TnxSessionOwnedDataAccessStateComponent.

See Also

[TnxSessionOwnedDataAccessStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxSessionOwnedDataAccessStateComponent Class > Constructors > Create Constructor

25.1.159.Methods

25.1.159.2.1 AfterConstruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSessionOwnedDataAccessStateComponent.AfterConstruction Method

TnxSessionOwnedDataAccessStateComponent Class

Pascal

```
public procedure AfterConstruction; override;
```

Remarks

This is AfterConstruction, a member of class TnxSessionOwnedDataAccessStateComponent.

See Also

[TnxSessionOwnedDataAccessStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxSessionOwnedDataAccessStateComponent Class > Methods > AfterConstruction Method

25.1.159.Properties

25.1.159.3.1 Session Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSessionOwnedDataAccessStateComponent.Session Property

TnxSessionOwnedDataAccessStateComponent Class

Pascal

```
published property Session: TnxBaseSession;
```

Remarks

The session the database is attached to. This property **must** be set for the component to work properly.

See Also

[TnxSessionOwnedDataAccessStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxSessionOwnedDataAccessStateComponent Class > Properties > Session Property

25.1.159.3.2 Timeout Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSessionOwnedDataAccessStateComponent.Timeout Property

TnxSessionOwnedDataAccessStateComponent Class

Pascal

```
published property Timeout: Integer;
```

Remarks

Set the timeout for the component in ms. If -1 is specified the component inherits the value of the session it is connected to.

See Also

[TnxSessionOwnedDataAccessStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxSessionOwnedDataAccessStateComponent Class > Properties > Timeout Property

25.1.16(TnxSessionPool Class)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSessionPool Class

[nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxSessionPool = class(TnxBaseSessionPool);
```

File

[nxdb](#)

Remarks

This is class TnxSessionPool.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxSessionPool Class

25.1.160.:Destructors

25.1.160.1.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSessionPool.Destroy Destructor

TnxSessionPool Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxSessionPool Class](#)

You are here: Symbol Reference > Classes > TnxSessionPool Class > Destructors > Destroy Destructor

25.1.160.:Properties

25.1.160.2.1 ServerEngine Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSessionPool.ServerEngine Property

TnxSessionPool Class

Pascal

```
public property ServerEngine: TnxBaseServerEngine;
```

Remarks

The connected Server Engine. Please note that every Session must(!) be connected to a Server engine before it can be opened

See Also

[TnxSessionPool Class](#)

You are here: Symbol Reference > Classes > TnxSessionPool Class > Properties > ServerEngine Property

25.1.16 TnxSetting Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSetting Class

[nxConfigSettings](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxSetting = class(TnxBaseSetting);
```

File

[nxConfigSettings](#)

Remarks

A setting that can get it's attributes from another setting.

See Also

[nxConfigSettings](#)

[Members](#)

[Methods](#)

You are here: Symbol Reference > Classes > TnxSetting Class

25.1.161 Methods

25.1.161.1 Assign Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSetting.Assign Method

[TnxSetting Class](#)

Pascal

```
public procedure Assign(
    aSource: TnxBaseSetting
); virtual;
```

Remarks

set the properties according to aSource

See Also

[TnxSetting Class](#)

You are here: Symbol Reference > Classes > TnxSetting Class > Methods > Assign Method

25.1.16 TnxSettings Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSettings Class

[nxConfigSettings](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxSettings = class(TnxBaseSettings);
```

File

[nxConfigSettings](#)

Remarks

Implementation of the TnxBaseSettings prototype.

See Also

[nxConfigSettings](#)

[Members](#)

[Methods](#)

You are here: Symbol Reference > Classes > TnxSettings Class

25.1.162. Constructors

25.1.162.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSettings.Create Constructor

TnxSettings Class

Pascal

```
public constructor Create;
```

Remarks

constructor

See Also

[TnxSettings Class](#)

You are here: Symbol Reference > Classes > TnxSettings Class > Constructors > Create Constructor

25.1.162. Destructors

25.1.162.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSettings.Destroy Destructor

TnxSettings Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor

See Also

[TnxSettings Class](#)

You are here: Symbol Reference > Classes > TnxSettings Class > Destructors > Destroy Destructor

25.1.162. Methods

25.1.162.3.1 AddSetting Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSettings.AddSetting Method

TnxSettings Class

Pascal

```
public function AddSetting(
    aSetting: TnxBaseSetting
): TnxBaseSetting; override;
```

Remarks

This is AddSetting, a member of class TnxSettings.

See Also

[TnxSettings Class](#)

You are here: Symbol Reference > Classes > TnxSettings Class > Methods > AddSetting Method

25.1.163. TnxSimpleLock Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSimpleLock Class

[nxsdServerEngine](#) | [Members](#) | [Methods](#)

Pascal

```
public TnxSimpleLock = class(TnxObject);
```

File

[nxsdServerEngine](#)

Remarks

A 100% thread safe lock used for counter base NexusDB locks.

See Also

[nxsdServerEngine](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxSimpleLock Class

25.1.163. Constructors

25.1.163.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSimpleLock.Create Constructor

TnxSimpleLock Class

Pascal

```
public constructor Create(
    aFailedCleanup: TNotifyEvent
);
```

Remarks

This is Create, a member of class TnxSimpleLock.

■ **See Also**

[TnxSimpleLock Class](#)

You are here: Symbol Reference > Classes > TnxSimpleLock Class > Constructors > Create Constructor

25.1.163.:Destructors

25.1.163.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxSimpleLock.Destroy Destructor

[TnxSimpleLock Class](#)

Pascal

```
public destructor Destroy; override;
```

■ **Remarks**

destructor.

■ **See Also**

[TnxSimpleLock Class](#)

You are here: Symbol Reference > Classes > TnxSimpleLock Class > Destructors > Destroy Destructor

25.1.163.:Methods

25.1.163.3.1 Failed Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxSimpleLock.Failed Method

[TnxSimpleLock Class](#)

Pascal

```
public procedure Failed(
    aPermanent: Boolean
);
```

■ **Remarks**

Internally used to coordinate long running operations and lost connections at the transport level... stopping e.g. running SQL statements after a few sec max if the connection is lost.

■ **See Also**

[TnxSimpleLock Class](#)

You are here: Symbol Reference > Classes > TnxSimpleLock Class > Methods > Failed Method

25.1.163.3.2 Lock Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxSimpleLock.Lock Method

[TnxSimpleLock Class](#)

Pascal

```
public procedure Lock;
```

Remarks

Increases the lock count.

See Also

[TnxSimpleLock Class](#)

You are here: Symbol Reference > Classes > TnxSimpleLock Class > Methods > Lock Method

25.1.163.3.3 Unlock Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSimpleLock.Unlock Method

[TnxSimpleLock Class](#)

Pascal

```
public procedure Unlock;
```

Remarks

Decreases the lock count.

See Also

[TnxSimpleLock Class](#)

You are here: Symbol Reference > Classes > TnxSimpleLock Class > Methods > Unlock Method

25.1.164 TnxSqlCheckValidationDescriptor Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlCheckValidationDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxSqlCheckValidationDescriptor = class(TnxBaseFieldsValidationDescriptor);
```

File

[nxsdDataDictionary](#)

Remarks

A validation descriptor to disallow field changes once it was initially set.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxSqlCheckValidationDescriptor Class

25.1.164.1 Methods

25.1.164.1.1 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlCheckValidationDescriptor.IsEqual Method

[TnxSqlCheckValidationDescriptor Class](#)

```
Pascal
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxSqlCheckValidationDescriptor.

See Also

[TnxSqlCheckValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxSqlCheckValidationDescriptor Class > Methods > IsEqual Method

25.1.164.1.2 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlCheckValidationDescriptor.LoadFromReader Method

[TnxSqlCheckValidationDescriptor Class](#)

```
Pascal
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxSqlCheckValidationDescriptor.

See Also

[TnxSqlCheckValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxSqlCheckValidationDescriptor Class > Methods > LoadFromReader Method

25.1.164.1.3 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlCheckValidationDescriptor.SaveToWriter Method

[TnxSqlCheckValidationDescriptor Class](#)

```
Pascal
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxSqlCheckValidationDescriptor.

See Also

[TnxSqlCheckValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxSqlCheckValidationDescriptor Class > Methods > SaveToWriter Method

25.1.164.1 Properties

25.1.164.2.1 Constraint Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSqlCheckValidationDescriptor.Constraint Property

TnxSqlCheckValidationDescriptor Class

Pascal

```
public property Constraint: WideString;
```

Remarks

This is Constraint, a member of class TnxSqlCheckValidationDescriptor.

See Also

[TnxSqlCheckValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxSqlCheckValidationDescriptor Class > Properties > Constraint Property

25.1.164.2.2 ErrorMessage Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSqlCheckValidationDescriptor.ErrorMessage Property

TnxSqlCheckValidationDescriptor Class

Pascal

```
public property ErrorMessage: string;
```

Remarks

This is ErrorMessage, a member of class TnxSqlCheckValidationDescriptor.

See Also

[TnxSqlCheckValidationDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxSqlCheckValidationDescriptor Class > Properties > ErrorMessage Property

25.1.165 TnxSqlUpdateObject Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSqlUpdateObject Class

[nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxSqlUpdateObject = class(TnxBaseUpdateObject);
```

File

[nxdb](#)

Remarks

Update Object for updating a cached datasets source datasets using SQL Statements.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class

25.1.165. Constructors

25.1.165.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSqlUpdateObject.Create Constructor

TnxSqlUpdateObject Class

Pascal

```
public constructor Create(
    AOwner: TComponent
); override;
```

Remarks

constructor

See Also

[TnxSqlUpdateObject Class](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class > Constructors > Create Constructor

25.1.165.2. Destructors

25.1.165.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSqlUpdateObject.Destroy Destructor

TnxSqlUpdateObject Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor

See Also

[TnxSqlUpdateObject Class](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class > Destructors > Destroy Destructor

25.1.165.3. Properties

25.1.165.3.1 DeleteSql Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxSqlUpdateObject.DeleteSql Property

TnxSqlUpdateObject Class

Pascal

```
published property DeleteSql: TStrings;
```

Remarks

SQL script used to delete records from underlying datasets use \$fieldname for specifying fieldvalues as parameters and param\$paramname for original parameters

See Also

[TnxSqlUpdateObject Class](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class > Properties > DeleteSql Property

25.1.165.3.2 InsertSql Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlUpdateObject.InsertSql Property

[TnxSqlUpdateObject Class](#)

Pascal

```
published property InsertSql: TStrings;
```

Remarks

SQL script used to insert new records to underlying datasets use \$fieldname for specifying fieldvalues as parameters and param\$paramname for original parameters

See Also

[TnxSqlUpdateObject Class](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class > Properties > InsertSql Property

25.1.165.3.3 ModifySql Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlUpdateObject.ModifySql Property

[TnxSqlUpdateObject Class](#)

Pascal

```
published property ModifySql: TStrings;
```

Remarks

SQL script used to update trecords of underlying datasets use :old\$fieldname, :new\$fieldname for specifying fieldvalues as parameters and param\$paramname for original parameters

See Also

[TnxSqlUpdateObject Class](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class > Properties > ModifySql Property

25.1.165.3.4 Params Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlUpdateObject.Params Property

[TnxSqlUpdateObject Class](#)

Pascal

```
published property Params: TParams;
```

Remarks

see tQuery.Params

See Also

[TnxSqlUpdateObject Class](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class > Properties > Params Property

25.1.165.3.5 Sql Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSqlUpdateObject.Sql Property

[TnxSqlUpdateObject Class](#) | [Example](#)

Pascal

```
public property Sql [aUpdateKind: TUpdateKind]: TStrings;
```

Remarks

Gets/Sets the SQL statement for the given aUpdateKind.

Example

```
query (in TnxQuery connected via SourceDataSet): SELECT OrderID, CustomerID FROM Orders  
WHERE ShipVia = 2 insert: INSERT INTO Orders (OrderID, CustomerID, ShipVia) VALUES  
(:OrderID, :CustomerID, :param$ShipVia) modify: UPDATE Orders SET OrderID = :OrderID;  
Customer = :CustomerID WHERE OrderID = :old$OrderID delete: DELETE FROM Orders WHERE  
OrderID = :OrderID
```

and add to the Params collection a param called "ShipVia", type ftInteger, value 2

See Also

[TnxSqlUpdateObject Class](#)
[Example](#)

You are here: Symbol Reference > Classes > TnxSqlUpdateObject Class > Properties > Sql Property

25.1.16 TnxStateComponent Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStateComponent Class

[nxIIComponent](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxStateComponent = class(TnxLoggableComponent);
```

File

[nxIIComponent](#)

Remarks

A state component has an integrated simple state engine for Initialazing, Starting, Stopping and Dactivating the component.

See Also

[nxIIComponent](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class

25.1.166. Constructors

25.1.166.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.Create Constructor

TnxStateComponent Class

Pascal

```
public constructor Create(
    aOwner: TComponent
) ; override;
```

Remarks

This is Create, a member of class TnxStateComponent.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Constructors > Create Constructor

25.1.166. Destructors

25.1.166.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.Destroy Destructor

TnxStateComponent Class

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxStateComponent.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Destructors > Destroy Destructor

25.1.166. Methods

25.1.166.3.1 BeforeDestruction Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.BeforeDestruction Method

TnxStateComponent Class

Pascal

```
public procedure BeforeDestruction; override;
```

Remarks

This is BeforeDestruction, a member of class TnxStateComponent.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > BeforeDestruction Method

25.1.166.3.2 Close Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.Close Method

TnxStateComponent Class

Pascal

```
public procedure Close;
```

Remarks

Calling the close method is equivalent to setting Active to false.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > Close Method

25.1.166.3.3 Connect Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.Connect Method

TnxStateComponent Class

Pascal

```
public procedure Connect;
```

Remarks

Calling the connect method is equivalent to setting Active to true.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > Connect Method

25.1.166.3.4 GetConfigSettings Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.GetConfigSettings Method

TnxStateComponent Class

Pascal

```
public procedure GetConfigSettings(
    aSettings: TnxBaseSettings
); override;
```

Remarks

This is GetConfigSettings, a member of class TnxStateComponent.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > GetConfigSettings Method

25.1.166.3.5 GetStatsCaptions Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxStateComponent.GetStatsCaptions Method**[TnxStateComponent Class](#)**Pascal**

```
public procedure GetStatsCaptions(
  const aList: TStrings
); override;
```

Remarks

This is GetStatsCaptions, a member of class TnxStateComponent.

See Also[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > [GetStatsCaptions Method](#)

25.1.166.3.6 GetStatsValues Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxStateComponent.GetStatsValues Method**[TnxStateComponent Class](#)**Pascal**

```
public procedure GetStatsValues(
  const aList: TStrings
); override;
```

Remarks

This is GetStatsValues, a member of class TnxStateComponent.

See Also[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > [GetStatsValues Method](#)

25.1.166.3.7 LoadConfig Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxStateComponent.LoadConfig Method**[TnxStateComponent Class](#)**Pascal**

```
public procedure LoadConfig(
  aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

This is LoadConfig, a member of class TnxStateComponent.

See Also[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > LoadConfig Method

25.1.166.3.8 LoadSettingsFromStream Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.LoadSettingsFromStream Method

TnxStateComponent Class

Pascal

```
public procedure LoadSettingsFromStream(
    aReader: TnxReader
); override;
```

Remarks

This is LoadSettingsFromStream, a member of class TnxStateComponent.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > LoadSettingsFromStream Method

25.1.166.3.9 Open Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.Open Method

TnxStateComponent Class

Pascal

```
public procedure Open;
```

Remarks

Calling the open method is equivalent to setting Active to true.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > Open Method

25.1.166.3.10 SaveConfig Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.SaveConfig Method

TnxStateComponent Class

Pascal

```
public procedure SaveConfig(
    aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

This is SaveConfig, a member of class TnxStateComponent.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > SaveConfig Method

25.1.166.3.11 SaveSettingsToStream Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.SaveSettingsToStream Method

TnxStateComponent Class

Pascal

```
public procedure SaveSettingsToStream(
  aWriter: TnxWriter
); override;
```

Remarks

This is SaveSettingsToStream, a member of class TnxStateComponent.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > SaveSettingsToStream Method

25.1.166.3.12 SwitchToSavedActive Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.SwitchToSavedActive Method

TnxStateComponent Class

Pascal

```
public procedure SwitchToSavedActive; virtual;
```

Remarks

A special command that can set the Active state to a before saved one. Also see: SavedActive

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > SwitchToSavedActive Method

25.1.166.3.13 UIStateVisible Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.UIStateVisible Method

TnxStateComponent Class

Pascal

```
public function UIStateVisible: Boolean; virtual;
```

Remarks

Set this to true if the state of this component should be visible in UI (legacy V1)

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Methods > UIStateVisible Method

25.1.166.4.Properties

25.1.166.4.1 Active Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.Active Property

TnxStateComponent Class

Pascal

```
public property Active: Boolean;
```

Remarks

(De)Activate the component. This is done by switching it through the state engine until the Active state is reached or an error occurs.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > Active Property

25.1.166.4.2 ActiveDesigntime Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.ActiveDesigntime Property

TnxStateComponent Class

Pascal

```
published property ActiveDesigntime: Boolean;
```

Remarks

Set this to true if you want the component to start automatically in design mode.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > ActiveDesigntime Property

25.1.166.4.3 ActiveRuntime Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStateComponent.ActiveRuntime Property

TnxStateComponent Class

Pascal

```
published property ActiveRuntime: Boolean;
```

Remarks

Set this to true if you want the component to start automatically on application startup.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > ActiveRuntime Property

25.1.166.4.4 Connected Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxStateComponent.Connected Property

TnxStateComponent Class

Pascal

```
public property Connected: Boolean;
```

Remarks

(De)Activate the component. This is done by switching it through the state engine until the Active state is reached or an error occurs.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > Connected Property

25.1.166.4.5 Enabled Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxStateComponent.Enabled Property

TnxStateComponent Class

Pascal

```
published property Enabled: Boolean;
```

Remarks

Enable/Disable this component.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > Enabled Property

25.1.166.4.6 OnStateChanged Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxStateComponent.OnStateChanged Property

TnxStateComponent Class

Pascal

```
published property OnStateChanged: TNotifyEvent;
```

Remarks

This event is triggered every time the state of the component changed.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > OnStateChanged Property

25.1.166.4.7 OnStateTransChanged Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxStateComponent.OnStateTransChanged Property

TnxStateComponent Class

Pascal

```
published property OnStateTransChanged: TNotifyEvent;
```

Remarks

This event is triggered every time the state transition of the component changed.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > OnStateTransChanged Property

25.1.166.4.8 SavedActive Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxStateComponent.SavedActive Property

TnxStateComponent Class

Pascal

```
public property SavedActive: byte;
```

Remarks

Can be used to save a certain state.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > SavedActive Property

25.1.166.4.9 StartTime Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxStateComponent.StartTime Property

TnxStateComponent Class

Pascal

```
public property StartTime: TDateTime;
```

Remarks

Time the component was last set to active.

See Also

[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > StartTime Property

25.1.166.4.10 State Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxStateComponent.State Property**[TnxStateComponent Class](#)**Pascal**

```
public property State: TnxState;
```

Remarks

Get/Sets the state of the component. When setting the stae that must be in conformance to the state engine.

See Also[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > State Property

25.1.166.4.11 StateTransition Property

NexusDB V2 VCL Reference[Contents | Index](#)**TnxStateComponent.StateTransition Property**[TnxStateComponent Class](#)**Pascal**

```
public property StateTransition: TnxStateTransition;
```

Remarks

Returns the current StateTransition the component is in. If it is in a proper state this will return none.

See Also[TnxStateComponent Class](#)

You are here: Symbol Reference > Classes > TnxStateComponent Class > Properties > StateTransition Property

25.1.16 TnxStatementDataSet Class*NexusDB V2 VCL Reference*[Contents | Index](#)**TnxStatementDataSet Class**[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxStatementDataSet = class(TnxDataset);
```

File[nxdb](#)**Remarks**

TQuery compatible implementation of a Nexus Sql Query

See Also
[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class

25.1.167. Constructors

25.1.167.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStatementDataSet.Create Constructor

TnxStatementDataSet Class

Pascal

```
public constructor Create(
    aOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Constructors > Create Constructor

25.1.167. Destructors

25.1.167.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStatementDataSet.Destroy Destructor

TnxStatementDataSet Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Destructors > Destroy Destructor

25.1.167. Methods

25.1.167.3.1 ParamByName Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStatementDataSet.ParamByName Method

TnxStatementDataSet Class

Pascal

```
public function ParamByName(
    const aName: string
): TParam;
```

Remarks

see tQuery.ParamByName

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Methods > ParamByName Method

25.1.167.3.2 Prepare Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.Prepare Method

[TnxStatementDataSet Class](#)

Pascal

```
public procedure Prepare; virtual;
```

Remarks

see tQuery.Prepare

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Methods > Prepare Method

25.1.167.3.3 Unprepare Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.Unprepare Method

[TnxStatementDataSet Class](#)

Pascal

```
public procedure Unprepare;
```

Remarks

see tQuery.UnPrepare

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Methods > Unprepare Method

25.1.167.Properties

25.1.167.4.1 DataSource Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.DataSource Property

[TnxStatementDataSet Class](#)

Pascal

```
published property DataSource: TDataSource;
```

Remarks

see tQuery.DataSource.

□ **See Also**

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > DataSource Property

25.1.167.4.2 Log Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.Log Property

[TnxStatementDataSet Class](#)

Pascal

```
public property Log: TStringList;
```

□ **Remarks**

Log returns a detailed analysis of the Query. You need to at least Prepare the Query for Log to be updated.

□ **See Also**

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > Log Property

25.1.167.4.3 ParamCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.ParamCount Property

[TnxStatementDataSet Class](#)

Pascal

```
public property ParamCount: Word;
```

□ **Remarks**

see tQuery.ParamCount

□ **See Also**

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > ParamCount Property

25.1.167.4.4 Params Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.Params Property

[TnxStatementDataSet Class](#)

Pascal

```
published property Params: TParams;
```

□ **Remarks**

see tQuery.Params

□ **See Also**

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > Params Property

25.1.167.4.5 Prepared Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.Prepared Property

TnxStatementDataSet Class

Pascal

```
public property Prepared: Boolean;
```

Remarks

see tQuery.Prepared

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > Prepared Property

25.1.167.4.6 ReadOnly Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.ReadOnly Property

TnxStatementDataSet Class

Pascal

```
published property ReadOnly: Boolean;
```

Remarks

Prevents modification of a live result set.

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > ReadOnly Property

25.1.167.4.7 RecordsRead Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementDataSet.RecordsRead Property

TnxStatementDataSet Class

Pascal

```
public property RecordsRead: Integer;
```

Remarks

RecordsRead is set after calling ExecSql and the value indicates how many records were read.

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > RecordsRead Property

25.1.167.4.8 RequestLive Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStatementDataSet.RequestLive Property

[TnxStatementDataSet Class](#)

Pascal

```
published property RequestLive: boolean;
```

Remarks

Only supported for "SELECT * FROM tablename" type queries currently.

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > RequestLive Property

25.1.167.4.9 RowsAffected Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStatementDataSet.RowsAffected Property

[TnxStatementDataSet Class](#)

Pascal

```
public property RowsAffected: Integer;
```

Remarks

see tQuery.RowsAffected

See Also

[TnxStatementDataSet Class](#)

You are here: Symbol Reference > Classes > TnxStatementDataSet Class > Properties > RowsAffected Property

25.1.16 TnxStoredProcedure Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStoredProcedure Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxStoredProcedure = class(TnxStatementDataSet);
```

File

nxdb

Remarks

This class is the client side representation of a stored procedure.

See Also

[nxdb](#)

[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxStoredProc Class

25.1.168. Methods

25.1.168.1.1 ExecProc Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStoredProc.ExecProc Method

TnxStoredProc Class

Pascal

```
public procedure ExecProc;
```

Remarks

see tQuery.ExecSql

See Also

[TnxStoredProc Class](#)

You are here: Symbol Reference > Classes > TnxStoredProc Class > Methods > ExecProc Method

25.1.168.1.2 Prepare Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStoredProc.Prepare Method

TnxStoredProc Class

Pascal

```
public procedure Prepare; override;
```

Remarks

see tQuery.Prepare

See Also

[TnxStoredProc Class](#)

You are here: Symbol Reference > Classes > TnxStoredProc Class > Methods > Prepare Method

25.1.168.1.3 RefreshParam Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStoredProc.RefreshParam Method

TnxStoredProc Class

Pascal

```
public procedure RefreshParam; virtual;
```

Remarks

Refreshed all Parameters.

See Also

[TnxStoredProc Class](#)

You are here: Symbol Reference > Classes > TnxStoredProc Class > Methods > RefreshParam Method

25.1.168.Properties

25.1.168.2.1 StoredProcName Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStoredProc.StoredProcName Property

TnxStoredProc Class

Pascal

```
published property StoredProcName: string;
```

Remarks

The SQL name of the procedure.

See Also

[TnxStoredProc Class](#)

You are here: Symbol Reference > Classes > TnxStoredProc Class > Properties > StoredProcName Property

25.1.169.TnxTable Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable Class

[Events](#) | [nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxTable = class(TnxIndexDataSet);
```

File

[nxdb](#)

Remarks

TnxTable is a TTable compatible implementation of a NexusDB dataset.

See Also

[Events](#)

[nxdb](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Classes > TnxTable Class

25.1.169.Constructors

25.1.169.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.Create Constructor

TnxTable Class

Pascal

```
public constructor Create(
```

```
AOwner: TComponent
); override;
```

Remarks

constructor.

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Constructors > Create Constructor

25.1.169.1.Destructors

25.1.169.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.Destroy Destructor

[TnxTable Class](#)

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Destructors > Destroy Destructor

25.1.169.2.Methods

25.1.169.3.1 AddIndex Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.AddIndex Method

[TnxTable Class](#) | [Related Topics](#)

Pascal

```
public procedure AddIndex(
  const aName: string;
  const aFields: string;
  aOptions: TIndexOptions;
  const aCaseInsFields: string = '';
  const aDescFields: string = '';
  aNewPrimary: Boolean = False
);
```

Remarks

see [tTable.AddIndex](#)

Related Topics

[AddIndex](#), [AddIndexEx](#), [DeleteIndex](#)

See Also

[TnxTable Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > AddIndex Method

25.1.169.3.2 AddIndexEx Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.AddIndexEx Method

[TnxTable Class](#) | [Related Topics](#)

Pascal

```
public function AddIndexEx(
    aIndexDesc: TnxIndexDescriptor;
    aNewPrimary: Boolean;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult;
```

Parameters

Parameters	Description
aIndexDesc	a full index descriptor for the new index
aTaskInfo	Task information instance; use TaskInfo.GetStatus to find out the status of the operation.

Remarks

With AddIndexEx you can use a complete index descriptor instead of just field names to define the index. This gives you much more control over the type of index created. The index creation will be executed asynchronously.

Related Topics

[AddIndex](#), [AddIndexEx](#), [DeleteIndex](#), [TnxAbstractTaskInfo](#), [GetStatus](#)

See Also

[TnxTable Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > AddIndexEx Method

25.1.169.3.3 ChangePassword Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.ChangePassword Method

[TnxTable Class](#)

Pascal

```
public procedure ChangePassword(
    const aNewPassword: string
);
```

Remarks

Changes the password of the table.

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > ChangePassword Method

25.1.169.3.4 CreateTable Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxTable.CreateTable Method**[TnxTable Class](#)**Pascal**

```
public procedure CreateTable(
    aTableScope: TnxTableScope = tsPersistent;
    aNoIndices: Boolean = False
);
```

Remarks

see tTable.CreateTable

See Also[TnxTable Class](#)*You are here: Symbol Reference > Classes > TnxTable Class > Methods > CreateTable Method*

25.1.169.3.5 CreateTableEx Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxTable.CreateTableEx Method**[TnxTable Class](#)**Pascal**

```
public procedure CreateTableEx(
    aBlockSize: TnxBlockSize;
    aTableScope: TnxTableScope = tsPersistent;
    aNoIndices: Boolean = False
);
```

Parameters

Parameters	Description
aBlockSize	block size for the newly created table

Remarks

CreateTableEx is the same as CreateTable, but with the possibility to set the BlockSize for the new table.

See Also[TnxTable Class](#)*You are here: Symbol Reference > Classes > TnxTable Class > Methods > CreateTableEx Method*

25.1.169.3.6 DeleteIndex Method

NexusDB V2 VCL Reference[Contents | Index](#)**TnxTable.DeleteIndex Method**[TnxTable Class](#) | [Related Topics](#)**Pascal**

```
public procedure DeleteIndex(
    const aIndexName: string
);
```

Remarks

see tTable.DeleteIndex

Related Topics

[AddIndex](#), [AddIndexEx](#), [DeleteIndex](#)

See Also

[TnxTable Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > DeleteIndex Method

25.1.169.3.7 DeleteTable Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxTable.DeleteTable Method

TnxTable Class

Pascal

```
public procedure DeleteTable;
```

Remarks

see tTable.DeleteTable

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > DeleteTable Method

25.1.169.3.8 EmptyTable Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxTable.EmptyTable Method

TnxTable Class

Pascal

```
public procedure EmptyTable;
```

Remarks

see tTable.EmptyTable

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > EmptyTable Method

25.1.169.3.9 Exists Method

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxTable.Exists Method

TnxTable Class

Pascal

```
public function Exists: Boolean; override;
```

Remarks

same as TnxDataset.Exists

See Also

TnxTable Class

You are here: Symbol Reference > Classes > TnxTable Class > Methods > Exists Method

25.1.169.3.10 PackTable Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.PackTable Method

[TnxTable Class](#) | [Related Topics](#)

Pascal

```
public function PackTable: TnxAbstractTaskInfo;
```

Remarks

See tnxDatabase.PackTable

Related Topics

TnxBaseSession.PackTable

See Also

[TnxTable Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > PackTable Method

25.1.169.3.11 PackTableEx Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.PackTableEx Method

[TnxTable Class](#) | [Related Topics](#)

Pascal

```
public function PackTableEx(
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult;
```

Remarks

See tnxDatabase.PackTable

Related Topics

TnxBaseSession.PackTable

See Also

[TnxTable Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > PackTableEx Method

25.1.169.3.12 ReIndexTable

25.1.169.3.12.1 ReIndexTable Method (Integer)

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.ReIndexTable Method (Integer)

[TnxTable Class](#) | [Related Topics](#)

```
Pascal
public function ReIndexTable(
    aIndexNum: Integer
): TnxAbstractTaskInfo; overload;
```

Remarks

See `TnxDatabase.ReIndexTable`

Related Topics

[TnxDatabase.ReIndexTable](#)

See Also

[TnxTable Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > ReIndexTable > ReIndexTable Method (Integer)

25.1.169.3.12.2 ReIndexTable Method (string)

[NexusDB V2 VCL Reference](#)

[Contents](#) | [Index](#)

TnxTable.ReIndexTable Method (string)

[TnxTable Class](#) | [Related Topics](#)

```
Pascal
public function ReIndexTable(
    const aIndexName: string
): TnxAbstractTaskInfo; overload;
```

Remarks

See `TnxDatabase.ReIndexTable`

Related Topics

[TnxDatabase.ReIndexTable](#)

See Also

[TnxTable Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > ReIndexTable > ReIndexTable Method (string)

25.1.169.3.13 ReIndexTableEx

25.1.169.3.13.1 ReIndexTableEx Method (Integer, TnxAbstractTaskInfo)

[NexusDB V2 VCL Reference](#)

[Contents](#) | [Index](#)

TnxTable.ReIndexTableEx Method (Integer, TnxAbstractTaskInfo)

[TnxTable Class](#) | [Related Topics](#)

```
Pascal
public function ReIndexTableEx(
    aIndexNum: Integer;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; overload;
```

Remarks

See `tnxDatabase.ReIndexTable`

Related Topics

[TnxDatabase.ReIndexTable](#)

See Also

[TnxTable Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > ReIndexTableEx > ReIndexTableEx Method (Integer, TnxAbstractTaskInfo)

25.1.169.3.13.2 ReIndexTableEx Method (string, TnxAbstractTaskInfo)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.ReIndexTableEx Method (string, TnxAbstractTaskInfo)

[TnxTable Class](#) | [Related Topics](#)

Pascal

```
public function ReIndexTableEx(
  const aIndexName: string;
  out aTaskInfo: TnxAbstractTaskInfo
): TnxResult; overload;
```

Remarks

See `tnxDatabase.ReIndexTable`

Related Topics

[TnxDatabase.ReIndexTable](#)

See Also

[TnxTable Class](#)

[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > ReIndexTableEx > ReIndexTableEx Method (string, TnxAbstractTaskInfo)

25.1.169.3.14 RenameTable Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.RenameTable Method

[TnxTable Class](#)

Pascal

```
public procedure RenameTable(
  const aNewTableName: string
);
```

Remarks

see `tTable.RenameTable`

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Methods > RenameTable Method

25.1.169.3.15 RestructureTable Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxTable.RestructureTable Method**[TnxTable Class](#) | [Related Topics](#)**Pascal**

```
public function RestructureTable(
    aDictionary: TnxDataDictionary;
    aMapperDesc: TnxBaseTableMapperDescriptor
): TnxAbstractTaskInfo;
```

Remarks

See [tnxDatabase.RestructureTable](#)

Related Topics

[TnxDatabase.RestructureTable](#)

See Also

[TnxTable Class](#)

[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxTable Class](#) > [Methods](#) > [RestructureTable Method](#)

25.1.169.3.16 RestructureTableEx Method

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxTable.RestructureTableEx Method**[TnxTable Class](#) | [Related Topics](#)**Pascal**

```
public function RestructureTableEx(
    aDictionary: TnxDataDictionary;
    aMapperDesc: TnxBaseTableMapperDescriptor;
    out aTaskInfo: TnxAbstractTaskInfo
): TnxResult;
```

Remarks

See [tnxDatabase.RestructureTable](#)

Related Topics

[TnxDatabase.RestructureTable](#)

See Also

[TnxTable Class](#)

[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxTable Class](#) > [Methods](#) > [RestructureTableEx Method](#)

25.1.169.4 Events

25.1.169.4.1 OnCreateTable Property

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**TnxTable.OnCreateTable Property**[TnxTable Class](#)

Pascal

```
published property OnCreateTable: TnxCreateTableEvent;
```

Remarks

This event is fired before the table is created by the server.

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Events > OnCreateTable Property

25.1.169.!Properties

25.1.169.5.1 Exclusive Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxTable.Exclusive Property

[TnxTable Class](#)

Pascal

```
published property Exclusive: Boolean;
```

Remarks

see tTable.Exclusive

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > Exclusive Property

25.1.169.5.2 FieldDefs Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxTable.FieldDefs Property

[TnxTable Class](#)

Pascal

```
published property FieldDefs;
```

Remarks

see tTable.FieldDefs

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > FieldDefs Property

25.1.169.5.3 IndexDefs Property

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxTable.IndexDefs Property

[TnxTable Class](#)

Pascal

```
published property IndexDefs;
```

Remarks

see tTable.IndexDefs.

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > IndexDefs Property

25.1.169.5.4 IndexFieldCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.IndexFieldCount Property

[TnxTable Class](#)

Pascal

```
public property IndexFieldCount;
```

Remarks

see tTable.IndexFieldCount.

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > IndexFieldCount Property

25.1.169.5.5 IndexFieldNames Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.IndexFieldNames Property

[TnxTable Class](#)

Pascal

```
published property IndexFieldNames;
```

Remarks

see tTable.IndexFieldNames

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > IndexFieldNames Property

25.1.169.5.6 IndexFields Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.IndexFields Property

[TnxTable Class](#)

Pascal

```
public property IndexFields;
```

Remarks

see tTable.IndexFields

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > IndexFields Property

25.1.169.5.7 IndexName Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.IndexName Property

TnxTable Class

Pascal

```
published property IndexName;
```

Remarks

see tTable.IndexName

See Also

TnxTable Class

You are here: Symbol Reference > Classes > TnxTable Class > Properties > IndexName Property

25.1.169.5.8 KeyExclusive Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.KeyExclusive Property

TnxTable Class

Pascal

```
public property KeyExclusive;
```

Remarks

see tTable.KeyExclusive

See Also

TnxTable Class

You are here: Symbol Reference > Classes > TnxTable Class > Properties > KeyExclusive Property

25.1.169.5.9 KeyFieldCount Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.KeyFieldCount Property

TnxTable Class

Pascal

```
public property KeyFieldCount;
```

Remarks

see tTable.KeyFieldCount

See Also

TnxTable Class

You are here: Symbol Reference > Classes > TnxTable Class > Properties > KeyFieldCount Property

25.1.169.5.10 KeyPartialLen Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.KeyPartialLen Property

TnxTable Class

Pascal

```
public property KeyPartialLen;
```

Remarks

see tTable.KeyPartialLen

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > KeyPartialLen Property

25.1.169.5.11 MasterFields Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.MasterFields Property

TnxTable Class

Pascal

```
published property MasterFields;
```

Remarks

see tTable.MasterFields

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > MasterFields Property

25.1.169.5.12 MasterSource Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.MasterSource Property

TnxTable Class

Pascal

```
published property MasterSource;
```

Remarks

see tTable.MasterSource

See Also

[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > MasterSource Property

25.1.169.5.13 Password Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTable.Password Property

TnxTable Class

Pascal

```
published property Password: string;
```

Remarks

The password of the current table.

See Also

[TnxTable Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxTable Class](#) > [Properties](#) > [Password Property](#)

25.1.169.5.14 ReadOnly Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.ReadOnly Property

[TnxTable Class](#)

Pascal

```
published property ReadOnly: Boolean;
```

Remarks

see [tTable.ReadOnly](#)

See Also

[TnxTable Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxTable Class](#) > [Properties](#) > [ReadOnly Property](#)

25.1.169.5.15 StoreDefs Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.StoreDefs Property

[TnxTable Class](#)

Pascal

```
published property StoreDefs: Boolean;
```

Remarks

see [tTable.StoreDefs](#)

See Also

[TnxTable Class](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxTable Class](#) > [Properties](#) > [StoreDefs Property](#)

25.1.169.5.16 TableName Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTable.TableName Property

[TnxTable Class](#)

Pascal

```
published property TableName: string;
```

Remarks

see [tTable.TableName](#)

See Also[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > TableName Property

25.1.169.5.17 WriteOnly Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxTable.WriteOnly Property**[TnxTable Class](#)**Pascal**

```
published property WriteOnly: Boolean;
```

Remarks

This property can only be changed if the table is inactive. It prevents the table from ever trying to read records from the server. This improves performance for tables that are only used to insert records.

See Also[TnxTable Class](#)

You are here: Symbol Reference > Classes > TnxTable Class > Properties > WriteOnly Property

25.1.17(TnxTableDescriptor Class)

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxTableDescriptor Class**[nxsdServerEngine](#)**Pascal**

```
public TnxTableDescriptor = class(TnxObject);
```

File[nxsdServerEngine](#)**Remarks**

A Table descriptor.

See Also[nxsdServerEngine](#)

You are here: Symbol Reference > Classes > TnxTableDescriptor Class

25.1.17TnxTablesDescriptor Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxTablesDescriptor Class**[nxsdDataDictionary](#) | [Members](#) | [Methods](#)**Pascal**

```
public TnxTablesDescriptor = class(TnxDictionaryItem);
```

File[nxsdDataDictionary](#)

Remarks

Holds a list (main) of named Table Descriptors.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class

25.1.171.1 Constructors

25.1.171.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.Create Constructor

TnxTablesDescriptor Class

Pascal

```
public constructor Create(
    aParent: TPersistent
); virtual;
```

Remarks

constructor.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Constructors > Create Constructor

25.1.171.2 Destructors

25.1.171.2.1 Destroy Destructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.Destroy Destructor

TnxTablesDescriptor Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Destructors > Destroy Destructor

25.1.171. Methods

25.1.171.3.1 AddTable Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTablesDescriptor.AddTable Method

TnxTablesDescriptor Class

Pascal

```
public function AddTable(
  const aName: string;
  aClass: TnxNestedTableDescriptorClass = nil
): TnxNestedTableDescriptor;
```

Parameters

Parameters	Description
aName	the name of the Table Descriptor
aClass	the class of the instance to be created.

Remarks

This function adds a new instance of a Table Descriptor to the data dictionary and returns the newly created Instance.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > AddTable Method

25.1.171.3.2 CheckValid Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTablesDescriptor.CheckValid Method

TnxTablesDescriptor Class

Pascal

```
public procedure CheckValid; override;
```

Remarks

This is CheckValid, a member of class TnxTablesDescriptor.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > CheckValid Method

25.1.171.3.3 Clear Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTablesDescriptor.Clear Method

TnxTablesDescriptor Class

Pascal

```
public procedure Clear;
```

Remarks

This function clears the Table Descriptors Descriptor.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > Clear Method

25.1.171.3.4 GetTableDescriptorFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.GetTableDescriptorFromName Method

[TnxTablesDescriptor Class](#)

Pascal

```
public function GetTableDescriptorFromName(
  const aName: string
): TnxNestedTableDescriptor;
```

Remarks

Returns the Table descriptor with the given aName.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > GetTableDescriptorFromName Method

25.1.171.3.5 GetTableIndexFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.GetTableIndexFromName Method

[TnxTablesDescriptor Class](#)

Pascal

```
public function GetTableIndexFromName(
  const aName: string
): Integer;
```

Remarks

Return the Table number for a given Table name, or -1 if not found.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > GetTableIndexFromName Method

25.1.171.3.6 GetTables Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.GetTables Method

[TnxTablesDescriptor Class](#)

Pascal

```
public procedure GetTables(
  aStrings: TStrings
);
```

Remarks

Returns a list of names of Table descriptors.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > GetTables Method

25.1.171.3.7 IsEqual Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.IsEqual Method

[TnxTablesDescriptor Class](#)

Pascal

```
public function IsEqual(
    aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxTablesDescriptor.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > IsEqual Method

25.1.171.3.8 LoadFromReader Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.LoadFromReader Method

[TnxTablesDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxTablesDescriptor.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > LoadFromReader Method

25.1.171.3.9 RemoveTable

25.1.171.3.9.1 RemoveTable Method (Integer)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.RemoveTable Method (Integer)

[TnxTablesDescriptor Class](#)

Pascal

```
public procedure RemoveTable(
    aInx: Integer
); overload;
```

Parameters**Parameters**

aInx

Description

the index of the descriptor to be removed in the list of Table Descriptors.

Remarks

Remove a Table Descriptor from the Data Dictionary.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > RemoveTable > RemoveTable Method (Integer)

25.1.171.3.9.2 RemoveTable Method (string)

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.RemoveTable Method (string)

[TnxTablesDescriptor Class](#)

Pascal

```
public procedure RemoveTable(
    const aName: string
); overload;
```

Parameters**Parameters**

aName

Description

the name of the descriptor to be removed.

Remarks

Remove a Table Descriptor from the Data Dictionary.

See Also

[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > RemoveTable > RemoveTable Method (string)

25.1.171.3.10 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTablesDescriptor.SaveToWriter Method

[TnxTablesDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxTablesDescriptor.

See Also[TnxTablesDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTablesDescriptor Class > Methods > SaveToWriter Method

25.1.17 TnxTextKeyFieldDescriptor Class

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxTextKeyFieldDescriptor Class

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)**Pascal**

```
public TnxTextKeyFieldDescriptor = class(TnxKeyFieldDescriptor);
```

File[nxsdDataDictionary](#)**Remarks**

This is a simple Text field version of Key Field descriptor.

See Also[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxTextKeyFieldDescriptor Class

25.1.172 Methods

25.1.172.1.1 IsEqual Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxTextKeyFieldDescriptor.IsEqual Method

[TnxTextKeyFieldDescriptor Class](#)**Pascal**

```
public function IsEqual(
  aDictionaryItem: TnxDictionaryItem
): Boolean; override;
```

Remarks

This is IsEqual, a member of class TnxTextKeyFieldDescriptor.

See Also[TnxTextKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTextKeyFieldDescriptor Class > Methods > IsEqual Method

25.1.172.1.2 LoadFromReader Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxTextKeyFieldDescriptor.LoadFromReader Method

[TnxTextKeyFieldDescriptor Class](#)

Pascal

```
public procedure LoadFromReader(
    aReader: TReader
); override;
```

Remarks

This is LoadFromReader, a member of class TnxTextKeyFieldDescriptor.

See Also

[TnxTextKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTextKeyFieldDescriptor Class > Methods > LoadFromReader Method

25.1.172.1.3 SaveToWriter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTextKeyFieldDescriptor.SaveToWriter Method

[TnxTextKeyFieldDescriptor Class](#)

Pascal

```
public procedure SaveToWriter(
    aWriter: TnxWriter;
    aClientVersion: Integer
); override;
```

Remarks

This is SaveToWriter, a member of class TnxTextKeyFieldDescriptor.

See Also

[TnxTextKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTextKeyFieldDescriptor Class > Methods > SaveToWriter Method

25.1.172. Properties

25.1.172.2.1 IgnoreCase Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTextKeyFieldDescriptor.IgnoreCase Property

[TnxTextKeyFieldDescriptor Class](#)

Pascal

```
public property IgnoreCase: Boolean;
```

Remarks

Set this to true if the sorting should not be case sensitive.

See Also

[TnxTextKeyFieldDescriptor Class](#)

You are here: Symbol Reference > Classes > TnxTextKeyFieldDescriptor Class > Properties > IgnoreCase Property

25.1.17 TnxThreadWithDatabase Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxThreadWithDatabase Class

[nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxThreadWithDatabase = class(TnxThreadWithSession);
```

File

[nxdb](#)

Remarks

This is class TnxThreadWithDatabase.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxThreadWithDatabase Class

25.1.173 Constructors

25.1.173.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxThreadWithDatabase.Create Constructor

TnxThreadWithDatabase Class

Pascal

```
public constructor Create(
  aThreadPriority: TThreadPriority;
  aSessionPool: TnxBaseSessionPool;
  const aAlias: string
);
```

Remarks

This is Create, a member of class TnxThreadWithDatabase.

See Also

[TnxThreadWithDatabase Class](#)

You are here: Symbol Reference > Classes > TnxThreadWithDatabase Class > Constructors > Create Constructor

25.1.173 Properties

25.1.173.2.1 Database Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxThreadWithDatabase.Database Property

TnxThreadWithDatabase Class

Pascal

```
public property Database: TnxDatabase;
```

Remarks

This is Database, a member of class TnxThreadWithDatabase.

See Also

[TnxThreadWithDatabase Class](#)

You are here: Symbol Reference > Classes > TnxThreadWithDatabase Class > Properties > Database Property

25.1.174 TnxThreadWithSession Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxThreadWithSession Class

[nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxThreadWithSession = class(TnxInitThread);
```

File

[nxdb](#)

Remarks

This is class TnxThreadWithSession.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxThreadWithSession Class

25.1.174.1 Constructors

25.1.174.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxThreadWithSession.Create Constructor

TnxThreadWithSession Class

Pascal

```
public constructor Create(
    aThreadPriority: TThreadPriority;
    aSessionPool: TnxBaseSessionPool
);
```

Remarks

This is Create, a member of class TnxThreadWithSession.

See Also

[TnxThreadWithSession Class](#)

You are here: Symbol Reference > Classes > TnxThreadWithSession Class > Constructors > Create Constructor

25.1.174. Properties

25.1.174.2.1 Session Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxThreadWithSession.Session Property

TnxThreadWithSession Class

Pascal

```
public property Session: TnxBaseSession;
```

Remarks

This is Session, a member of class TnxThreadWithSession.

See Also

[TnxThreadWithSession Class](#)

You are here: Symbol Reference > Classes > TnxThreadWithSession Class > Properties > Session Property

25.1.175. TnxThreadWithTable Class

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxThreadWithTable Class

[nxdb](#) | [Members](#) | [Properties](#)

Pascal

```
public TnxThreadWithTable = class(TnxThreadWithDatabase);
```

File

[nxdb](#)

Remarks

This is class TnxThreadWithTable.

See Also

[nxdb](#)
[Members](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxThreadWithTable Class

25.1.175. Constructors

25.1.175.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxThreadWithTable.Create Constructor

TnxThreadWithTable Class

Pascal

```
public constructor Create(
  aThreadPriority: TThreadPriority;
  aSessionPool: TnxBaseSessionPool;
  const aAlias: string;
  const aTableName: string;
  const aIndexName: string;
```

```
    const aPassword: string  
);
```

Remarks

This is Create, a member of class TnxThreadWithTable.

See Also

[TnxThreadWithTable Class](#)

You are here: Symbol Reference > Classes > TnxThreadWithTable Class > Constructors > Create Constructor

25.1.175.Properties

25.1.175.2.1 Table Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxThreadWithTable.Table Property

[TnxThreadWithTable Class](#)

Pascal

```
public property Table: TnxTable;
```

Remarks

This is Table, a member of class TnxThreadWithTable.

See Also

[TnxThreadWithTable Class](#)

You are here: Symbol Reference > Classes > TnxThreadWithTable Class > Properties > Table Property

25.1.17(TnxTransContext Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext Class

[nxdb](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxTransContext = class(TnxSessionOwnedDataAccessStateComponent);
```

File

[nxdb](#)

Remarks

Container for a Transaction Context.

See Also

[nxdb](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxTransContext Class

25.1.176. Constructors

25.1.176.1.1 Create Constructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTransContext.Create Constructor

TnxTransContext Class

Pascal

```
public constructor Create(
    AOwner: TComponent
) ; override;
```

Remarks

constructor.

See Also

[TnxTransContext Class](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Constructors > Create Constructor

25.1.176. Destructors

25.1.176.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTransContext.Destroy Destructor

TnxTransContext Class

Pascal

```
public destructor Destroy; override;
```

Remarks

destructor.

See Also

[TnxTransContext Class](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Destructors > Destroy Destructor

25.1.176. Methods

25.1.176.3.1 Commit Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTransContext.Commit Method

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public procedure Commit;
```

Remarks

This method commits a pending transaction (level). If an error occurs (e.g. deadlock detected or transaction level has been marked as corrupted) the transaction is **not** automatically rolled back. You have to call Rollback in that case.

For more info see the nexus manual.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Methods > Commit Method

25.1.176.3.2 Rollback Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.Rollback Method

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public procedure Rollback;
```

Remarks

Rollback rolls (as the name says) back a pending transaction (level). All changes since the last StartTransaction (or equivalent) call are lost. For more info see the nexus manual.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Methods > Rollback Method

25.1.176.3.3 StartTransaction Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.StartTransaction Method

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public procedure StartTransaction(
  aSnapShot: Boolean = False
);
```

Parameters

Parameters

aSnapShot

Description

If true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

StartTransaction tries to start a transaction. If it fails an exception is triggered.

Related Topics

StartTransaction, StartTransactionWith, TryStartTransaction, Commit, Rollback, InTransaction, TransactionCorrupted

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Methods > StartTransaction Method

25.1.176.3.4 StartTransactionWith Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.StartTransactionWith Method

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public procedure StartTransactionWith(
  const aTables: array of TnxTable;
  aSnapshot: Boolean = False
);
```

Parameters

Parameters	Description
aTables	the tables that should be part of the transaction
aSnapshot	if true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

Start a transaction if an exclusive lock can be granted on all specified tables. If the grant fails one or more tables or an error occurs, an exception is raised.

Related Topics

[StartTransaction](#), [StartTransactionWithEx](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Methods > StartTransactionWith Method

25.1.176.3.5 StartTransactionWithEx Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.StartTransactionWithEx Method

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public function StartTransactionWithEx(
  const aTables: array of TnxTable;
  aSnapshot: Boolean = False
): TnxResult;
```

Parameters

Parameters	Description
------------	-------------

aTables	the tables that should be part of the transaction
aSnapshot	if true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

Start a transaction if an exclusive lock can be granted on all specified tables. If the grant fails one or more tables or an error occurs, an error code is returned.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxTransContext Class](#) > [Methods](#) > [StartTransactionWithEx Method](#)

25.1.176.3.6 TransactionCorrupted Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.TransactionCorrupted Method

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public procedure TransactionCorrupted;
```

Remarks

This method marks the transaction (level) as corrupted and thus preventing it from ever being committed. For more info see the nexus manual.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: [Symbol Reference](#) > [Classes](#) > [TnxTransContext Class](#) > [Methods](#) > [TransactionCorrupted Method](#)

25.1.176.3.7 TryStartTransaction Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.TryStartTransaction Method

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public function TryStartTransaction(
  aSnapshot: Boolean = False
): Boolean;
```

Parameters

Parameters

Description

aSnapShot

if true it is handled as a SnapshotTransaction; for more info see the nexus manual

Remarks

Tries start a transaction (if none was started by the current TnxDatabase), returns true if a new transaction was started, false if a transaction was already started by in this context or triggers an exception if an error occurs.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Methods > TryStartTransaction Method

25.1.176.4 Properties

25.1.176.4.1 DatabaseCount Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.DatabaseCount Property

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public property DatabaseCount: Integer;
```

Remarks

Returns the number of dataset instances attached to this database.

Related Topics

[DataSetCount](#), [DataSets](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Properties > DatabaseCount Property

25.1.176.4.2 Databases Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.Databases Property

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public property Databases [aInx: Integer]: TnxDatabase;
```

Remarks

Returns the dataset with index aInx out of the list of attached dataset instances.

Related Topics

[DatabaseCount](#), [Databases](#), [TableCount](#), [Tables](#), [QueryCount](#), [Queries](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Properties > Databases Property

25.1.176.4.3 Default Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.Default Property

[TnxTransContext Class](#)

Pascal

```
published property Default: Boolean;
```

Remarks

This is Default, a member of class TnxTransContext.

See Also

[TnxTransContext Class](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Properties > Default Property

25.1.176.4.4 FailSafe Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.FailSafe Property

[TnxTransContext Class](#)

Pascal

```
published property FailSafe: Boolean;
```

Remarks

Sets failsafe mode on or off. For more info see NexusDB manual.

See Also

[TnxTransContext Class](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Properties > FailSafe Property

25.1.176.4.5 InTransaction Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransContext.InTransaction Property

[TnxTransContext Class](#) | [Related Topics](#)

Pascal

```
public property InTransaction: Boolean;
```

Remarks

Returns true if a transaction for the current database is active otherwise false.

Related Topics

[StartTransaction](#), [StartTransactionWith](#), [TryStartTransaction](#), [Commit](#), [Rollback](#), [InTransaction](#), [TransactionCorrupted](#)

See Also

[TnxTransContext Class](#)
[Related Topics](#)

You are here: Symbol Reference > Classes > TnxTransContext Class > Properties > InTransaction Property

25.1.17 TnxWinsockTransport Class

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport Class

[nxtwWinsockTransport](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public TnxWinsockTransport = class(TnxBasePooledTransport);
```

File

[nxtwWinsockTransport](#)

Remarks

Transport based on Winsock 2.

See Also

[nxtwWinsockTransport](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class

25.1.17.1 Constructors

25.1.17.1.1 Create Constructor

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport.Create Constructor

[TnxWinsockTransport Class](#)

Pascal

```
public constructor Create(
  aOwner: TComponent
); override;
```

Remarks

This is Create, a member of class TnxWinsockTransport.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Constructors > Create Constructor

25.1.177.:Destructors

25.1.177.2.1 Destroy Destructor

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxWinsockTransport.Destroy Destructor

TnxWinsockTransport Class

Pascal

```
public destructor Destroy; override;
```

Remarks

This is Destroy, a member of class TnxWinsockTransport.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Destructors > Destroy Destructor

25.1.177.:Methods

25.1.177.3.1 BeginBroadcast Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxWinsockTransport.BeginBroadcast Method

TnxWinsockTransport Class

Pascal

```
public function BeginBroadcast(
    aHandler: InxBroadcastReplyHandler;
    aInterval: Integer
): InxActiveBroadcast; override;
```

Parameters

Parameters

aList

Description

gets the list of servers responding

aTimeout

the time the method waits for servers to respond

Remarks

Gets a list of available servers by broadcasting.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > BeginBroadcast Method

25.1.177.3.2 GetBoundAddresses Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxWinsockTransport.GetBoundAddresses Method

TnxWinsockTransport Class

Pascal

```
public procedure GetBoundAddresses (
    aList: TStrings
```

```
); override;
```

Remarks

Returns in aList the list of IP addresses the transport is bound to.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > [GetBoundAddresses Method](#)

25.1.177.3.3 GetConfigSettings Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport.GetConfigSettings Method

[TnxWinsockTransport Class](#)

Pascal

```
public procedure GetConfigSettings(
    aSettings: TnxBaseSettings
); override;
```

Remarks

This is GetConfigSettings, a member of class TnxWinsockTransport.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > [GetConfigSettings Method](#)

25.1.177.3.4 GetName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport.GetName Method

[TnxWinsockTransport Class](#)

Pascal

```
public function GetName: string; override;
```

Remarks

This is GetName, a member of class TnxWinsockTransport.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > [GetName Method](#)

25.1.177.3.5 LoadConfig Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport.LoadConfig Method

[TnxWinsockTransport Class](#)

Pascal

```
public procedure LoadConfig(
```

```
aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

This is LoadConfig, a member of class TnxWinsockTransport.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > LoadConfig Method

25.1.177.3.6 LoadSettingsFromStream Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport.LoadSettingsFromStream Method

[TnxWinsockTransport Class](#)

Pascal

```
public procedure LoadSettingsFromStream(
    aReader: TnxReader
); override;
```

Remarks

This is LoadSettingsFromStream, a member of class TnxWinsockTransport.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > LoadSettingsFromStream Method

25.1.177.3.7 SaveConfig Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport.SaveConfig Method

[TnxWinsockTransport Class](#)

Pascal

```
public procedure SaveConfig(
    aConfig: TnxBaseComponentConfiguration
); override;
```

Remarks

This is SaveConfig, a member of class TnxWinsockTransport.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > SaveConfig Method

25.1.177.3.8 SaveSettingsToStream Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxWinsockTransport.SaveSettingsToStream Method

[TnxWinsockTransport Class](#)

Pascal

```
public procedure SaveSettingsToStream(
    aWriter: TnxWriter
); override;
```

■ **Remarks**

This is SaveSettingsToStream, a member of class TnxWinsockTransport.

■ **See Also**

TnxWinsockTransport Class

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Methods > SaveSettingsToStream Method

25.1.177.4 Properties

25.1.177.4.1 BroadcastThreadPriority Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxWinsockTransport.BroadcastThreadPriority Property

TnxWinsockTransport Class

Pascal

```
published property BroadcastThreadPriority: TThreadPriority;
```

■ **Remarks**

The priority of the server-side broadcast thread which responds to client broadcasts.

■ **See Also**

TnxWinsockTransport Class

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Properties > BroadcastThreadPriority Property

25.1.177.4.2 CallbackThreadCount Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxWinsockTransport.CallbackThreadCount Property

TnxWinsockTransport Class

Pascal

```
public property CallbackThreadCount: Byte;
```

■ **Remarks**

0: messages from server to client is not possible. higher: maximum number of messages a client can process at the same time

■ **See Also**

TnxWinsockTransport Class

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Properties > CallbackThreadCount Property

25.1.177.4.3 CallbackThreadPriority Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxWinsockTransport.CallbackThreadPriority Property

[TnxWinsockTransport Class](#)**Pascal**

```
public property CallbackThreadPriority: TThreadPriority;
```

Remarks

The priority of the callback thread.

See Also[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Properties > [CallbackThreadPriority Property](#)

[25.1.177.4.4 ConnectionFilter Property](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxWinsockTransport.ConnectionFilter Property**[TnxWinsockTransport Class](#)**Pascal**

```
published property ConnectionFilter: TnxBaseConnectionFilter;
```

Remarks

Use if you wish to filter connections by their IP address.

See Also[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Properties > [ConnectionFilter Property](#)

[25.1.177.4.5 ListenAddresses Property](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxWinsockTransport.ListenAddresses Property**[TnxWinsockTransport Class](#)**Pascal**

```
published property ListenAddresses: TStrings;
```

Remarks

This is ListenAddresses, a member of class TnxWinsockTransport.

See Also[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Properties > [ListenAddresses Property](#)

[25.1.177.4.6 ListenThreadPriority Property](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxWinsockTransport.ListenThreadPriority Property**[TnxWinsockTransport Class](#)**Pascal**

```
published property ListenThreadPriority: TThreadPriority;
```

Remarks

The priority of the listen thread.

See Also

[TnxWinsockTransport Class](#)

You are here: Symbol Reference > Classes > TnxWinsockTransport Class > Properties > ListenThreadPriority Property

25.2 Interfaces

25.2.1 InxActiveBroadcast Interface

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxActiveBroadcast Interface

[nxllTransport](#) | [Members](#) | [Methods](#)

Pascal

```
public InxActiveBroadcast = interface;
```

File

[nxllTransport](#)

Remarks

This interface is returned by `TnxBaseTransport.BeginBroadcast`.

See Also

[nxllTransport](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Interfaces > InxActiveBroadcast Interface

25.2.1.1 Methods

25.2.1.1.1 Resume Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxActiveBroadcast.Resume Method

[InxActiveBroadcast Interface](#)

Pascal

```
procedure Resume;
```

Remarks

Resumes broadcasts.

See Also

[InxActiveBroadcast Interface](#)

You are here: Symbol Reference > Interfaces > InxActiveBroadcast Interface > Methods > Resume Method

25.2.1.1.2 Stop Method

NexusDB V2 VCL Reference[Contents | Index](#)**InxActiveBroadcast.Stop Method**[InxActiveBroadcast Interface](#)**Pascal**

```
procedure Stop;
```

Remarks

Stops all broadcasts and stops listening for replies.

See Also[InxActiveBroadcast Interface](#)

You are here: Symbol Reference > Interfaces > InxActiveBroadcast Interface > Methods > Stop Method

25.2.1.1.3 Suspend Method

NexusDB V2 VCL Reference[Contents | Index](#)**InxActiveBroadcast.Suspend Method**[InxActiveBroadcast Interface](#)**Pascal**

```
procedure Suspend;
```

Remarks

Suspends further broadcasts but continues to list for replies,

See Also[InxActiveBroadcast Interface](#)

You are here: Symbol Reference > Interfaces > InxActiveBroadcast Interface > Methods > Suspend Method

25.2.2 InxBroadcastReply Interface

NexusDB V2 VCL Reference[Contents | Index](#)**InxBroadcastReply Interface**[nxIITransport](#) | [Members](#) | [Methods](#)**Pascal**

```
public InxBroadcastReply = interface;
```

File[nxIITransport](#)**Remarks**

This interface is passed into InxBroadcastReplyHandler.ReplyReceived.

See Also[nxIITransport](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Interfaces > InxBroadcastReply Interface

25.2.2.1 Methods

25.2.2.1.1 GetServerID Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxBroadcastReply.GetServerID Method

InxBroadcastReply Interface

Pascal

```
function GetServerID: TnxGuid;
```

Remarks

GUID of the server. Can be 0 (for < 2.0 servers). The GUID will be identical for replies on different transports from the same server.

See Also

[InxBroadcastReply Interface](#)

You are here: Symbol Reference > Interfaces > InxBroadcastReply Interface > Methods > [GetServerID Method](#)

25.2.2.1.2 GetServerName Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxBroadcastReply.GetServerName Method

InxBroadcastReply Interface

Pascal

```
function GetServerName: string;
```

Remarks

Transport dependant name of the server.

See Also

[InxBroadcastReply Interface](#)

You are here: Symbol Reference > Interfaces > InxBroadcastReply Interface > Methods > [GetServerName Method](#)

25.2.2.1.3 GetServerVersion Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxBroadcastReply.GetServerVersion Method

InxBroadcastReply Interface

Pascal

```
function GetServerVersion: Integer;
```

Remarks

Can be 0 (for < 2.0 server). Otherwise it specifies the Version of the server.

See Also

[InxBroadcastReply Interface](#)

You are here: Symbol Reference > Interfaces > InxBroadcastReply Interface > Methods > [GetServerVersion Method](#)

25.2.3 InxBroadcastReplyHandler Interface

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxBroadcastReplyHandler Interface

[nxIITransport](#) | [Members](#) | [Methods](#)

Pascal

```
public InxBroadcastReplyHandler = interface;
```

File

[nxIITransport](#)

Remarks

This interface has to be passed into TnxBaseTransport.BeginBroadcast.

See Also

[nxIITransport](#)

[Members](#)

[Methods](#)

You are here: Symbol Reference > Interfaces > InxBroadcastReplyHandler Interface

25.2.3.1 Methods

25.2.3.1.1 ReplyReceived Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxBroadcastReplyHandler.ReplyReceived Method

InxBroadcastReplyHandler Interface

Pascal

```
procedure ReplyReceived(
    aReply: InxBroadcastReply
);
```

Remarks

Called each time the Transport receives a reply to the broadcast. May be called multiple times for the same server.

See Also

[InxBroadcastReplyHandler Interface](#)

You are here: Symbol Reference > Interfaces > InxBroadcastReplyHandler Interface > Methods > ReplyReceived Method

25.2.4 InxFieldDescriptor Interface

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxFieldDescriptor Interface

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public InxFieldDescriptor = interface(InxInterface);
```

File

[nxsdDataDictionary](#)

Remarks

This is class InxFieldDescriptor.

See Also

[nxsdDataDictionary](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Interfaces > InxFieldDescriptor Interface

25.2.4.1 Methods

25.2.4.1.1 GetFieldDecPI Method

NexusDB V2 VCL Reference

[Contents | Index](#)

InxFieldDescriptor.GetFieldDecPI Method

InxFieldDescriptor Interface

Pascal

```
function GetFieldDecPI: Integer;
```

Remarks

This is GetFieldDecPI, a member of class InxFieldDescriptor.

See Also

[InxFieldDescriptor Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldDescriptor Interface > Methods > GetFieldDecPI Method

25.2.4.1.2 GetFieldType Method

NexusDB V2 VCL Reference

[Contents | Index](#)

InxFieldDescriptor.GetFieldType Method

InxFieldDescriptor Interface

Pascal

```
function GetFieldType: TnxFieldType;
```

Remarks

This is GetFieldType, a member of class InxFieldDescriptor.

See Also

[InxFieldDescriptor Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldDescriptor Interface > Methods > GetFieldType Method

25.2.4.1.3 GetFieldUnits Method

NexusDB V2 VCL Reference

[Contents | Index](#)

InxFieldDescriptor.GetFieldUnits Method

InxFieldDescriptor Interface

Pascal

```
function GetFieldUnits: Integer;
```

Remarks

This is GetFieldUnits, a member of class InxFieldDescriptor.

See Also

[InxFieldDescriptor Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldDescriptor Interface > Methods > GetFieldUnits Method

25.2.4.2 Properties

25.2.4.2.1 FieldDecPI Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldDescriptor.FieldDecPI Property

[InxFieldDescriptor Interface](#)

Pascal

```
property FieldDecPI: Integer;
```

Remarks

This is FieldDecPI, a member of class InxFieldDescriptor.

See Also

[InxFieldDescriptor Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldDescriptor Interface > Properties > FieldDecPI Property

25.2.4.2.2 FieldType Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldDescriptor.FieldType Property

[InxFieldDescriptor Interface](#)

Pascal

```
property FieldType: TnxFieldType;
```

Remarks

This is FieldType, a member of class InxFieldDescriptor.

See Also

[InxFieldDescriptor Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldDescriptor Interface > Properties > FieldType Property

25.2.4.2.3 FieldUnits Property

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldDescriptor.FieldUnits Property

[InxFieldDescriptor Interface](#)

Pascal

```
property FieldUnits: Integer;
```

Remarks

This is FieldUnits, a member of class InxFieldDescriptor.

See Also

[InxFieldDescriptor Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldDescriptor Interface > Properties > FieldUnits Property

25.2.5 InxFieldsSource Interface

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource Interface

[nxsdDataDictionary](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public InxFieldsSource = interface(InxInterface);
```

File

[nxsdDataDictionary](#)

Remarks

This is class InxFieldsSource.

See Also

[nxsdDataDictionary](#)

[Members](#)

[Methods](#)

[Properties](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface

25.2.5.1 Methods

25.2.5.1.1 BlobsSupported Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource.BlobsSupported Method

InxFieldsSource Interface

Pascal

```
function BlobsSupported: Boolean;
```

Remarks

This is BlobsSupported, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Methods > BlobsSupported Method

25.2.5.1.2 GetCursor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource.GetCursor Method

InxFieldsSource Interface

Pascal
function GetCursor: TObject;

Remarks

This is GetCursor, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Methods > GetCursor Method

25.2.5.1.3 GetFieldAsVariant Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource.GetFieldAsVariant Method**InxFieldsSource Interface**

Pascal
function GetFieldAsVariant(
 aBuffer: PnxByteArray;
 aField: Integer
): OleVariant;

Remarks

This is GetFieldAsVariant, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Methods > GetFieldAsVariant Method

25.2.5.1.4 GetFieldCount Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource.GetFieldCount Method**InxFieldsSource Interface**

Pascal
function GetFieldCount: Integer;

Remarks

This is GetFieldCount, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Methods > GetFieldCount Method

25.2.5.1.5 GetFieldDescriptor Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource.GetFieldDescriptor Method

InxFieldsSource Interface

Pascal

```
function GetFieldDescriptor(
    aIndex: Integer
): InxFieldDescriptor;
```

Remarks

This is GetFieldDescriptor, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Methods > [GetFieldDescriptor Method](#)

25.2.5.1.6 GetFieldForFilter Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource.GetFieldForFilter Method**InxFieldsSource Interface**

Pascal

```
function GetFieldForFilter(
    aField: Integer;
    aName: PChar;
    aData: PnxByteArray;
    out aType: TnxWord16;
    out aSize: TnxWord16;
    out aIsNull: Boolean;
    out aValue: Pointer
): Boolean;
```

Remarks

This is GetFieldForFilter, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Methods > [GetFieldForFilter Method](#)

25.2.5.1.7 GetFieldFromName Method

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

InxFieldsSource.GetFieldFromName Method**InxFieldsSource Interface**

Pascal

```
function GetFieldFromName(
    const aFieldName: string
): Integer;
```

Remarks

This is GetFieldFromName, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Methods > GetFieldFromName Method

25.2.5.2 Properties

25.2.5.2.1 FieldCount Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxFieldsSource.FieldCount Property

InxFieldsSource Interface

Pascal

```
property FieldCount: Integer;
```

Remarks

This is FieldCount, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Properties > FieldCount Property

25.2.5.2.2 FieldDescriptor Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxFieldsSource.FieldDescriptor Property

InxFieldsSource Interface

Pascal

```
property FieldDescriptor [aIndex: Integer]: InxFieldDescriptor;
```

Remarks

This is FieldDescriptor, a member of class InxFieldsSource.

See Also

[InxFieldsSource Interface](#)

You are here: Symbol Reference > Interfaces > InxFieldsSource Interface > Properties > FieldDescriptor Property

25.2.6 InxLogData Interface

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxLogData Interface

[nxlComponent](#) | [Members](#) | [Methods](#) | [Properties](#)

Pascal

```
public InxLogData = interface(InxInterface);
```

File

[nxlComponent](#)

Remarks

This is class InxLogData.

See Also

[nxIIComponent](#)
[Members](#)
[Methods](#)
[Properties](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface

25.2.6.1 Methods

25.2.6.1.1 GetCategory Method

NexusDB V2 VCL Reference

[Contents | Index](#)

InxLogData.GetCategory Method

InxLogData Interface

Pascal

```
function GetCategory: string;
```

Remarks

This is GetCategory, a member of class InxLogData.

See Also

[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Methods > GetCategory Method

25.2.6.1.2 GetMsg Method

NexusDB V2 VCL Reference

[Contents | Index](#)

InxLogData.GetMsg Method

InxLogData Interface

Pascal

```
function GetMsg: string;
```

Remarks

This is GetMsg, a member of class InxLogData.

See Also

[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Methods > GetMsg Method

25.2.6.1.3 GetMsgNumber Method

NexusDB V2 VCL Reference

[Contents | Index](#)

InxLogData.GetMsgNumber Method

InxLogData Interface

Pascal

```
function GetMsgNumber: cardinal;
```

Remarks

This is GetMsgNumber, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Methods > GetMsgNumber Method

25.2.6.1.4 GetPriority Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**InxLogData.GetPriority Method**[InxLogData Interface](#)**Pascal**

```
function GetPriority: TnxLogPriority;
```

Remarks

This is GetPriority, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Methods > GetPriority Method

25.2.6.1.5 GetUTCWhen Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**InxLogData.GetUTCWhen Method**[InxLogData Interface](#)**Pascal**

```
function GetUTCWhen: TDateTime;
```

Remarks

This is GetUTCWhen, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Methods > GetUTCWhen Method

25.2.6.1.6 GetWhen Method

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**InxLogData.GetWhen Method**[InxLogData Interface](#)**Pascal**

```
function GetWhen: TDateTime;
```

Remarks

This is GetWhen, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Methods > GetWhen Method

25.2.6.1.7 GetWhenBias Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxLogData.GetWhenBias Method

InxLogData Interface

Pascal

```
function GetWhenBias: Integer;
```

Remarks

This is GetWhenBias, a member of class InxLogData.

See Also

[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Methods > GetWhenBias Method

25.2.6.2 Properties

25.2.6.2.1 Category Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxLogData.Category Property

InxLogData Interface

Pascal

```
property Category: string;
```

Remarks

This is Category, a member of class InxLogData.

See Also

[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Properties > Category Property

25.2.6.2.2 Msg Property

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxLogData.Msg Property

InxLogData Interface

Pascal

```
property Msg: string;
```

Remarks

This is Msg, a member of class InxLogData.

See Also

[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Properties > Msg Property

25.2.6.2.3 MsgNumber Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**InxLogData.MsgNumber Property**[InxLogData Interface](#)**Pascal**

```
property MsgNumber: cardinal;
```

Remarks

This is MsgNumber, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Properties > MsgNumber Property

25.2.6.2.4 Priority Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**InxLogData.Priority Property**[InxLogData Interface](#)**Pascal**

```
property Priority: TnxLogPriority;
```

Remarks

This is Priority, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Properties > Priority Property

25.2.6.2.5 UTCWhen Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**InxLogData.UTCWhen Property**[InxLogData Interface](#)**Pascal**

```
property UTCWhen: TDateTime;
```

Remarks

This is UTCWhen, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Properties > UTCWhen Property

25.2.6.2.6 When Property

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**InxLogData.When Property**

[InxLogData Interface](#)

Pascal
`property When: TDateTime;`

Remarks

This is When, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Properties > When Property

[25.2.6.2.7 WhenBias Property](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)[InxLogData.WhenBias Property](#)[InxLogData Interface](#)

Pascal
`property WhenBias: Integer;`

Remarks

This is WhenBias, a member of class InxLogData.

See Also[InxLogData Interface](#)

You are here: Symbol Reference > Interfaces > InxLogData Interface > Properties > WhenBias Property

[25.2.7 InxSessionCallback Interface](#)[NexusDB V2 VCL Reference](#)[Contents | Index](#)[InxSessionCallback Interface](#)[nxIITransport](#) | [Members](#) | [Methods](#)

Pascal
`public InxSessionCallback = interface(InxInterface);`

File[nxIITransport](#)**Remarks**

This is class InxSessionCallback.

See Also[nxIITransport](#)
[Members](#)
[Methods](#)

You are here: Symbol Reference > Interfaces > InxSessionCallback Interface

25.2.7.1 Methods

25.2.7.1.1 Post Method

NexusDB V2 VCL Reference

InxSessionCallback.Post Method

[Contents](#) | [Index](#)

InxSessionCallback Interface

Pascal

```
procedure Post(
    aThreadPriority: TnxThreadPriority;
    aMsgID: TnxMsgID;
    arequestData: Pointer;
    arequestDataLen: TnxWord32;
    aTimeOut: Integer
);
```

Parameters

Parameters	Description
aMsgID	the message id
arequestData	the data of the broadcast request
arequestDataLen	the length of arequestData
aTimeOut	timeout in msec

Remarks

This functions posts a request and does NOT wait for the answer.

See Also

[InxSessionCallback Interface](#)

You are here: Symbol Reference > Interfaces > InxSessionCallback Interface > Methods > Post Method

25.2.7.1.2 Request Method

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

InxSessionCallback.Request Method

InxSessionCallback Interface

Pascal

```
procedure Request(
    aThreadPriority: TnxThreadPriority;
    aMsgID: TnxMsgID;
    aTimeOut: Integer;
    arequestData: Pointer;
    arequestDataLen: TnxWord32;
    aReplyCallback: TnxReplyCallback;
    aReplyCookie: Integer
);
```

Parameters

Parameters	Description
aMsgID	the message id
aTimeOut	timeout in msec
arequestData	the data of the broadcast request

arequestDataLen	the length of arequestData
aReplyCallback	the callback function to be called once the reply arrived.
aReplyCookie	additional information received.

Remarks

This functions sends a request and waits for the answer.

See Also

[InxSessionCallback Interface](#)

You are here: Symbol Reference > Interfaces > InxSessionCallback Interface > Methods > Request Method

25.3 Functions

25.3.1 _DoNothingStub Function

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

_DoNothingStub Function

[nxsdServerEngine](#)

Pascal

```
procedure _DoNothingStub;
```

File

[nxsdServerEngine](#)

Remarks

internal use only

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Functions > _DoNothingStub Function

25.3.2 _DoNothingStub2 Function

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

_DoNothingStub2 Function

[nxsdServerEngine](#)

Pascal

```
function _DoNothingStub2: PnxBoolean;
```

File

[nxsdServerEngine](#)

Remarks

internal use only

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Functions > _DoNothingStub2 Function

25.3.3 nxCheckValidAliasName Function

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxCheckValidAliasName Function

nxsdServerEngine

Pascal

```
function nxCheckValidAliasName(
  const aAliasName: string;
  aEmptyAsUnknown: Boolean
): TnxResult;
```

File

nxsdServerEngine

Remarks

This is function nxCheckValidAliasName.

See Also

nxsdServerEngine

You are here: Symbol Reference > Functions > nxCheckValidAliasName Function

25.3.4 nxCheckValidRelativeTableName Function

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxCheckValidRelativeTableName Function

nxsdServerEngine

Pascal

```
function nxCheckValidRelativeTableName(
  const aTableName: string
): TnxResult;
```

File

nxsdServerEngine

Remarks

This is function nxCheckValidRelativeTableName.

See Also

nxsdServerEngine

You are here: Symbol Reference > Functions > nxCheckValidRelativeTableName Function

25.3.5 nxCheckValidRootTableName Function

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxCheckValidRootTableName Function

nxsdServerEngine

Pascal

```
function nxCheckValidRootTableName(
  const aTableName: string;
  aEmptyAsUnknown: Boolean;
  aAllowSystem: Boolean = False
```

```
): TnxResult;
```

File

[nxsdServerEngine](#)

Remarks

This is function nxCheckValidRootTableName.

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Functions > nxCheckValidRootTableName Function

25.3.6 nxCheckValidStoredProcName Function

NexusDB V2 VCL Reference

[Contents | Index](#)

nxCheckValidStoredProcName Function

[nxsdServerEngine](#)

Pascal

```
function nxCheckValidStoredProcName(
    const aStoredProcName: string;
    aEmptyAsUnknown: Boolean
): TnxResult;
```

File

[nxsdServerEngine](#)

Remarks

This is function nxCheckValidStoredProcName.

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Functions > nxCheckValidStoredProcName Function

25.3.7 nxCheckValidTableName Function

NexusDB V2 VCL Reference

[Contents | Index](#)

nxCheckValidTableName Function

[nxsdServerEngine](#)

Pascal

```
function nxCheckValidTableName(
    const aTableName: string;
    aEmptyAsUnknown: Boolean;
    aAllowSystem: Boolean = False
): TnxResult;
```

File

[nxsdServerEngine](#)

Remarks

This is function nxCheckValidTableName.

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Functions > nxCheckValidTableName Function

25.3.8 nxSplitAddress Function

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxSplitAddress Function

[nxIITransport](#)

Pascal

```
procedure nxSplitAddress(
    aOriginal: string;
    var aName: string;
    var aAddress: string
);
```

File

[nxIITransport](#)

Remarks

This function separates the server name in form name@address.

See Also

[nxIITransport](#)

You are here: Symbol Reference > Functions > nxSplitAddress Function

25.3.9 nxTableNameIsTempDatabase Function

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxTableNameIsTempDatabase Function

[nxsdServerEngine](#)

Pascal

```
function nxTableNameIsTempDatabase (
    const aTableName: string
): Boolean;
```

File

[nxsdServerEngine](#)

Remarks

returns true if the given aTableName is a database specific temp table

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Functions > nxTableNameIsTempDatabase Function

25.3.10 nxTableNameIsTempGlobal Function

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxTableNameIsTempGlobal Function

[nxsdServerEngine](#)

Pascal

```
function nxTableNameIsTempGlobal(
    const aTableName: string
): Boolean;
```

File[nxsdServerEngine](#)**Remarks**

returns true if the given aTableName is a global temp table

See Also[nxsdServerEngine](#)*You are here:* Symbol Reference > Functions > nxTableNameIsTempGlobal Function

25.3.11 nxTableNameIsTempStatement Function

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

nxTableNameIsTempStatement Function

[nxsdServerEngine](#)**Pascal**

```
function nxTableNameIsTempStatement(
    const aTableName: string
): Boolean;
```

File[nxsdServerEngine](#)**Remarks**

returns true if the given aTableName is a temp statement

See Also[nxsdServerEngine](#)*You are here:* Symbol Reference > Functions > nxTableNameIsTempStatement Function

25.4 Structs, Records, Enums

25.4.1 Tnx1xFileType Enumeration

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

Tnx1xFileType Enumeration

[nxsdDataDictionary](#)**Pascal**

```
public Tnx1xFileType = (public ftBaseFile, public ftIndexFile, public ftBlobFile);
```

File[nxsdDataDictionary](#)**Members**

Members	Description
ftBaseFile	..base file: at least data & dictionary.
ftIndexFile	..index file.

ftBlobFile	..Blob file.
------------	--------------

Remarks

File types for the data dictionary.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Structs, Records, Enums > Tnx1xFileType Enumeration

25.4.2 TnxApplyAt Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxApplyAt Enumeration

[nxsdDataDictionary](#)

Pascal

```
public TnxApplyAt = (public aaServer, public aaClient);
```

File

[nxsdDataDictionary](#)

Members

Members	Description
aaServer	on the server
aaClient	on the client

Remarks

Type used to indicate where a validation object should be applied.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxApplyAt Enumeration

25.4.3 TnxBaseHeader Record

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseHeader Record

[nxllTransport](#)

Pascal

```
public TnxBaseHeader = packed record
  public bhTransportID: TnxTransportID;
  public bhSessionID: TnxSessionID;
  public bhVersion: TnxByte8;
  public bhThreadPriority: TnxThreadPriority;
  public bhFlags: TnxMessageHeaderFlags;
  public bhReserved3: Byte;
  public bhMsgID: TnxMsgID;
  public bhLength: TnxWord32;
  public bhParts: TnxWord32;
  public bhErrorCode: TnxWord16;
  public bhMsgType: Byte;
  public bhCompressType: Byte;
  public bhUncompressedSize: TnxWord32;
```

```
end;
```

File

`nxlITransport`

Members

Members	Description
<code>bhTransportID</code>	.. the id of the transport the message came from.
<code>bhSessionID</code>	.. the session id.
<code>bhVersion</code>	.. the request id.
<code>bhThreadPriority</code>	remote thread priority to use, if a the message request is executed in a separate thread
<code>bhFlags</code>	flags sent with the message
<code>bhReserved3</code>	reserved for future use
<code>bhMsgID</code>	.. the message id.
<code>bhLength</code>	.. the length of the message including the headers.
<code>bhParts</code>	.. the number of message parts for the message consists of.
<code>bhErrorCode</code>	.. the error code.
<code>bhMsgType</code>	.. the message type.
<code>bhCompressType</code>	.. the compression type for the message data
<code>bhUncompressedSize</code>	.. the size of the uncompressed full message.

Remarks

The base header for a communication messages.

See Also

`nxlITransport`

You are here: Symbol Reference > Structs, Records, Enums > `TnxBaseHeader Record`

25.4.4 TnxBlobCopyMode Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBlobCopyMode Enumeration

`nxsdTypes`

Pascal

```
public TnxBlobCopyMode = (public nxbcmNoCopy, public nxbcmCopyFull, public nxbcmCreateLink);
```

File

`nxsdTypes`

Members

Members	Description
<code>nxbcmNoCopy</code>	.. do not handle blobs.
<code>nxbcmCopyFull</code>	.. copy blobs into result set.
<code>nxbcmCreateLink</code>	.. copy reference to blob into result set.

Remarks

Defines if Blobs should be linked, copied or left out of SQL results.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > [TnxBlobCopyMode Enumeration](#)

25.4.5 TnxBookmark Record

[NexusDB V2 VCL Reference](#)

TnxBookmark Record

[Contents](#) | [Index](#)

[nxsdTypes](#)

Pascal

```
public TnxBookmark = packed record
  public bmHash: TnxWord32;
  public bmIndexID: Integer;
  public bmPos: TnxIndexPathPosition;
  public bmKeyValid: Boolean;
  public bmDeleted: Boolean;
  public bmFill11: Byte;
  public bmRefNr: TnxInt64;
  public bmKeyLen: Integer;
  public bmKey: array [0..1] of Byte;
end;
```

File

[nxsdTypes](#)

Members

Members	Description
bmHash	Bookmark Hashcode
bmIndexID	ID of index the bookmark belongs to
bmPos	current position in the index
bmKeyValid	true if the bookmark key is valid
bmDeleted	true if the record of the bookmark was deleted
bmFill11	filler byte for alignment
bmRefNr	ref nr of the record
bmKeyLen	length of bookmark key
bmKey	bookmark key

Remarks

Bookmark data definition

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > [TnxBookmark Record](#)

25.4.6 TnxCachedDataSetOption Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCachedDataSetOption Enumeration

nxdb

Pascal

```
public TnxCachedDataSetOption = (public cdsoCopyIndices, public cdsoSetVisibleFields);
```

File

nxdb

Members

Members

Members	Description
cdsoCopyIndices	copy indices from source.

Description

Remarks

Possible options for a cached dataset.

See Also

nxdb

You are here: Symbol Reference > Structs, Records, Enums > TnxCachedDataSetOption Enumeration

25.4.7 TnxClassListType Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxClassListType Enumeration

nxsdTypes

Pascal

```
public TnxClassListType = (public setAutoIncEngines, public setBlobEngines, public setIndicesE
```

File

nxsdTypes

Members

Members

Members	Description
setAutoIncEngines	list holds AutoInc Engines

Description

setBlobEngines	list holds Blob Engines
----------------	-------------------------

setIndicesEngines	list holds Index Engines
-------------------	--------------------------

setEncryptionEngines	list holds Encryption Engines
----------------------	-------------------------------

setDefaultDescriptors	list holds Default Descriptors
-----------------------	--------------------------------

setValidationDescriptors	list holds Validation Descriptors
--------------------------	-----------------------------------

setRecordDescriptors	list holds Record Descriptors
----------------------	-------------------------------

setRIActionDescriptors	list holds Ref Integrity Action Descriptors
------------------------	---

Remarks

possible types for class lists

See Also

nxsdTypes

You are here: Symbol Reference > Structs, Records, Enums > TnxClassListType Enumeration

25.4.8 TnxDataMessage Record

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataMessage Record

nxlITransport

Pascal

```
public TnxDataMessage = record
  public dmMsg: TnxMsgID;
  public dmSource: TObject;
  public dmSessionID: TnxSessionID;
  public dmErrorCode: TnxResult;
  public dmData: Pointer;
  public dmDataLen: TnxMemSize;
end;
```

File

nxlITransport

Members

Members	Description
dmMsg	.. the message id which must be unique in the transport context.
dmSource	.. the source of the message.
dmSessionID	.. the session id.
dmErrorCode	.. the request ID the message belongs to.
dmData	the data of the message.
dmDataLen	the length of the data.

Remarks

the structure for storing one data message part.

See Also

nxlITransport

You are here: Symbol Reference > Structs, Records, Enums > TnxDataMessage Record

25.4.9 TnxDataSetInternalState Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataSetInternalState Enumeration

nxdb

Pascal

```
public TnxDataSetInternalState = (public disClosing, public disShutdownFromCursor, public dis
```

File

nxdb

Members

Members	Description

<code>disClosing</code>	internal use only.
<code>disShutdownFromCursor</code>	internal use only.
<code>disIndexInvalid</code>	internal use only.

Remarks

Dataset states are used internally by the dataset class. There's no need for developers to manipulate these.

See Also

[nxdb](#)

You are here: Symbol Reference > Structs, Records, Enums > [TnxDataSetInternalState Enumeration](#)

25.4.10 TnxDataSetOption Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataSetOption Enumeration

[nxdb](#) | Related Topics

Pascal

```
public TnxDataSetOption = (public dsoOptimisticLocks, public dsoAddKeyAsVariantField, public
```

File

[nxdb](#)

Members

Members	Description
<code>dsoOptimisticLocks</code>	Use optimistic record locks for edit.
<code>dsoAddKeyAsVariantField</code>	Adds a field to access the current key value as variant
<code>dsoRecNoSupport</code>	Allows use of RecNo if the table supports it

Remarks

Dataset options definition

Related Topics

[TnxDataset.Options](#)

See Also

[nxdb](#)

[Related Topics](#)

You are here: Symbol Reference > Structs, Records, Enums > [TnxDataSetOption Enumeration](#)

25.4.11 TnxEngineAction Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxEngineAction Enumeration

[nxsdServerEngine](#)

Pascal

```
public TnxEngineAction = (public eaCreate, public eaDestroy, public eaChildCreate, public eaCh
```

File

[nxsdServerEngine](#)**Members**

Members	Description
eaCreate	After
eaDestroy	Before
eaChildCreate	After
eaChildDestroy	Before
eaAliasAdd	Before After Alias, Path : string
eaAliasDelete	Before After Alias : string
eaAliasModify	Before After Alias, NewName, NewPath : string
eaAliasGetList	Before After List : TObjectList
eaAliasGetPath	Before After Alias : string Path : PString
eaCloseInactiveFolders	Before After
eaCloseInactiveTables	Before After
eaGetRegisteredClassList	Before After ClassListType : TnxClassListType List : TStrings -- database
eaTransactionStart	Before After FailSafe, SnapShot : Boolean
eaTransactionCommit	Before After
eaTransactionRollback	Before After
eaTableBuild	Before After OverWrite : Boolean TableName : string Dictionary : TnxDataDictionary
eaTablePack	Before TableName : string
eaTableBackup	Before TableName : string TargetDB : TnxAbstractDatabase
eaTableRebuildIndex	Before TableName : string IndexID : Integer
eaTableRestructure	Before TableName : string Dictionary : TnxDataDictionary FieldMap : TStrings
eaTableAddIndex	Before TableName : string IndexDesc : TnxIndexDescriptor IsNewDefault: Boolean
eaTableDropIndex	Before TableName : string IndexName : string IndexID : Integer
eaTableRename	Before After OldName : string NewName : string
eaTableRecover	Before TableName : string
eaTableDelete	Before After TableName : string
eaTableEmpty	Before After TableName : string
eaTableExists	Before After TableName : string Exists : PBoolean
eaTableGetDictionary	Before After TableName : string Dictionary : TnxDataDictionary
eaTableGetList	Before After List : TStrings
eaGetChangedTables	Before After Tables : TStrings

eaRecordGet	After LockType : TnxLockType Data : PnxByteArray RefNr : TnxInt64 -> only valid server side
eaRecordInsert	Before After LockType : TnxLockType Data : PnxByteArray RefNr : TnxInt64 -> only valid after & server side
eaRecordModify	Before After NewData : PnxByteArrayOldData : PnxByteArray -> nil@Before ReleaseLock : Boolean RefNr : TnxInt64 -> only valid server side
eaRecordDelete	Before After OldData : PnxByteArray RefNr : TnxInt64 -> only valid before & server side
eaStatementPrepare	Before After StatementType : TnxStatementType QueryText : WideString Stream : TStream
eaStatementExec	Before After Cursor : PnxAbstractCursor OpenMode : TnxOpenMode Stream : TStream
eaStatementSetParams	Before After Params : PnxSqlParamList (const!) Stream : TStream
eaStatementGetParams	Before After Params : PnxSqlParamList (const!) Stream : TStream

Remarks

Action types for Engine based actions. These types are fired and passed on to Extenders.

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxEngineAction Enumeration

25.4.12 TnxFieldType Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldType Enumeration

[nxsdTypes](#)

Pascal

```
public TnxFieldType = (public nxtBoolean, public nxtChar, public nxtWideChar, public nxtByte,
```

File

[nxsdTypes](#)

Members

Members	Description
nxtBoolean	..8-bit Boolean flag.
nxtChar	..8-bit character.
nxtWideChar	..16-bit character (UNICODE).
nxtByte	..8-bit unsigned Integer).
nxtWord16	..16-bit unsigned Integer).
nxtWord32	..32-bit unsigned Integer.
nxtInt8	..8-bit signed Integer.

nxtInt16	..16-bit signed Integer.
nxtInt32	..32-bit signed Integer.
nxtInt64	..64-bit signed Integer.
nxtAutoInc	..32-bit unsigned Integer; auto incrementing
nxtSingle	..IEEE single (4 bytes).
nxtDouble	..IEEE double (8 bytes).
nxtExtended	..IEEE extended (10 bytes).
nxtCurrency	..Delphi currency type (8 bytes, scaled Integer).
nxtDate	..date type (4 bytes).
nxtTime	..time type (4 bytes).
nxtDateTime	..date/time type (8 bytes).
nxtInterval	..date/time interval type (8 bytes).
nxtBlob	..variable length BLOB field - general binary data
nxtBlobMemo	..variable length BLOB field - text memo.
nxtBlobGraphic	..variable length BLOB field - graphics object
nxtByteArray	..array of bytes.
nxtShortString	..length byte string.
nxtNullString	..null-terminated string.
nxtWideString	..null-terminated string of wide chars.
nxtRecRev	..32-bit unsigned Integer; auto incrementing on every record update.
nxtGuid	..128-bit byte array.
nxtBCD	..identical to Delphi currency type (8 bytes, scaled Integer).
nxtBlobWideMemo	..variable length BLOB field - unicode memo.
nxtFmtBCD	..variable length BCD field

Remarks

NexusDB field types.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxFieldType Enumeration

25.4.13 TnxFieldType Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldType Enumeration

nxdb

Pascal

```
public TnxFieldType = (public ftSimple, public ftSqlWhere);
```

File

nxdb

Members**Members**

ftSimple

Description

Use simple expression filter

ftSqlWhere

Use SQL WHERE based filter

Remarks

Possible Filter types for TnxDataset.Filter

See Also

nxdb

You are here: Symbol Reference > Structs, Records, Enums > TnxFilterType Enumeration

25.4.14 TnxIndexPathPosition Enumeration

NexusDB V2 VCL Reference[Contents](#) | [Index](#)

TnxIndexPathPosition Enumeration

nxsdTypes

Pascal

```
public TnxIndexPathPosition = (public ippUnknown, public ippBof, public ippOnCrackBefore, public
```

File

nxsdTypes

Members**Members**

ippUnknown

Description

..unknown

ippBof

..before all keys

ippOnCrackBefore

..in between two keys

ippOnCrackAfter

..in between two keys

ippOnKey

..on a key

ippEof

..after all keys

Remarks

Possible positions in an index

See Also

nxsdTypes

You are here: Symbol Reference > Structs, Records, Enums > TnxIndexPathPosition Enumeration

25.4.15 TnxKeyBuffer Record

NexusDB V2 VCL Reference[Contents](#) | [Index](#)

TnxKeyBuffer Record

nxdb

Pascal

```
public TnxKeyBuffer = packed record
  public kbModified: Boolean;
  public kbExclusive: Boolean;
```

```

public kbFieldCount: Integer;
public kbPartialLen: Integer;
end;

```

File

[nxdb](#)

Members

Members	Description
kbModified	internal use only.
kbExclusive	internal use only.
kbFieldCount	internal use only.
kbPartialLen	internal use only.

Remarks

Used internally to address the different key buffers.

See Also

[nxdb](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxKeyBuffer Record

25.4.16 TnxKeyIndex Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxKeyIndex Enumeration

[nxdb](#)

Pascal

```
public TnxKeyIndex = (public kiLookup, public kiRangeStart, public kiRangeEnd, public kiCurRangeStart, public kiCurRangeEnd, public kiSave);
```

File

[nxdb](#)

Members

Members	Description
kiLookup	internal use only
kiRangeStart	internal use only
kiRangeEnd	internal use only
kiCurRangeStart	internal use only
kiCurRangeEnd	internal use only.
kiSave	internal use only

Remarks

Used internally to address the different key buffers.

See Also

[nxdb](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxKeyIndex Enumeration

25.4.17 TnxLockConflictType Enumeration

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxLockConflictType Enumeration

nxsdTypes

Pascal

```
public TnxLockConflictType = (public lctUnknown, public lctRecordLock, public lctReadLock, pu
```

File

nxsdTypes

Remarks

This is record TnxLockConflictType.

See Also

nxsdTypes

You are here: Symbol Reference > Structs, Records, Enums > TnxLockConflictType Enumeration

25.4.18 TnxLockPresent Enumeration

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxLockPresent Enumeration

nxsdTypes

Pascal

```
public TnxLockPresent = (public lpNotAtAll, public lpYesByUs, public lpYesByAnother);
```

File

nxsdTypes

Members

Members	Description
lpNotAtAll	..no, not at all.
lpYesByUs	..yes, and by current session.
lpYesByAnother	..yes, and by another session.

Remarks

Used to indicate whether a lock is present in a certain server object.

See Also

nxsdTypes

You are here: Symbol Reference > Structs, Records, Enums > TnxLockPresent Enumeration

25.4.19 TnxLockRequestType Enumeration

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxLockRequestType Enumeration

nxsdTypes

Pascal

```
public TnxLockRequestType = (public lrtRecordLock, public lrtReadLock, public lrtWriteLock);
```

File[nxsdTypes](#)**Remarks**

This is record TnxLockRequestType.

See Also[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxLockRequestType Enumeration

25.4.20 TnxLockResult Enumeration

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxLockResult Enumeration

[nxsdTypes](#)**Pascal**

```
public TnxLockResult = (public lrGrantedNew, public lrGrantedExisting, public lrConflictUnknown)
```

File[nxsdTypes](#)**Remarks**

This is record TnxLockResult.

See Also[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxLockResult Enumeration

25.4.21 TnxLockType Enumeration

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxLockType Enumeration

[nxsdTypes](#)**Pascal**

```
public TnxLockType = (public nxltNoLock, public nxltReadLock, public nxltWriteLock);
```

File[nxsdTypes](#)**Members**

Members	Description
nxltNoLock	..no lock at all.
nxltReadLock	..read lock (not for record locks).
nxltWriteLock	..write lock.

Remarks

Lock Types fro NexusDB.

See Also[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxLockType Enumeration

25.4.22 TnxLogPriority Enumeration

[NexusDB V2 VCL Reference](#)

TnxLogPriority Enumeration

[Contents | Index](#)

nxlComponent

Pascal

```
public TnxLogPriority = (public lpDebug, public lpInfo, public lpWarn, public lpError, public
```

File

[nxlComponent](#)

Remarks

This is record TnxLogPriority.

See Also

[nxlComponent](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxLogPriority Enumeration

25.4.23 TnxMessageHeaderFlag Enumeration

[NexusDB V2 VCL Reference](#)

TnxMessageHeaderFlag Enumeration

[Contents | Index](#)

nxlTransport

Pascal

```
public TnxMessageHeaderFlag = (public mhfErrorMessage, public mhfReserved1, public mhfReserved2,
```

File

[nxlTransport](#)

Members

Members	Description
mhfErrorMessage	the message header is followed by a string
mhfReserved1	reserver for future use
mhfReserved2	reserver for future use
mhfReserved3	reserver for future use
mhfReserved4	reserver for future use
mhfReserved5	reserver for future use
mhfReserved6	reserver for future use
mhfReserved7	reserver for future use

Remarks

the flags that are sent with every message

See Also

[nxlTransport](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxMessageHeaderFlag Enumeration

25.4.24 TnxNullBehaviour Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxNullBehaviour Enumeration

[nxsdDataDictionary](#)

Pascal

```
public TnxNullBehaviour = (public nbTop, public nbBottom, public nbAsAscend, public nbFilterNull)
```

File

[nxsdDataDictionary](#)

Members

Members	Description
nbTop	sort to top
nbBottom	sort to bottom
nbAsAscend	sort first if index is Ascending, last if Descending
nbFilterNull	dont include records that have NULL in a field in the index
nbFilterNonNull	only records with the NULL value in the field is included in the index

Remarks

Type to define the Null behaviour of a key

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxNullBehaviour Enumeration

25.4.25 TnxOpenMode Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxOpenMode Enumeration

[nxsdTypes](#)

Pascal

```
public TnxOpenMode = (public omReadOnly, public omWriteOnly, public omReadWrite);
```

File

[nxsdTypes](#)

Members

Members	Description
omReadOnly	..read only mode.
omWriteOnly	..write only mode.
omReadWrite	..read/write mode.

Remarks

Open modes for opening databases, tables.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxOpenMode Enumeration

25.4.26 TnxParamType Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxParamType Enumeration

[nxsdTypes](#)

Pascal

```
public TnxParamType = (public nxptUnknown, public nxptInput, public nxptOutput, public nxptInp
```

File

[nxsdTypes](#)

Members

Members	Description
nxptUnknown	unknown
nxptInput	Input only parameter
nxptOutput	Output only parameter
nxptInputOutput	Input/Output parameter
nxptResult	Parameter defines result

Remarks

Parameter Types

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxParamType Enumeration

25.4.27 TnxRecordCountOption Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxRecordCountOption Enumeration

[nxsdTypes](#)

Pascal

```
public TnxRecordCountOption = (public rcoTotalRecordCount, public rcoTotalKeyCount, public rco
```

File

[nxsdTypes](#)

Members

Members	Description
rcoTotalRecordCount	..total number of records in this table.
rcoTotalKeyCount	..total number of keys in the current index.
rcoRangedKeyCount	..number of keys in the current range.

rcoServerFilteredKeyCount	...number of keys in the current range that have records matching the current server-side filter.
rcoClientFilteredKeyCount	...number of keys in the current range that have records matching the current server- and client-side filter.

Remarks

Options for GetRecordCountEx.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxRecordCountOption Enumeration

25.4.28 TnxRecordGetBatchExOption Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxRecordGetBatchExOption Enumeration

[nxsdTypes](#)

Pascal

```
public TnxRecordGetBatchExOption = (public gboBlobs, public gboBookmarks, public gboBackwards)
```

File

[nxsdTypes](#)

Members

Members	Description
gboBlobs	handle blobs in batch operation.
gboBookmarks	copy bookmarks in batch operation.
gboBackwards	read backwards instead of forward

Remarks

Options for the batch mode.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxRecordGetBatchExOption Enumeration

25.4.29 TnxSearchKeyAction Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSearchKeyAction Enumeration

[nxsdTypes](#)

Pascal

```
public TnxSearchKeyAction = (public skaEqualFirst, public skaGreaterEqual, public skaGreater,
```

File

[nxsdTypes](#)

Members

Members	Description
---------	-------------

skaEqualFirst	..first key in index order that is equal.
skaGreaterEqual	..first key in index order that is equal or greater.
skaGreater	..first key in index order that is greater.
skaEqualLast	..last key in index order that is equal.
skaSmallerEqual	..last key in index order that is equal or smaller.
skaSmaller	..last key in index order that is smaller.

Remarks

Used for quick locating/marketing of keys in the SQL and filter engines.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxSearchKeyAction Enumeration

25.4.30 TnxServerModule Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxServerModule Enumeration

[nxServerComp](#)

Pascal

```
public TnxServerModule = (public nsmEventLog, public nsmSQLEngine, public nsmWinsockTransport,
```

File

[nxServerComp](#)

Members

Members	Description
nsmEventLog	Event Log.
nsmSQLEngine	SQL Engine.
nsmWinsockTransport	Winsock Transport.
nsmNamedPipetransport	Named Pipe Transport.

Remarks

supported Server modules.

See Also

[nxServerComp](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxServerModule Enumeration

25.4.31 TnxSetKeyOption Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSetKeyOption Enumeration

[nxdb](#)

Pascal

```
public TnxSetKeyOption = (public skoExclusive, public skoPartialMatch);
```

File

[nxdb](#)

Members

Members	Description
skoExclusive	exclusive full match
skoPartialMatch	partial match

Remarks

SetKey option

See Also

[nxdb](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxSetKeyOption Enumeration

25.4.32 TnxSettingType Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSettingType Enumeration

[nxllComponent](#)

Pascal

```
public TnxSettingType = (public nxstString, public nxstInteger, public nxstBoolean, public nxstStream, public nxstList, public nxstThreadPriority, public nxstSingleOption, public nxstMultiOption)
```

File

[nxllComponent](#)

Members

Members	Description
nxstString	string
nxstInteger	integer
nxstBoolean	Boolean
nxstStream	Stream
nxstList	Selection List
nxstThreadPriority	Thread Priority
nxstSingleOption	Radio Button
nxstMultiOption	Checkboxes

Remarks

Settings data type

See Also

[nxllComponent](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxSettingType Enumeration

25.4.33 TnxShareMode Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxShareMode Enumeration

[nxsdTypes](#)

Pascal

```
public TnxShareMode = (public smExclusive, public smShared, public smShareRead);
```

File

nxsdTypes

Members

Members	Description
smExclusive	..exclusive, no sharing.
smShared	..allows others to Read or Write.
smShareRead	..allows others to Read only.

Remarks

Share modes for opening databases, tables.

See Also

nxsdTypes

You are here: Symbol Reference > Structs, Records, Enums > TnxShareMode Enumeration

25.4.34 TnxSqlParamDesc Record

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxSqlParamDesc Record

nxsdTypes

Pascal

```
public TnxSqlParamDesc = record
  public piParamName: string;
  public piParamType: TnxParamType;
  public piDataType: TnxFieldType;
  public piData: TnxDynByteArray;
end;
```

File

nxsdTypes

Members

Members	Description
piParamName	..parameter name.
piParamType	..parameter type.
piDataType	..data type.
piData	..data.

Remarks

The following type defines the data needed for a SQL parameter.

See Also

nxsdTypes

You are here: Symbol Reference > Structs, Records, Enums > TnxSqlParamDesc Record

25.4.35 TnxState Enumeration

NexusDB V2 VCL Reference
TnxState Enumeration

[Contents](#) | [Index](#)

nxllComponent

Pascal

```
public TnxState = (public nxsInactive, public nxsUnsupported, public nxsStopped, public nxsStarted)
```

File

[nxllComponent](#)

Members

Members

Members	Description
nxsInactive	Inactive
nxsUnsupported	Unsupported state - error
nxsStopped	Stopped
nxsStarted	Started

Remarks

Possible states for TnxStateComponent

See Also

[nxllComponent](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxState Enumeration

25.4.36 TnxStatementExecDirectPhase Enumeration

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxStatementExecDirectPhase Enumeration

nxsdServerEngine

Pascal

```
public TnxStatementExecDirectPhase = (public sedAlloc, public sedPrepare, public sedSetParams,
```

File

[nxsdServerEngine](#)

Members

Members

Members	Description
sedAlloc	Allocating resources
sedPrepare	Preparing
sedSetParams	Setting Parameters
sedExec	Executing
sedGetParams	Getting Parameters
sedFree	Destroying

Remarks

The state of the currently executed Statement

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxStatementExecDirectPhase Enumeration

25.4.37 TnxStatementType Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStatementType Enumeration

nxsdTypes

Pascal

```
public TnxStatementType = (public stQuery, public stStoredProcedure);
```

File

nxsdTypes

Members

Members	Description
stQuery	normal Query
stStoredProcedure	Stored Procedure

Remarks

Type of a Statement

See Also

nxsdTypes

You are here: Symbol Reference > Structs, Records, Enums > TnxStatementType Enumeration

25.4.38 TnxStateTransition Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxStateTransition Enumeration

nxllComponent

Pascal

```
public TnxStateTransition = (public nxstNone, public nxstInitializing, public nxstDeactivating,
```

File

nxllComponent

Members

Members	Description
nxstNone	None
nxstInitializing	Component initializing
nxstDeactivating	Component deactivating
nxstStarting	Component starting
nxstStopping	Component stopping
nxstInvalid	Invalid state

Remarks

Possible state transitions for TnxStateComponent

See Also[nxllComponent](#)*You are here:* Symbol Reference > Structs, Records, Enums > TnxStateTransition Enumeration

25.4.39 TnxTableScope Enumeration

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxTableScope Enumeration**[nxsdServerEngine](#)**Pascal**

```
public TnxTableScope = (public tsPersistent, public tsTempGlobal, public tsTempDatabase, public
```

File[nxsdServerEngine](#)**Members**

Members	Description
tsPersistent	Persistent Table
tsTempGlobal	Global Temporary Table
tsTempDatabase	Database based Temporary Table
tsTempUnnamed	Unnamed Table - not visible to clients

Remarks

Scope of a table

See Also[nxsdServerEngine](#)*You are here:* Symbol Reference > Structs, Records, Enums > TnxTableScope Enumeration

25.4.40 TnxTaskStatus Record

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxTaskStatus Record**[nxsdTypes](#)**Pascal**

```
public TnxTaskStatus = packed record
  public tsStartTime: TnxWord32;
  public tsSnapshotTime: TnxWord32;
  public tsTotalRecs: TnxWord32;
  public tsRecsRead: TnxWord32;
  public tsRecsWritten: TnxWord32;
  public tsErrorCode: TnxResult;
  public tsPercentDone: Byte;
  public tsFinished: Boolean;
end;
```

File[nxsdTypes](#)**Members**

Members	Description
tsStartTime	..start time (tick count from server).
tsSnapshotTime	..snapshot time (tick count from server).
tsTotalRecs	..total count of records to read.
tsRecsRead	..count of records read.
tsRecsWritten	..count of records written.
tsErrorCode	..error result for process.
tsPercentDone	..RecsRead*100/TotalRecs.
tsFinished	..process has finished.

Remarks

Task operation status info.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxTaskStatus Record

25.4.41 TnxTransportLogOption Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransportLogOption Enumeration

[nxllTransport](#)

Pascal

```
public TnxTransportLogOption = (public nxtpLogErrors, public nxtpLogRequests, public nxtpLogReplies)
```

File

[nxllTransport](#)

Members

Members	Description
nxtpLogErrors	Log errors only.
nxtpLogRequests	Log all requests. Note that this has a major impact on performance.
nxtpLogReplies	Log all replies Note that this has a major impact on performance.

Remarks

Log options for the transport.

See Also

[nxllTransport](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxTransportLogOption Enumeration

25.4.42 TnxTransportMode Enumeration

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxTransportMode Enumeration

[nxllTransport](#)

Pascal

```
public TnxTransportMode = (public nxmSend, public nxmListen);
```

File

[nxllTransport](#)

Members

Members	Description
nxmSend	.. the transport is set up for sending.
nxmListen	.. the transport is set up for receiving.

Remarks

The mode of the transport.

See Also

[nxllTransport](#)

You are here: Symbol Reference > Structs, Records, Enums > TnxTransportMode Enumeration

25.5 Types

25.5.1 EnxComponentExceptionClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

EnxComponentExceptionClass Type

[nxllComponent](#)

Pascal

```
EnxComponentExceptionClass = class of EnxComponentException;
```

File

[nxllComponent](#)

Remarks

Class definition for Component Exceptions

See Also

[nxllComponent](#)

You are here: Symbol Reference > Types > EnxComponentExceptionClass Type

25.5.2 PnxBaseHeader Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

PnxBaseHeader Type

[nxllTransport](#)

Pascal

```
PnxBaseHeader = ^TnxBaseHeader;
```

File

[nxllTransport](#)

Remarks

The base header for a communication messages.

See Also

nxIITransport

You are here: Symbol Reference > Types > PnxBaseHeader Type

25.5.3 PnxBaseSession Type

*NexusDB V2 VCL Reference***PnxBaseSession Type**[Contents](#) | [Index](#)

nxdb

Pascal

```
PnxBaseSession = ^TnxBaseSession;
```

File

nxdb

Remarks

This is type PnxBaseSession.

See Also

nxdb

You are here: Symbol Reference > Types > PnxBaseSession Type

25.5.4 PnxBookmark Type

*NexusDB V2 VCL Reference***PnxBookmark Type**[Contents](#) | [Index](#)

nxsdTypes

Pascal

```
PnxBookmark = ^TnxBookmark;
```

File

nxsdTypes

Remarks

Pointer to bookmark data

See Also

nxsdTypes

You are here: Symbol Reference > Types > PnxBookmark Type

25.5.5 PnxDataMessage Type

*NexusDB V2 VCL Reference***PnxDataMessage Type**[Contents](#) | [Index](#)

nxIITransport

Pascal

```
PnxDataMessage = ^TnxDataMessage;
```

File

[nxllTransport](#)

Remarks

the structure for storing one data message part.

See Also

[nxllTransport](#)

You are here: Symbol Reference > Types > PnxDataMessage Type

25.5.6 PnxKeyBuffer Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

PnxKeyBuffer Type

[nxdb](#)

Pascal

```
PnxKeyBuffer = ^TnxKeyBuffer;
```

File

[nxdb](#)

Remarks

Used internally to address the different key buffers.

See Also

[nxdb](#)

You are here: Symbol Reference > Types > PnxKeyBuffer Type

25.5.7 PnxLockResult Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

PnxLockResult Type

[nxsdTypes](#)

Pascal

```
PnxLockResult = ^TnxLockResult;
```

File

[nxsdTypes](#)

Remarks

This is type PnxLockResult.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Types > PnxLockResult Type

25.5.8 PnxSqlParamDesc Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

PnxSqlParamDesc Type

[nxsdTypes](#)

Pascal

```
PnxSqlParamDesc = ^TnxSqlParamDesc;
```

File

nxsdTypes

Remarks

Pointer to TnxSqlParamDesc.

See Also

nxsdTypes

You are here: Symbol Reference > Types > PnxSqlParamDesc Type

25.5.9 PnxSqlParamList Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

PnxSqlParamList Type

nxsdTypes

Pascal

```
PnxSqlParamList = ^TnxSqlParamList;
```

File

nxsdTypes

Remarks

This is type PnxSqlParamList.

See Also

nxsdTypes

You are here: Symbol Reference > Types > PnxSqlParamList Type

25.5.10 PnxTaskStatus Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

PnxTaskStatus Type

nxsdTypes

Pascal

```
PnxTaskStatus = ^TnxTaskStatus;
```

File

nxsdTypes

Remarks

Task operation status info.

See Also

nxsdTypes

You are here: Symbol Reference > Types > PnxTaskStatus Type

25.5.11 TFieldDefClass Type

NexusDB V2 VCL Reference
TFieldDefClass Type

[Contents](#) | [Index](#)

nxdb

Pascal

```
TFieldDefClass = class of TFieldDef;
```

File

nxdb

Remarks

A Class of TFieldDef (needed for abstraction purposes only.)

See Also

nxdb

You are here: Symbol Reference > Types > TFieldDefClass Type

25.5.12 TFormatSettings Type

NexusDB V2 VCL Reference
TFormatSettings Type

[Contents](#) | [Index](#)

nxsdTypes

Pascal

```
TFormatSettings = 0..0;
```

File

nxsdTypes

Remarks

This is type TFormatSettings.

See Also

nxsdTypes

You are here: Symbol Reference > Types > TFormatSettings Type

25.5.13 TNotifyErrorEvent Type

NexusDB V2 VCL Reference
TNotifyErrorEvent Type

[Contents](#) | [Index](#)

nxBaseServerComp

Pascal

```
TNotifyErrorEvent = procedure (Sender: TObject; const anError: String) of object;
```

File

nxBaseServerComp

Remarks

The type used for error the error notification event.

See Also

nxBaseServerComp

You are here: Symbol Reference > Types > TNotifyErrorEvent Type

25.5.14 TnxAbstractServerObjectClass Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractServerObjectClass Type

nxsdServerEngine

Pascal

```
TnxAbstractServerObjectClass = class of TnxAbstractServerObject;
```

File

nxsdServerEngine

Remarks

Class definition for Server Objects.

See Also

nxsdServerEngine

You are here: Symbol Reference > Types > TnxAbstractServerObjectClass Type

25.5.15 TnxAbstractSessionClass Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAbstractSessionClass Type

nxsdServerEngine

Pascal

```
TnxAbstractSessionClass = class of TnxAbstractSession;
```

File

nxsdServerEngine

Remarks

This is type TnxAbstractSessionClass.

See Also

nxsdServerEngine

You are here: Symbol Reference > Types > TnxAbstractSessionClass Type

25.5.16 TnxAddSessionEvent Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxAddSessionEvent Type

nxllTransport

Pascal

```
TnxAddSessionEvent = function (aSender : TnxBaseTransport; const aUserName : string; const aPa
```

File

nxllTransport

Parameters

Parameters	Description
aSender	the transport that received the event
aUserName	the user name that should be used for authentication
aPassword	the password that should be used for authentication
aTimeOut	the timeout in msec
aClientVersion	the version of the client who wants to connect
aSessionID	the returned SessionID
aServerVersion	the returned Server version

Remarks

Definition for the OnAddSession event.

See Also

[nxlITransport](#)

You are here: Symbol Reference > Types > TnxAddSessionEvent Type

25.5.17 TnxApplyAtSet Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxApplyAtSet Type

[nxsdDataDictionary](#)

Pascal

```
TnxApplyAtSet = set of TnxApplyAt;
```

File

[nxsdDataDictionary](#)

Remarks

Set of TnxApplyAt

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxApplyAtSet Type

25.5.18 TnxAutoIncStepRange Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxAutoIncStepRange Type

[nxsdDataDictionary](#)

Pascal

```
TnxAutoIncStepRange = 1..High(TnxWord32);
```

File

[nxsdDataDictionary](#)

Remarks

This is type TnxAutoIncStepRange.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxAutoIncStepRange Type

25.5.19 TnxBaseAutoIncDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseAutoIncDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseAutoIncDescriptorClass = class of TnxBaseAutoIncDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for AutoInc Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseAutoIncDescriptorClass Type

25.5.20 TnxBaseBlobDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseBlobDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseBlobDescriptorClass = class of TnxBaseBlobDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Blob Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseBlobDescriptorClass Type

25.5.21 TnxBaseBlockHeapDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseBlockHeapDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseBlockHeapDescriptorClass = class of TnxBaseBlockHeapDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Block Heap Descriptors

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseBlockHeapDescriptorClass Type

25.5.22 TnxBaseCustomDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseCustomDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseCustomDescriptorClass = class of TnxBaseCustomDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Custom Descriptor

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseCustomDescriptorClass Type

25.5.23 TnxBaseDefaultValueDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseDefaultValueDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseDefaultValueDescriptorClass = class of TnxBaseDefaultValueDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for TnxBaseDefaultValueDescriptor.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseDefaultValueDescriptorClass Type

25.5.24 TnxBaseEngineExtenderClass Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseEngineExtenderClass Type

nxsdServerEngine

Pascal

```
TnxBaseEngineExtenderClass = class of TnxBaseEngineExtender;
```

File

[nxsdServerEngine](#)

Remarks

class definition for Server Extenders

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Types > TnxBaseEngineExtenderClass Type

25.5.25 TnxBaseFieldsValidationDescriptorClass Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldsValidationDescriptorClass Type

nxsdDataDictionary

Pascal

```
TnxBaseFieldsValidationDescriptorClass = class of TnxBaseFieldsValidationDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for TnxBaseFieldsValidationDescriptor.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseFieldsValidationDescriptorClass Type

25.5.26 TnxBaseFieldValidationDescriptorClass Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseFieldValidationDescriptorClass Type

nxsdDataDictionary

Pascal

```
TnxBaseFieldValidationDescriptorClass = class of TnxBaseFieldValidationDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for TnxBaseFieldValidationDescriptor.

See Also

[nxsdDataDictionary](#)*You are here:* Symbol Reference > Types > TnxBaseFieldValidationDescriptorClass Type

25.5.27 TnxBaseHeapDescriptorClass Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBaseHeapDescriptorClass Type

[nxsdDataDictionary](#)**Pascal**

```
TnxBaseHeapDescriptorClass = class of TnxBaseHeapDescriptor;
```

File[nxsdDataDictionary](#)**Remarks**

Class definition for Heap Descriptors.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Types > TnxBaseHeapDescriptorClass Type

25.5.28 TnxBaseIndicesDescriptorClass Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBaseIndicesDescriptorClass Type

[nxsdDataDictionary](#)**Pascal**

```
TnxBaseIndicesDescriptorClass = class of TnxBaseIndicesDescriptor;
```

File[nxsdDataDictionary](#)**Remarks**

Class definition for the base class of Indices Descriptors.

See Also[nxsdDataDictionary](#)*You are here:* Symbol Reference > Types > TnxBaseIndicesDescriptorClass Type

25.5.29 TnxBasePooledTransportClass Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxBasePooledTransportClass Type

[nxptBasePooledTransport](#)**Pascal**

```
TnxBasePooledTransportClass = class of TnxBasePooledTransport;
```

File[nxptBasePooledTransport](#)

Remarks

This is type TnxBasePooledTransportClass.

See Also

[nxptBasePooledTransport](#)

You are here: Symbol Reference > Types > TnxBasePooledTransportClass Type

25.5.30 TnxBaseRecordCompressionDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordCompressionDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseRecordCompressionDescriptorClass = class of TnxBaseRecordCompressionDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Compression Descriptors

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseRecordCompressionDescriptorClass Type

25.5.31 TnxBaseRecordDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseRecordDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseRecordDescriptorClass = class of TnxBaseRecordDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Record Descriptors

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseRecordDescriptorClass Type

25.5.32 TnxBaseStreamDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseStreamDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseStreamDescriptorClass = class of TnxBaseStreamDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Stream Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseStreamDescriptorClass Type

25.5.33 TnxBaseTableDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTableDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxBaseTableDescriptorClass = class of TnxBaseTableDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for base Table Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxBaseTableDescriptorClass Type

25.5.34 TnxBaseTransportClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxBaseTransportClass Type

[nxllTransport](#)

Pascal

```
TnxBaseTransportClass = class of TnxBaseTransport;
```

File

[nxllTransport](#)

Remarks

Class definition for transports.

See Also

[nxllTransport](#)

You are here: Symbol Reference > Types > TnxBaseTransportClass Type

25.5.35 TnxBaseUpdateHandlerClass Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBaseUpdateHandlerClass Type

nxdb

Pascal

```
TnxBaseUpdateHandlerClass = class of TnxBaseUpdateHandler;
```

File

nxdb

Remarks

Class defintion for TnxBaseUpdateHandler.

See Also

nxdb

You are here: Symbol Reference > Types > TnxBaseUpdateHandlerClass Type

25.5.36 TnxBlockSignature Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxBlockSignature Type

nxsdTypes

Pascal

```
TnxBlockSignature = array [0..3] of Char;
```

File

nxsdTypes

Remarks

Type for the table block signature. The first 4 characters hold engine and Version information.

See Also

nxsdTypes

You are here: Symbol Reference > Types > TnxBlockSignature Type

25.5.37 TnxCachedDataSetOptions Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxCachedDataSetOptions Type

nxdb

Pascal

```
TnxCachedDataSetOptions = set of TnxCachedDataSetOption;
```

File

nxdb

Remarks

Set of TnxCachedDataSetOption

See Also

nxdb

You are here: Symbol Reference > Types > TnxCachedDataSetOptions Type

25.5.38 TnxChooseServerEvent Type

NexusDB V2 VCL Reference[Contents](#) | [Index](#)

TnxChooseServerEvent Type

nxlITransport

Pascal

```
TnxChooseServerEvent = procedure (aSender : TnxBaseTransport; aServerNames : TStrings; var aSe
```

File

nxlITransport

Parameters

Parameters	Description
aSender	the transport that received the event
aServerNames	the names of available servers
aServerName	the name of the (pre)selected server
aResult	set to true to force connect to aServerName

Remarks

The definition for the OnChooseServer event

See Also

nxlITransport

You are here: Symbol Reference > Types > TnxChooseServerEvent Type

25.5.39 TnxConnectionLostEvent Type

NexusDB V2 VCL Reference[Contents](#) | [Index](#)

TnxConnectionLostEvent Type

nxlITransport

Pascal

```
TnxConnectionLostEvent = procedure (aSender : TnxBaseTransport; aSessionID : TnxSessionID) of
```

File

nxlITransport

Parameters

Parameters	Description
aSender	the transport that received the event
aSessionID	the session id

Remarks

The definition for the OnConnectionLost event.

See Also

nxlITransport

You are here: Symbol Reference > Types > TnxConnectionLostEvent Type

25.5.40 TnxCreateTableEvent Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCreateTableEvent Type

nxdb

Pascal

```
TnxCreateTableEvent = procedure (aSender : TnxDataSet; aDictionary : TnxDataDictionary) of ob;
```

File

nxdb

Remarks

Method defintion for CreateTable callback event.

See Also

nxdb

You are here: Symbol Reference > Types > TnxCreateTableEvent Type

25.5.41 TnxCursorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCursorClass Type

nxdb

Pascal

```
TnxCursorClass = class of TnxCursor;
```

File

nxdb

Remarks

Class defintion for Cursors.

See Also

nxdb

You are here: Symbol Reference > Types > TnxCursorClass Type

25.5.42 TnxCursorID Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCursorID Type

nxsdTypes

Pascal

```
TnxCursorID = type TnxBaseID;
```

File

nxsdTypes

Remarks

Strong type for Cursor IDs

▀ **See Also**

[nxsdTypes](#)

You are here: Symbol Reference > Types > TnxCursorID Type

25.5.43 TnxCustomDescsDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxCustomDescsDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxCustomDescsDescriptorClass = class of TnxCustomDescsDescriptor;
```

▀ **File**

[nxsdDataDictionary](#)

▀ **Remarks**

Class definition for Custom Descriptors.

▀ **See Also**

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxCustomDescsDescriptorClass Type

25.5.44 TnxDatabaseID Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDatabaseID Type

[nxsdTypes](#)

Pascal

```
TnxDatabaseID = type TnxBaseID;
```

▀ **File**

[nxsdTypes](#)

▀ **Remarks**

Strong type for Database IDs

▀ **See Also**

[nxsdTypes](#)

You are here: Symbol Reference > Types > TnxDatabaseID Type

25.5.45 TnxDataSetInternalStates Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDataSetInternalStates Type

[nxdb](#)

Pascal

```
TnxDataSetInternalStates = set of TnxDataSetInternalState;
```

File

nxdb

Remarks

Dataset states are used internally by the dataset class. There's no need for developers to manipulate these.

See Also

nxdb

You are here: Symbol Reference > Types > TnxDataSetInternalStates Type

25.5.46 TnxDataSetOptions Type

NexusDB V2 VCL Reference[Contents](#) | [Index](#)

TnxDataSetOptions Type

nxdb

Pascal

```
TnxDataSetOptions = set of TnxDataSetOption;
```

File

nxdb

Remarks

Blob Cache Options.

See Also

nxdb

You are here: Symbol Reference > Types > TnxDataSetOptions Type

25.5.47 TnxDictionaryItemClass Type

NexusDB V2 VCL Reference[Contents](#) | [Index](#)

TnxDictionaryItemClass Type

nxsdDataDictionary

Pascal

```
TnxDictionaryItemClass = class of TnxDictionaryItem;
```

File

nxsdDataDictionary

Remarks

Class definition for dictionary items.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Types > TnxDictionaryItemClass Type

25.5.48 TnxDirectKeySetFieldsMethod Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxDirectKeySetFieldsMethod Type

nxdb

Pascal

```
TnxDirectKeySetFieldsMethod = procedure (const aValues1 : array of const; const aValues2 : array of const);
```

File

nxdb

Remarks

Method definition for DirectKeySetFields callback method.

See Also

nxdb

You are here: Symbol Reference > Types > TnxDirectKeySetFieldsMethod Type

25.5.49 TnxEngineActions Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxEngineActions Type

nxsdServerEngine

Pascal

```
TnxEngineActions = set of TnxEngineAction;
```

File

nxsdServerEngine

Remarks

Set of Engine Actions

See Also

nxsdServerEngine

You are here: Symbol Reference > Types > TnxEngineActions Type

25.5.50 TnxFieldDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldDescriptorClass Type

nxsdDataDictionary

Pascal

```
TnxFieldDescriptorClass = class of TnxFieldDescriptor;
```

File

nxsdDataDictionary

Remarks

Class definition for Field Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxFieldDescriptorClass Type

25.5.51 TnxFieldsDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxFieldsDescriptorClass = class of TnxFieldsDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Fields Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxFieldsDescriptorClass Type

25.5.52 TnxFieldsValidationsDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldsValidationsDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxFieldsValidationsDescriptorClass = class of TnxFieldsValidationsDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

This is type TnxFieldsValidationsDescriptorClass.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxFieldsValidationsDescriptorClass Type

25.5.53 TnxFieldType Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldType Type

[nxsdTypes](#)

Pascal

```
TnxFieldType = set of TnxFieldType;
```

File

[nxsdTypes](#)

Remarks

NexusDB field types.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Types > TnxFieldTypes Type

25.5.54 TnxFieldValidationsDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFieldValidationsDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxFieldValidationsDescriptorClass = class of TnxFieldValidationsDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Validation Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxFieldValidationsDescriptorClass Type

25.5.55 TnxFileDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFileDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxFileDescriptorClass = class of TnxFileDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for File Descriptors

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxFileDescriptorClass Type

25.5.56 TnxFilesDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilesDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxFilesDescriptorClass = class of TnxFilesDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Files Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxFilesDescriptorClass Type

25.5.57 TnxFilterID Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFilterID Type

[nxsdTypes](#)

Pascal

```
TnxFilterID = type TnxBaseID;
```

File

[nxsdTypes](#)

Remarks

Strong type for Filter IDs.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Types > TnxFilterID Type

25.5.58 TnxFindServersEvent Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxFindServersEvent Type

[nxllTransport](#)

Pascal

```
TnxFindServersEvent = procedure (aSender : TnxBaseTransport; aStarting : Boolean) of object;
```

File

[nxllTransport](#)

Parameters

Parameters	Description
aSender	the transport that received the event
aStarting	set to true if the event was called before starting to search otherwise its false

Remarks

The definition for the OnFindServers event

See Also

[nxllTransport](#)

You are here: Symbol Reference > Types > TnxFindServersEvent Type

25.5.59 TnxGrowSize Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxGrowSize Type

nxsdTypes

Pascal

```
TnxGrowSize = 1..High(Integer);
```

File

nxsdTypes

Remarks

Type for the table grow size.

See Also

nxsdTypes

You are here: Symbol Reference > Types > TnxGrowSize Type

25.5.60 TnxHeapBlobDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxHeapBlobDescriptorClass Type

nxsdDataDictionary

Pascal

```
TnxHeapBlobDescriptorClass = class of TnxHeapBlobDescriptor;
```

File

nxsdDataDictionary

Remarks

Class definition for Heap Blob Descriptors.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Types > TnxHeapBlobDescriptorClass Type

25.5.61 TnxHeapRecordDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxHeapRecordDescriptorClass Type

nxsdDataDictionary

Pascal

```
TnxHeapRecordDescriptorClass = class of TnxHeapRecordDescriptor;
```

File

nxsdDataDictionary

Remarks

Class definition for Heap Record Descriptors.

▀ **See Also**

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxHeapRecordDescriptorClass Type

25.5.62 TnxIndexDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIndexDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxIndexDescriptorClass = class of TnxIndexDescriptor;
```

▀ **File**

[nxsdDataDictionary](#)

▀ **Remarks**

Class definition for Index descriptors.

▀ **See Also**

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxIndexDescriptorClass Type

25.5.63 TnxIterationListClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxIterationListClass Type

[nxllComponent](#)

Pascal

```
TnxIterationListClass = class of TnxIterationList;
```

▀ **File**

[nxllComponent](#)

▀ **Remarks**

Class definition for TnxIterationList

▀ **See Also**

[nxllComponent](#)

You are here: Symbol Reference > Types > TnxIterationListClass Type

25.5.64 TnxKeyDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxKeyDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxKeyDescriptorClass = class of TnxBaseKeyDescriptor;
```

File[nxsdDataDictionary](#)**Remarks**

Class definition for Key descriptors.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxKeyDescriptorClass Type

25.5.65 TnxKeyFieldDescriptorClass Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxKeyFieldDescriptorClass Type

[nxsdDataDictionary](#)**Pascal**

```
TnxKeyFieldDescriptorClass = class of TnxKeyFieldDescriptor;
```

File[nxsdDataDictionary](#)**Remarks**

Class definition for Key Field descriptors.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxKeyFieldDescriptorClass Type

25.5.66 TnxKeyFilterID Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxKeyFilterID Type

[nxsdTypes](#)**Pascal**

```
TnxKeyFilterID = type TnxBaseID;
```

File[nxsdTypes](#)**Remarks**

Strong type for KeyFilter IDs.

See Also[nxsdTypes](#)

You are here: Symbol Reference > Types > TnxKeyFilterID Type

25.5.67 TnxLocaleDescriptorClass Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxLocaleDescriptorClass Type

[nxsdDataDictionary](#)**Pascal**

```
TnxLocaleDescriptorClass = class of TnxLocaleDescriptor;
```

File[nxsdDataDictionary](#)**Remarks**

Class definition for TnxLocaleDescriptor.

See Also[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxLocaleDescriptorClass Type

25.5.68 TnxLoginCallback Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxLoginCallback Type

[nxBaseServerComp](#)**Pascal**

```
TnxLoginCallback = procedure (aSender : TnxBaseServer; var aUserName : string; var aPassword : string);
```

File[nxBaseServerComp](#)**Parameters**

Parameters	Description
aSender	the component that triggered the event
aUserName	the user name for logging on to the server, might be reinitialized.
aPassword	the fitting password

Remarks

This is the call back type for TnxBaseServer.OnShowLogin.

See Also[nxBaseServerComp](#)

You are here: Symbol Reference > Types > TnxLoginCallback Type

25.5.69 TnxLoginEvent Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)

TnxLoginEvent Type

[nxdb](#)**Pascal**

```
TnxLoginEvent = procedure (aSender : TnxBaseSession; var aUserName : string; var aPassword : string);
```

File[nxdb](#)

Remarks

An event to get a user name and password for login purposes. Called from TnxBaseSession.

See Also

[nxdb](#)

You are here: Symbol Reference > Types > TnxLoginEvent Type

25.5.70 TnxLogPriorities Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxLogPriorities Type

[nxllComponent](#)

Pascal

```
TnxLogPriorities = set of TnxLogPriority;
```

File

[nxllComponent](#)

Remarks

This is type TnxLogPriorities.

See Also

[nxllComponent](#)

You are here: Symbol Reference > Types > TnxLogPriorities Type

25.5.71 TnxMainIndicesDescriptorClass Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMainIndicesDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxMainIndicesDescriptorClass = class of TnxMainIndicesDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for the class of Main-Index Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxMainIndicesDescriptorClass Type

25.5.72 TnxMappingMethod Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxMappingMethod Type

[nxsdServerEngine](#)

Pascal

```
TnxMappingMethod = function (const aSource; var aTarget) : TnxResult of object;
```

File

[nxsdServerEngine](#)

Remarks

Abstract Prototype for MappingMethods

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Types > TnxMappingMethod Type

25.5.73 TnxMessageHeaderFlags Type

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxMessageHeaderFlags Type

[nxllTransport](#)

Pascal

```
TnxMessageHeaderFlags = set of TnxMessageHeaderFlag;
```

File

[nxllTransport](#)

Remarks

set of Message flags

See Also

[nxllTransport](#)

You are here: Symbol Reference > Types > TnxMessageHeaderFlags Type

25.5.74 TnxNestedTableDescriptorClass Type

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxNestedTableDescriptorClass Type

[nxsdDataDictionary](#)

Pascal

```
TnxNestedTableDescriptorClass = class of TnxNestedTableDescriptor;
```

File

[nxsdDataDictionary](#)

Remarks

Class definition for Nested(Sub-) table Descriptors.

See Also

[nxsdDataDictionary](#)

You are here: Symbol Reference > Types > TnxNestedTableDescriptorClass Type

25.5.75 TnxNetIdleEvent Type

NexusDB V2 VCL Reference
TnxNetIdleEvent Type

[Contents](#) | [Index](#)

nxptBasePooledTransport

Pascal

```
TnxNetIdleEvent = procedure (aSender: TObject; aWaitEvent: THandle) of object;
```

File

nxptBasePooledTransport

Remarks

This is type TnxNetIdleEvent.

See Also

nxptBasePooledTransport

You are here: Symbol Reference > Types > TnxNetIdleEvent Type

25.5.76 TnxOnAcceptConnection Type

NexusDB V2 VCL Reference
TnxOnAcceptConnection Type

[Contents](#) | [Index](#)

nxtwWinsockTransport

Pascal

```
TnxOnAcceptConnection = procedure (aSender : TnxCustomConnectionFilter; aTransport : TnxWinso
```

File

nxtwWinsockTransport

Remarks

The type of event method used in TnxCustomConnectionFilter.

See Also

nxtwWinsockTransport

You are here: Symbol Reference > Types > TnxOnAcceptConnection Type

25.5.77 TnxPersistentClass Type

NexusDB V2 VCL Reference
TnxPersistentClass Type

[Contents](#) | [Index](#)

nxlComponent

Pascal

```
TnxPersistentClass = class of TnxPersistent;
```

File

nxlComponent

Remarks

Class defintion for TnxPersistent.

See Also

nxlComponent

You are here: Symbol Reference > Types > TnxPersistentClass Type

25.5.78 TnxRecordGetBatchExOptions Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxRecordGetBatchExOptions Type

nxsdTypes

Pascal

```
TnxRecordGetBatchExOptions = set of TnxRecordGetBatchExOption;
```

File

nxsdTypes

Remarks

Options for the batch mode.

See Also

nxsdTypes

You are here: Symbol Reference > Types > TnxRecordGetBatchExOptions Type

25.5.79 TnxRegisterableComponentClass Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxRegisterableComponentClass Type

nxlComponent

Pascal

```
TnxRegisterableComponentClass = class of TnxRegisterableComponent;
```

File

nxlComponent

Remarks

Class definition for TnxRegisterableComponent.

See Also

nxlComponent

You are here: Symbol Reference > Types > TnxRegisterableComponentClass Type

25.5.80 TnxRemoveSessionEvent Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxRemoveSessionEvent Type

nxlTransport

Pascal

```
TnxRemoveSessionEvent = function (aSender : TnxBaseTransport; aSessionID : TnxSessionID) : TnxSessionID;
```

File

nxlTransport

Parameters**Parameters**

aSender

aSessionID

Description

the transport that received the event

the session id

Remarks

The definition for the OnRemoveSession event

See Also[nxllTransport](#)

You are here: Symbol Reference > Types > TnxRemoveSessionEvent Type

25.5.81 TnxReplyCallback Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxReplyCallback Type**[nxllTransport](#)**Pascal**

```
TnxReplyCallback = procedure (aMsgID : TnxMsgID; aErrorCode : TnxResult; aReplyData : Pointer;
```

File[nxllTransport](#)**Parameters****Parameters**

aMsgId

aErrorCode

aReplyData

aReplyDataLen

aReplyCookie

Description

the Message ID that triggered the callback

the error code for the message

the data returned to the request

the length of aReplyData

additional information transferred

Remarks

Definition for the callback functions used by the transports.

See Also[nxllTransport](#)

You are here: Symbol Reference > Types > TnxReplyCallback Type

25.5.82 TnxServerFilterTimeoutEvent Type

[NexusDB V2 VCL Reference](#)[Contents | Index](#)**TnxServerFilterTimeoutEvent Type**[nxdb](#) | Related Topics**Pascal**

```
TnxServerFilterTimeoutEvent = procedure (Sender : TnxDataset; var Cancel : Boolean) of object;
```

File[nxdb](#)

Remarks

This is an event handler definition for Filter timeouts. It gives the programmer a chance to commence with a filtering even if the filter timeout is reached.

Related Topics

[TnxDataset.FilterTimeout](#)

See Also

[nxdb](#)
[Related Topics](#)

You are here: Symbol Reference > Types > TnxServerFilterTimeoutEvent Type

25.5.83 TnxServerModules Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxServerModules Type

[nxServerComp](#)

Pascal

```
TnxServerModules = set of TnxServerModule;
```

File

[nxServerComp](#)

Remarks

Set of supported Server Modules.

See Also

[nxServerComp](#)

You are here: Symbol Reference > Types > TnxServerModules Type

25.5.84 TnxSetKeyOptions Type

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

TnxSetKeyOptions Type

[nxdb](#)

Pascal

```
TnxSetKeyOptions = set of TnxSetKeyOption;
```

File

[nxdb](#)

Remarks

Set of SetKey options.

See Also

[nxdb](#)

You are here: Symbol Reference > Types > TnxSetKeyOptions Type

25.5.85 TnxSqlParamList Type

NexusDB V2 VCL Reference
TnxSqlParamList Type

[Contents](#) | [Index](#)

nxsdTypes

Pascal

```
TnxSqlParamList = array of TnxSqlParamDesc;
```

File

nxsdTypes

Remarks

SQL Parameters.

See Also

nxsdTypes

You are here: Symbol Reference > Types > TnxSqlParamList Type

25.5.86 TnxStatementID Type

NexusDB V2 VCL Reference
TnxStatementID Type

[Contents](#) | [Index](#)

nxsdTypes

Pascal

```
TnxStatementID = type TnxBaseID;
```

File

nxsdTypes

Remarks

Strong type for Statement IDs.

See Also

nxsdTypes

You are here: Symbol Reference > Types > TnxStatementID Type

25.5.87 TnxTablesDescriptorClass Type

NexusDB V2 VCL Reference
TnxTablesDescriptorClass Type

[Contents](#) | [Index](#)

nxsdDataDictionary

Pascal

```
TnxTablesDescriptorClass = class of TnxTablesDescriptor;
```

File

nxsdDataDictionary

Remarks

Class definition for (Main) Table Descriptors.

See Also

nxsdDataDictionary

You are here: Symbol Reference > Types > TnxTablesDescriptorClass Type

25.5.88 TnxTaskID Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTaskID Type

nxsdTypes

Pascal

```
TnxTaskID = type TnxBaseID;
```

File

nxsdTypes

Remarks

Strong type for Task IDs.

See Also

nxsdTypes

You are here: Symbol Reference > Types > TnxTaskID Type

25.5.89 TnxTransContextID Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTransContextID Type

nxsdTypes

Pascal

```
TnxTransContextID = type TnxBaseID;
```

File

nxsdTypes

Remarks

Strong type for transaction context IDs

See Also

nxsdTypes

You are here: Symbol Reference > Types > TnxTransContextID Type

25.5.90 TnxTransID Type

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

TnxTransID Type

nxsdTypes

Pascal

```
TnxTransID = type TnxBaseID;
```

File

nxsdTypes

Remarks

Strong type for Transaction IDs.

See Also

[nxsdTypes](#)

You are here: Symbol Reference > Types > TnxTransID Type

25.5.91 TnxTransportLogOptions Type

NexusDB V2 VCL Reference

[Contents | Index](#)

TnxTransportLogOptions Type

[nxllTransport](#)

Pascal

```
TnxTransportLogOptions = set of TnxTransportLogOption;
```

File

[nxllTransport](#)

Remarks

Log options for the transport.

See Also

[nxllTransport](#)

You are here: Symbol Reference > Types > TnxTransportLogOptions Type

25.6 Variables

25.6.1 nxGetFailedFlag Variable

NexusDB V2 VCL Reference

[Contents | Index](#)

nxGetFailedFlag Variable

[nxsdServerEngine](#)

Pascal

```
nxGetFailedFlag: function : PnxBoolean = _DoNothingStub2;
```

File

[nxsdServerEngine](#)

Remarks

callback function called to get failure specific flags

See Also

[nxsdServerEngine](#)

You are here: Symbol Reference > Variables > nxGetFailedFlag Variable

25.6.2 nxLockTimeout Variable

NexusDB V2 VCL Reference

[Contents | Index](#)

nxLockTimeout Variable

nxsdServerEngine

Pascal

```
nxLockTimeout: procedure = _DoNothingStub;
```

File

nxsdServerEngine

Remarks

callback function called when a locktimeout occurs

See Also

nxsdServerEngine

You are here: Symbol Reference > Variables > nxLockTimeout Variable

25.6.3 nxUnlockTimeout Variable

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxUnlockTimeout Variable

nxsdServerEngine

Pascal

```
nxUnlockTimeout: procedure = _DoNothingStub;
```

File

nxsdServerEngine

Remarks

callback function called when a unlocktimeout occurs

See Also

nxsdServerEngine

You are here: Symbol Reference > Variables > nxUnlockTimeout Variable

25.7 Constants

25.7.1 nxatAliasName Constant

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxatAliasName Constant

nxsdServerEngine

Pascal

```
nxatAliasName = 'Name';
```

File

nxsdServerEngine

Remarks

Alias Type - Name.

See Also

nxsdServerEngine

You are here: Symbol Reference > Constants > nxatAliasName Constant

25.7.2 nxatAliasPath Constant

NexusDB V2 VCL Reference

nxatAliasPath Constant

[Contents](#) | [Index](#)

nxsdServerEngine

Pascal

```
nxatAliasPath = 'Path';
```

File

nxsdServerEngine

Remarks

Alias Type - Path.

See Also

nxsdServerEngine

You are here: Symbol Reference > Constants > nxatAliasPath Constant

25.7.3 nxc_MagicNumber_Cursor Constant

NexusDB V2 VCL Reference

nxc_MagicNumber_Cursor Constant

[Contents](#) | [Index](#)

nxsdServerEngine

Pascal

```
nxc_MagicNumber_Cursor: TnxWord32 = $FF2412FF;
```

File

nxsdServerEngine

Remarks

CURS.

See Also

nxsdServerEngine

You are here: Symbol Reference > Constants > nxc_MagicNumber_Cursor Constant

25.7.4 nxc_MagicNumber_Database Constant

NexusDB V2 VCL Reference

nxc_MagicNumber_Database Constant

[Contents](#) | [Index](#)

nxsdServerEngine

Pascal

```
nxc_MagicNumber_Database: TnxWord32 = $FF3102FF;
```

File

nxsdServerEngine

Remarks

DBAS.

▀ **See Also**

[nxsdServerEngine](#)

You are here: Symbol Reference > Constants > nxc_MagicNumber_Database Constant

25.7.5 nxc_MagicNumber_Session Constant

NexusDB V2 VCL Reference

[Contents | Index](#)

nxc_MagicNumber_Session Constant

[nxsdServerEngine](#)

Pascal

```
nxc_MagicNumber_Session: TnxWord32 = $FF2422FF;
```

▀ **File**

[nxsdServerEngine](#)

▀ **Remarks**

SESS.

▀ **See Also**

[nxsdServerEngine](#)

You are here: Symbol Reference > Constants > nxc_MagicNumber_Session Constant

25.7.6 nxc_MagicNumber_Statement Constant

NexusDB V2 VCL Reference

[Contents | Index](#)

nxc_MagicNumber_Statement Constant

[nxsdServerEngine](#)

Pascal

```
nxc_MagicNumber_Statement: TnxWord32 = $FF23C3FF;
```

▀ **File**

[nxsdServerEngine](#)

▀ **Remarks**

STMT.

▀ **See Also**

[nxsdServerEngine](#)

You are here: Symbol Reference > Constants > nxc_MagicNumber_Statement Constant

25.7.7 nxc_MagicNumber_TaskInfo Constant

NexusDB V2 VCL Reference

[Contents | Index](#)

nxc_MagicNumber_TaskInfo Constant

[nxsdServerEngine](#)

Pascal

```
nxc_MagicNumber_TaskInfo: TnxWord32 = $FF3023FF;
```

File

nxsdServerEngine

Remarks

TASK.

See Also

nxsdServerEngine

You are here: Symbol Reference > Constants > nxc_MagicNumber_TaskInfo Constant

25.7.8 nxc_MagicNumber_TransContext Constant

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxc_MagicNumber_TransContext Constant

nxsdServerEngine

Pascal

```
nxc_MagicNumber_TransContext: TnxWord32 = $FF3237FF;
```

File

nxsdServerEngine

Remarks

TCTX.

See Also

nxsdServerEngine

You are here: Symbol Reference > Constants > nxc_MagicNumber_TransContext Constant

25.7.9 nxc_MainSettingsExtension Constant

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxc_MainSettingsExtension Constant

nxBaseServerComp

Pascal

```
nxc_MainSettingsExtension = '.nxdbworksettings';
```

File

nxBaseServerComp

Remarks

The default extension the current server settings file

See Also

nxBaseServerComp

You are here: Symbol Reference > Constants > nxc_MainSettingsExtension Constant

25.7.10 nxc_SigHeaderBlockV1 Constant

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxc_SigHeaderBlockV1 Constant

nxsdDataDictionary

Pascal

```
nxc_SigHeaderBlockV1: TnxBlockSignature = 'NX!1';
```

File

[nsdDataDictionary](#)

Remarks

Signature for NX V1 tables

See Also

[nsdDataDictionary](#)

You are here: Symbol Reference > Constants > nxc_SigHeaderBlockV1 Constant

25.7.11 nxc_SigHeaderBlockV2 Constant

NexusDB V2 VCL Reference

[Contents | Index](#)

nxc_SigHeaderBlockV2 Constant

nxsdDataDictionary

Pascal

```
nxc_SigHeaderBlockV2: TnxBlockSignature = 'NX!2';
```

File

[nsdDataDictionary](#)

Remarks

Signature for NX V2 tables

See Also

[nsdDataDictionary](#)

You are here: Symbol Reference > Constants > nxc_SigHeaderBlockV2 Constant

25.7.12 nxcBlobTypes Constant

NexusDB V2 VCL Reference

[Contents | Index](#)

nxcBlobTypes Constant

nxsdTypes

Pascal

```
nxcBlobTypes = [nxtBlob..nxtBlobGraphic, nxtBlobWideMemo];
```

File

[nsdTypes](#)

Remarks

Constant for pointing to the last Blob field in the field types. For performance reasons.

See Also

[nsdTypes](#)

You are here: Symbol Reference > Constants > nxcBlobTypes Constant

25.7.13 nxcDefaultServerSearchTimeout Constant

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxcDefaultServerSearchTimeout Constant

nxllTransport

Pascal

```
nxcDefaultServerSearchTimeout = 3000;
```

File

nxllTransport

Remarks

The default timeout for searching servers via broadcast.

See Also

nxllTransport

You are here: Symbol Reference > Constants > nxcDefaultServerSearchTimeout Constant

25.7.14 nxDefaultFilterTimeout Constant

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxDefaultFilterTimeout Constant

nxdb

Pascal

```
nxDefaultFilterTimeout = 500;
```

File

nxdb

Remarks

The default timeout for setting filters in ms.

See Also

nxdb

You are here: Symbol Reference > Constants > nxDefaultFilterTimeout Constant

25.7.15 nxMaxBlobChunk Constant

NexusDB V2 VCL Reference

[Contents](#) | [Index](#)

nxMaxBlobChunk Constant

nxdb

Pascal

```
nxMaxBlobChunk: Integer = 512 * 1024;
```

File

nxdb

Remarks

The maximum number of bytes read/written to the server at once.

0 means do not limit "chunk" sizes, any other value determines the number of bytes.

See Also

nxdb

You are here: Symbol Reference > Constants > nxMaxBlobChunk Constant

25.8 Files

25.8.1 nxBaseServerComp.pas

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**nxBaseServerComp.pas**[Classes](#) | [Constants](#) | [Types](#)**Constants**

Constant	Description
nxc_MainSettingsExtension	The default extension the current server settings file

Types

Type	Description
TNotifyErrorEvent	The type used for error the error notification event.
TnxLoginCallback	This is the call back type for TnxBaseServer.OnShowLogin.

See Also[Classes](#)
[Constants](#)
[Types](#)*You are here:* Symbol Reference > Files > nxBaseServerComp.pas

25.8.2 nxConfigSettings.pas

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**nxConfigSettings.pas**[Classes](#)

Implements classes to persist settings to inifiles.

See Also[Classes](#)*You are here:* Symbol Reference > Files > nxConfigSettings.pas

25.8.3 nxdb.pas

NexusDB V2 VCL Reference[Contents](#) | [Index](#)**nxdb.pas**[Classes](#) | [Constants](#) | [Enumerations](#) | [Records](#) | [Types](#)**Constants**

Constant	Description
----------	-------------

<code>nxDefaultFilterTimeout</code>	The default timeout for setting filters in ms.
<code>nxMaxBlobChunk</code>	The maximum number of bytes read/written to the server at once. 0 means do not limit "chunk" sizes, any other value determines the number of bytes.

Enumerations

Enumeration	Description
<code>TnxCachedDataSetOption</code>	Possible options for a cached dataset.
<code>TnxDataSetInternalState</code>	Dataset states are used internally by the dataset class. There's no need for developers to manipulate these.
<code>TnxDataSetOption</code>	Dataset options definition
<code>TnxFilterType</code>	Possible Filter types for <code>TnxDataset.Filter</code>
<code>TnxKeyIndex</code>	Used internally to address the different key buffers.
<code>TnxSetKeyOption</code>	<code>SetKey</code> option

Types

Type	Description
<code>PnxBaseSession</code>	This is type <code>PnxBaseSession</code> .
<code>PnxKeyBuffer</code>	Used internally to address the different key buffers.
<code>TFieldDefClass</code>	A Class of <code>TFieldDef</code> (needed for abstraction purposes only.)
<code>TnxBaseUpdateHandlerClass</code>	Class defintion for <code>TnxBaseUpdateHandler</code> .
<code>TnxCachedDataSetOptions</code>	Set of <code>TnxCachedDataSetOption</code>
<code>TnxCreateTableEvent</code>	Method defintion for <code>CreateTable</code> callback event.
<code>TnxCursorClass</code>	Class defintion for Cursors.
<code>TnxDataSetInternalStates</code>	Dataset states are used internally by the dataset class. There's no need for developers to manipulate these.
<code>TnxDataSetOptions</code>	Blob Cache Options.
<code>TnxDirectKeySetFieldsMethod</code>	Method defintion for <code>DirectKeySetFields</code> callback method.
<code>TnxLoginEvent</code>	An event to get a user name and password for login purposes. Called from <code>TnxBaseSession</code> .
<code>TnxServerFilterTimeoutEvent</code>	This is an event handler definition for Filter timeouts. It gives the programmer a chance to commence with a filtering even if the filter timeout is reached.
<code>TnxSetKeyOptions</code>	Set of <code>SetKey</code> options.

See Also

[Classes](#)
[Constants](#)
[Enumerations](#)
[Records](#)
[Types](#)

You are here: Symbol Reference > Files > nxdb.pas

25.8.4 nxlComponent.pas

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

nxlComponent.pas

[Classes](#) | [Enumerations](#) | [Interfaces](#) | [Types](#)

Enumeration

Enumeration	Description
TnxLogPriority	This is record TnxLogPriority.
TnxSettingType	Settings data type
TnxState	Possible states for TnxStateComponent
TnxStateTransition	Possible state transitions for TnxStateComponent

Interface

Interface	Description
InxLogData	This is class InxLogData.

Type

Type	Description
EnxComponentExceptionClass	Class definition for Component Exceptions
TnxIterationListClass	Class definition for TnxIterationList
TnxLogPriorities	This is type TnxLogPriorities.
TnxPersistentClass	Class defintion for TnxPersistent.
TnxRegisterableComponentClass	Class definition for TnxRegisterableComponent.

See Also

[Classes](#)
[Enumerations](#)
[Interfaces](#)
[Types](#)

You are here: Symbol Reference > Files > nxlComponent.pas

25.8.5 nxlTransport.pas

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

nxlTransport.pas

[Classes](#) | [Constants](#) | [Enumerations](#) | [Functions](#) | [Interfaces](#) | [Records](#) | [Types](#)

Constant

Constant	Description
nxcDefaultServerSearchTimeout	The default timeout for searching servers via broadcast.

Enumeration

Enumeration	Description
TnxMessageHeaderFlag	the flags that are sent with every message

TnxTransportLogOption	Log options for the transport.
TnxTransportMode	The mode of the transport.

Functions

Function	Description
nxSplitAddress	This function separates the server name in form name@address.

Interfaces

Interface	Description
InxActiveBroadcast	This interface is returned by TnxBaseTransport.BeginBroadcast.
InxBroadcastReply	This interface is passed into InxBroadcastReplyHandler.ReplyReceived.
InxBroadcastReplyHandler	This interface has to be passed into TnxBaseTransport.BeginBroadcast.
InxSessionCallback	This is class InxSessionCallback.

Types

Type	Description
PnxBaseHeader	The base header for a communication messages.
PnxDataMessage	the structure for storing one data message part.
TnxAddSessionEvent	Definition for the OnAddSession event.
TnxBaseTransportClass	Class definition for trnasports.
TnxChooseServerEvent	The definition for the OnChooseServer event
TnxConnectionLostEvent	The definition for the OnConnectionLost event.
TnxFindServersEvent	The definition for the OnFindServers event
TnxMessageHeaderFlags	set of Message flags
TnxRemoveSessionEvent	The definition for the OnRemoveSession event
TnxReplyCallback	Definition for the callback functions used by the transports.
TnxTransportLogOptions	Log options for the transport.

See Also

[Classes](#)
[Constants](#)
[Enumerations](#)
[Functions](#)
[Interfaces](#)
[Records](#)
[Types](#)

You are here: Symbol Reference > Files > nxlITransport.pas

25.8.6 nxptBasePooledTransport.pas

[NexusDB V2 VCL Reference](#)

nxptBasePooledTransport.pas

[Contents | Index](#)

[Classes](#) | [Types](#)

Types

Type	Description
TnxBasePooledTransportClass	This is type TnxBasePooledTransportClass.
TnxNetIdleEvent	This is type TnxNetIdleEvent.

See Also

[Classes](#)
[Types](#)

You are here: Symbol Reference > Files > nxptBasePooledTransport.pas

25.8.7 nxsdDataDictionary.pas

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

nxsdDataDictionary.pas

[Classes](#) | [Constants](#) | [Enumerations](#) | [Interfaces](#) | [Types](#)

Basic data dictionary.

Constants

Constant	Description
nxc_SigHeaderBlockV1	Signature for NX V1 tables
nxc_SigHeaderBlockV2	Signature for NX V2 tables

Enumerations

Enumeration	Description
Tnx1xFileType	File types for the data dictionary.
TnxApplyAt	Type used to indicate where a validation object should be applied.
TnxNullBehaviour	Type to define the Null behaviour of a key

Interfaces

Interface	Description
InxFieldDescriptor	This is class InxFieldDescriptor.
InxFieldsSource	This is class InxFieldsSource.

Types

Type	Description
TnxApplyAtSet	Set of TnxApplyAt
TnxAutoIncStepRange	This is type TnxAutoIncStepRange.
TnxBaseAutoIncDescriptorClass	Class definition for AutoInc Descriptors.
TnxBaseBlobDescriptorClass	Class definition for Blob Descriptors.
TnxBaseBlockHeapDescriptorClass	Class definition for Block Heap Descriptors
TnxBaseCustomDescriptorClass	Class definition for Custom Descriptor
TnxBaseDefaultValueDescriptorClass	Class definition for TnxBaseDefaultValueDescriptor.
TnxBaseFieldsValidationDescriptorClass	Class definition for TnxBaseFieldsValidationDescriptor.

TnxBaseFieldValidationDescriptorClass	Class definition for TnxBaseFieldValidationDescriptor.
TnxBaseHeapDescriptorClass	Class definition for Heap Descriptors.
TnxBaseIndicesDescriptorClass	Class definition for the base class of Indices Descriptors.
TnxBaseRecordCompressionDescriptorClass	Class definition for Compression Descriptors
TnxBaseRecordDescriptorClass	Class definition for Record Descriptors
TnxBaseStreamDescriptorClass	Class definition for Stream Descriptors.
TnxBaseTableDescriptorClass	Class definition for base Table Descriptors.
TnxCustomDescsDescriptorClass	Class definition for Custom Descriptors.
TnxDictionaryItemClass	Class definition for dictionary items.
TnxFieldDescriptorClass	Class definition for Field Descriptors.
TnxFieldsDescriptorClass	Class definition for Fields Descriptors.
TnxFieldsValidationsDescriptorClass	This is type TnxFieldsValidationsDescriptorClass.
TnxFieldValidationsDescriptorClass	Class definition for Validation Descriptors.
TnxFileDescriptorClass	Class definition for File Descriptors
TnxFilesDescriptorClass	Class definition for Files Descriptors.
TnxHeapBlobDescriptorClass	Class definition for Heap Blob Descriptors.
TnxHeapRecordDescriptorClass	Class definition for Heap Record Descriptors.
TnxIndexDescriptorClass	Class definition for Index descriptors.
TnxKeyDescriptorClass	Class definition for Key descriptors.
TnxKeyFieldDescriptorClass	Class definition for Key Field descriptors.
TnxLocaleDescriptorClass	Class definition for TnxLocaleDescriptor.
TnxMainIndicesDescriptorClass	Class definition for the class of Main-Index Descriptors.
TnxNestedTableDescriptorClass	Class definition for Nested(Sub-) table Descriptors.
TnxTablesDescriptorClass	Class definition for (Main) Table Descriptors.

See Also

[Classes](#)
[Constants](#)
[Enumerations](#)
[Interfaces](#)
[Types](#)

You are here: Symbol Reference > Files > nxsdDataDictionary.pas

25.8.8 nxsdServerEngine.pas

[NexusDB V2 VCL Reference](#)
[nxsdServerEngine.pas](#)

[Contents | Index](#)

[Classes](#) | [Constants](#) | [Enumerations](#) | [Functions](#) | [Types](#) | [Variables](#)
 Base server engine classes.

Constants

Constant	Description
nxatAliasName	Alias Type - Name.
nxatAliasPath	Alias Type - Path.
nxc_MagicNumber_Cursor	CURS.
nxc_MagicNumber_Database	DBAS.
nxc_MagicNumber_Session	SESS.
nxc_MagicNumber_Statement	STMT.
nxc_MagicNumber_TaskInfo	TASK.
nxc_MagicNumber_TransContext	TCTX.

Enumerations

Enumeration	Description
TnxEngineAction	Action types for Engine based actions. These types are fired and passed on to Extenders.
TnxStatementExecDirectPhase	The state of the currently executed Statement
TnxTableScope	Scope of a table

Functions

Function	Description
_DoNothingStub	internal use only
_DoNothingStub2	internal use only
nxCheckValidAliasName	This is function nxCheckValidAliasName.
nxCheckValidRelativeTableName	This is function nxCheckValidRelativeTableName.
nxCheckValidRootTableName	This is function nxCheckValidRootTableName.
nxCheckValidStoredProcName	This is function nxCheckValidStoredProcName.
nxCheckValidTableName	This is function nxCheckValidTableName.
nxTableNameIsTempDatabase	returns true if the given aTableName is a database specific temp table
nxTableNameIsTempGlobal	returns true if the given aTableName is a global temp table
nxTableNameIsTempStatement	returns true if the given aTableName is a temp statement

Types

Type	Description
TnxAbstractServerObjectClass	Class definition for Server Objects.
TnxAbstractSessionClass	This is type TnxAbstractSessionClass.
TnxBaseEngineExtenderClass	class definition for Server Extenders
TnxEngineActions	Set of Engine Actions
TnxMappingMethod	Abstract Prototype for MappingMethods

Variables

Variable	Description

<code>nxGetFailedFlag</code>	callback function called to get failure specific flags
<code>nxLockTimeout</code>	callback function called when a locktimeout occurs
<code>nxUnlockTimeout</code>	callback function called when a unlocktimeout occurs

See Also

[Classes](#)
[Constants](#)
[Enumerations](#)
[Functions](#)
[Types](#)
[Variables](#)

You are here: Symbol Reference > Files > `nxsdServerEngine.pas`

25.8.9 nxsdTypes.pas

[NexusDB V2 VCL Reference](#)

[Contents | Index](#)

nxsdTypes.pas

[Constants](#) | [Enumerations](#) | [Records](#) | [Types](#)

Constants

Constant	Description
<code>nxcBlobTypes</code>	Constant for pointing to the last Blob field in the field types. For performance reasons.

Enumerations

Enumeration	Description
<code>TnxBlobCopyMode</code>	Defines if Blobs should be linked, copied or left out of SQL results.
<code>TnxClassListType</code>	possible types for class lists
<code>TnxFieldType</code>	NexusDB field types.
<code>TnxIndexPathPosition</code>	Possible positions in an index
<code>TnxLockConflictType</code>	This is record <code>TnxLockConflictType</code> .
<code>TnxLockPresent</code>	Used to indicate whether a lock is present in a certain server object.
<code>TnxLockRequestType</code>	This is record <code>TnxLockRequestType</code> .
<code>TnxLockResult</code>	This is record <code>TnxLockResult</code> .
<code>TnxLockType</code>	Lock Types fro NexusDB.
<code>TnxOpenMode</code>	Open modes for opening databases, tables.
<code>TnxParamType</code>	Parameter Types
<code>TnxRecordCountOption</code>	Options for <code>GetRecordCountEx</code> .
<code>TnxRecordGetBatchExOption</code>	Options for the batch mode.
<code>TnxSearchKeyAction</code>	Used for quick locating/marketing of keys in the SQL and filter engines.
<code>TnxShareMode</code>	Share modes for opening databases, tables.

TnxStatementType	Type of a Statement
------------------	---------------------

Types

Type	Description
PnxBookmark	Pointer to bookmark data
PnxLockResult	This is type PnxLockResult.
PnxSqlParamDesc	Pointer to TnxSqlParamDesc.
PnxSqlParamList	This is type PnxSqlParamList.
PnxTaskStatus	Task operation status info.
TFormatSettings	This is type TFormatSettings.
TnxBlockSignature	Type for the table block signature. The first 4 characters hold engine and Version information.
TnxCursorID	Strong type for Cursor IDs
TnxDatabaseID	Strong type for Database IDs
TnxFieldTypes	NexusDB field types.
TnxFilterID	Strong type for Filter IDs.
TnxGrowSize	Type for the table grow size.
TnxKeyFilterID	Strong type for KeyFilter IDs.
TnxRecordGetBatchExOptions	Options for the batch mode.
TnxSqlParamList	SQL Parameters.
TnxStatementID	Strong type for Statement IDs.
TnxTaskID	Strong type for Task IDs.
TnxTransContextID	Strong type for transaction context IDs
TnxTransID	Strong type for Transaction IDs.

See Also

[Constants](#)
[Enumerations](#)
[Records](#)
[Types](#)

You are here: Symbol Reference > Files > nxsdTypes.pas

25.8.10 nxServerComp.pas

[NexusDB V2 VCL Reference](#)
nxServerComp.pas

[Contents](#) | [Index](#)

[Classes](#) | [Enumerations](#) | [Types](#)

Enumerations

Enumeration	Description
TnxServerModule	supported Server modules.

Types

Type	Description
TnxServerModules	Set of supported Server Modules.

See Also

[Classes](#)
[Enumerations](#)
[Types](#)

You are here: Symbol Reference > Files > nxServerComp.pas

25.8.11 nxServerManager.pas

NexusDB V2 VCL Reference
nxServerManager.pas

[Contents](#) | [Index](#)

[Classes](#)

See Also

[Classes](#)

You are here: Symbol Reference > Files > nxServerManager.pas

25.8.12 nxtwWinsockTransport.pas

NexusDB V2 VCL Reference
nxtwWinsockTransport.pas

[Contents](#) | [Index](#)

[Classes](#) | [Types](#)

Types

Type	Description
TnxOnAcceptConnection	The type of event method used in TnxCustomConnectionFilter.

See Also

[Classes](#)
[Types](#)

You are here: Symbol Reference > Files > nxtwWinsockTransport.pas

26 Supported Visual Studio versions

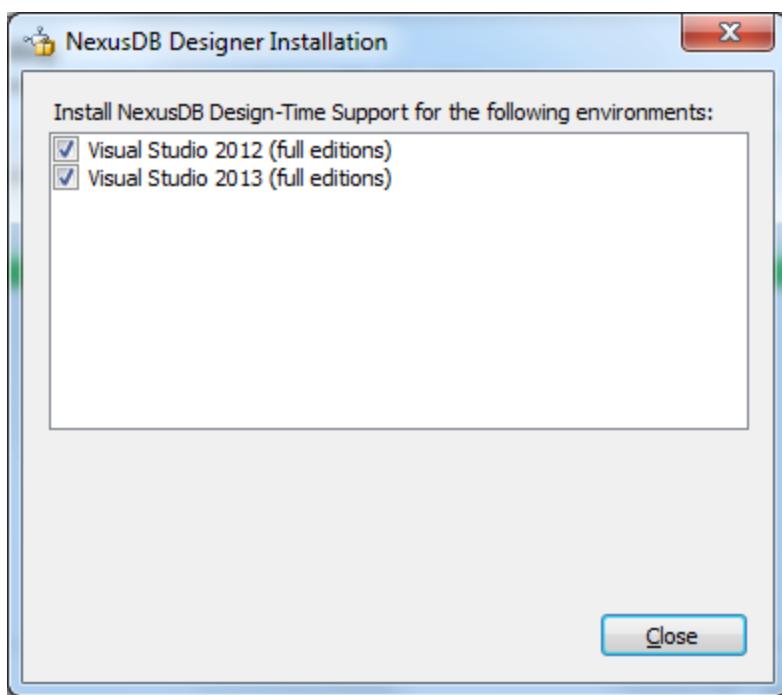
The current version of the Provider supports Visual Studio 2012, Visual Studio 2013, and Visual Studio 2015 IDEs. Visual Studio 2017 is supported when creating connections in code. Versions older than Visual Studio 2012 will not be supported in future Provider updates.

27 Installing the NexusDB ADO.NET Provider

It is recommended to uninstall any already installed versions of the NexusDB Provider (old name) or NexusDB Visual Studio Pack first. Then, run the new installer, and follow the prompts.

The installer we provide assumes that it is being run on a development machine with Visual Studio installed. For installation/deployment on machines without a development environment, see Deploying an Application.

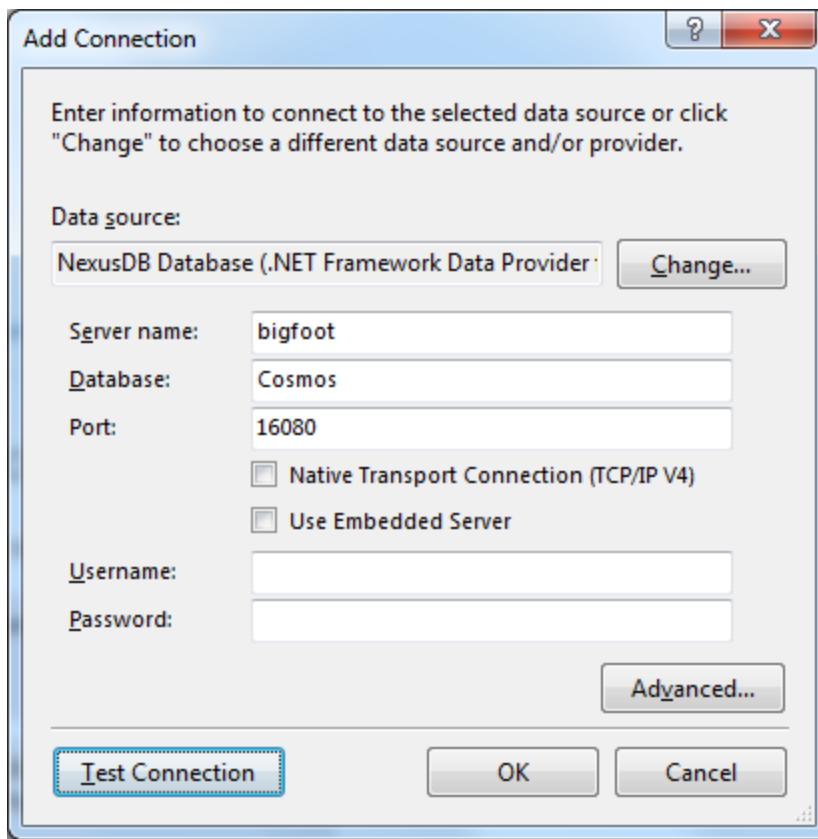
At the end, the installer will present the following dialog, to choose the Visual Studio versions you want to use with the provider. It is recommended to uncheck and check any already checked entries to activate IDE support.



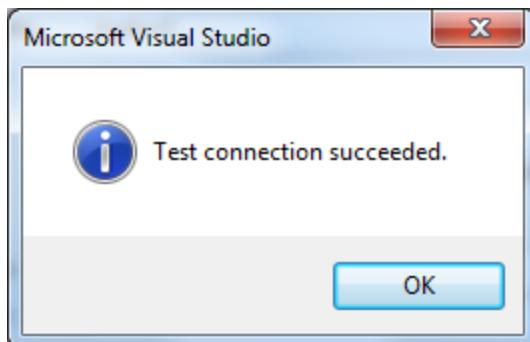
NOTE! In order to take advantage of the Native TCP/IP transport, support DLL binaries have been provided with the Provider. The files are named AdoServerConnectorV4.Dll for the 32 bit NexusDB Server and AdoServerConnectorV4_64.Dll for the 64 bit NexusDB Server. These files are installed in the 32 and 64 bit system directories as well as under the installation\Bin directory. Both NexusDB Server and the Provider makes use of the DLL.

You can now start up Visual Studio, open the Server Explorer, right-click the Data Connection node and choose Add Connection. The NexusDB ADO.NET Provider should be available in the Add Connection dialog, either by being the default provider, or by clicking the Change... button.

Enter the IP address or DNS name of your NexusDB Server machine, enter a known existing database name, and click Test Connection.



You should see



You can now proceed to add NexusDB database support to your projects.

28 Visual Studio Pack Editions

Feature Comparison

The Visual Studio Pack comes in 3 different editions depending on which edition you have purchased. These editions are:

- Standard
- Professional
- Enterprise

To know what your edition contains, check the Feature Matrix:

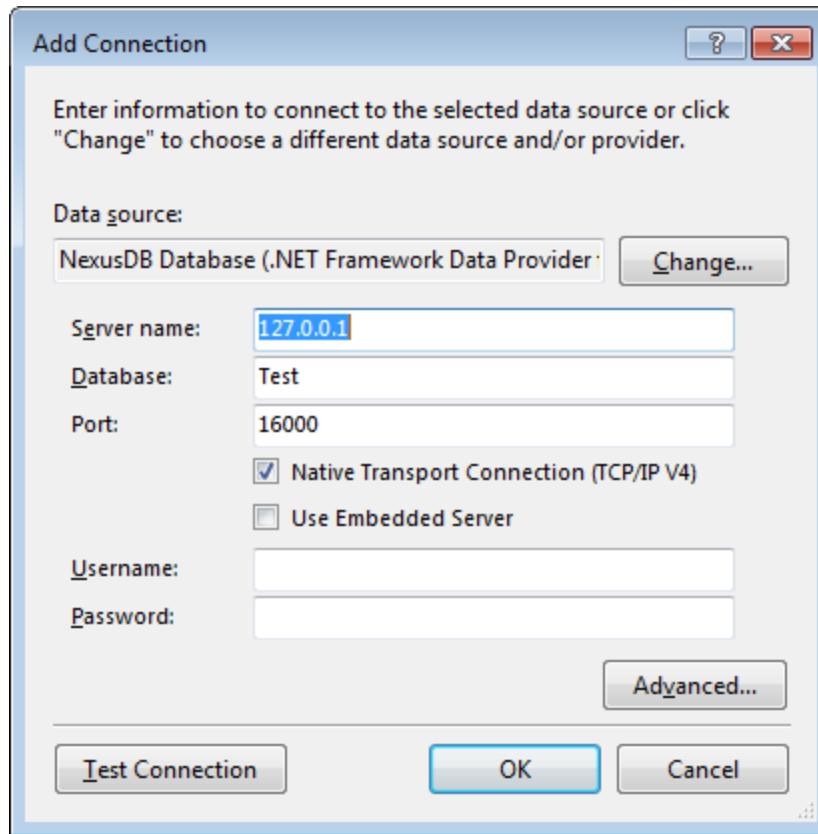
	Enterprise Edition	Professional Edition	Standard Edition
ADO.Net Provider	✓	✓	✓
Visual Studio Integration	✓	✓	✓
Entity Framework Support	✓	✓	✓
ODBC Driver	✓	✓	
PHP Driver	✓	✓	
Encrypted Transports	✓		
Encrypted Table Engines	✓		

29 Choosing the server connection method

There are two ways to connect to a remote NexusDB Server:

- Native TCP/IP transport
- ADO.NET Connector

The Native checkbox in the Visual Studio Add Connection dialog determines which one is used. The default, and recommended, setting is Native selected (the checkbox is checked):

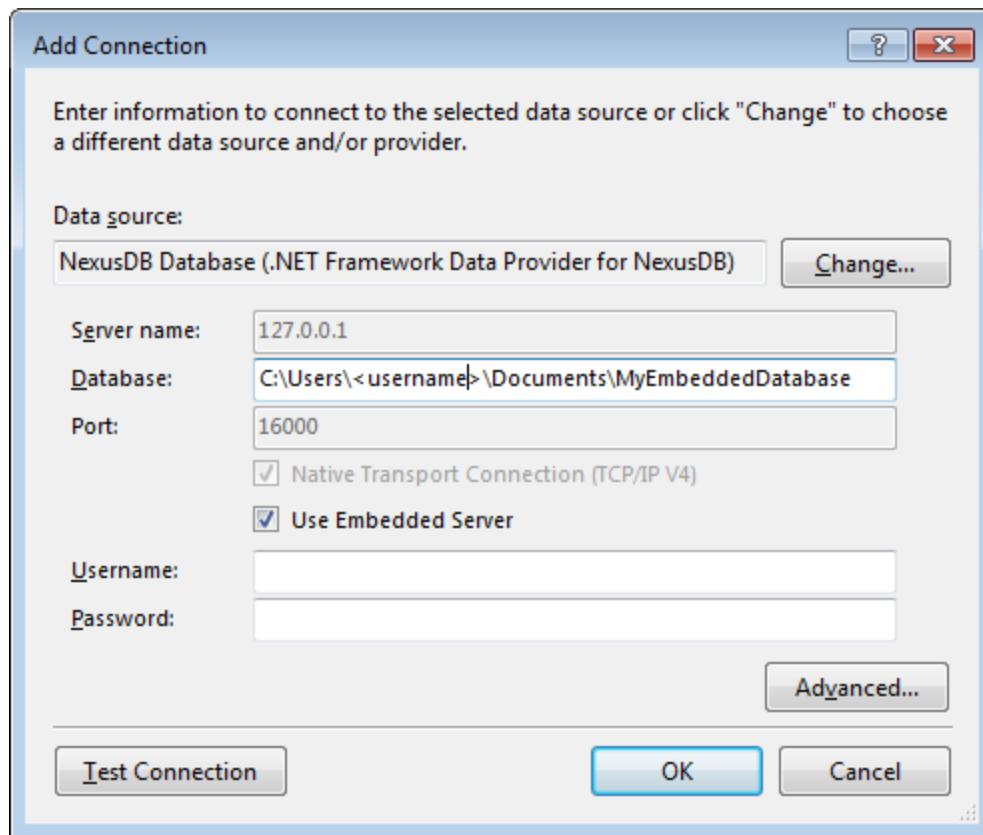


The Native TCP/IP transport is the recommended connection method. This connection method uses the NexusDB TCP/IP transport. The advantage of this transport is that it will detect unexpected disconnections from clients, letting the NexusDB Server free up any resources the client may have in use. Because it relies on the AdoServerConnector DLL, which is a Windows DLL, it requires that the client runs on a Windows OS.

The ADO.NET Connector is a managed connection method that does not rely on a Windows DLL.

30 Single User Mode

The Visual Studio Pack supports creating applications that does not rely on a remote NexusDB Server. Tick the "Use Embedded Server" checkbox to enable this feature.



31 Entity Framework Support

In order for Entity Framework (EF) to cooperate with our ADO.NET Provider, there are configuration steps that must be followed. **Please make sure all of the following steps are completed.**

Framework version supported

The current release of the NexusDB ADO.NET Provider supports version 6 of Entity Framework, starting from EF version 6.1.2. Earlier EF version 6 suffered from bugs affecting our provider. The support is currently EF 6 only, the old version 5 or work in progress version 7 (currently in beta from Microsoft) is not supported.

Visual Studio Prerequisites

In order for the VS 2012/2013 IDEs to support version 6 of EF, the correct framework support tools must be installed. The VS 2012/2013 IDEs currently come with EF 5 tool support. You must manually install the version 6 tools, which you can find on this link:

[Visual Studio EF 6 Tool downloads](#)

Visual Studio 2015 comes with EF 6 tooling support.

Install NexusDB ADO.NET Provider

Perform the installation steps to make sure the provider and NexusDB Server support DLL is available.

Install Entity Framework in your project

Each individual project where you want to use EF needs to have the framework added to it. Open or create a new project, then follow the instructions on the [Install Entity Framework](#) link.

Modify your project

After adding EF to your project, you need to look under the References node for your project in the Solution Explorer, and remove the "EntityFramework.SqlServer" reference. This reference is added automatically when adding EF to the project, but is not needed for a NexusDB based project.

Also, under the project's References node, add references to NexusDB.ADOProvider.dll and NexusDB.ADOProvider.Linq.dll.

You must manually edit the App.Config file for your project in order to add support for the NexusDB ADO.Net Provider. Double-click the App.Config file in the Visual Studio Solution Explorer, then remove the default Sql Server provider info that is present in the entityFramework node. There should be no text left between <entityFramework> and </entityFramework>. Then, paste in the following lines between the <entityFramework> node start and end markers.

```
<defaultConnectionFactory type="NexusDB.ADOProvider.NxConnectionFactory,
NexusDB.ADOProvider.Linq, Culture=neutral,
PublicKeyToken=86f660e26976e61e"></defaultConnectionFactory>
<providers>
```

```

<provider invariantName="NexusDB.ADOProvider"
type="NexusDB.ADOProvider.NxProviderServices, NexusDB.ADOProvider.Linq,
Culture=neutral, PublicKeyToken=86f660e26976e61e" />
</providers>

```

The end result should look like this, including the entityFramework start/end tags:

```

<entityFramework>
  <defaultConnectionFactory type="NexusDB.ADOProvider.NxConnectionFactory,
NexusDB.ADOProvider.Linq, Culture=neutral,
PublicKeyToken=86f660e26976e61e"></defaultConnectionFactory>
  <providers>
    <provider invariantName="NexusDB.ADOProvider"
type="NexusDB.ADOProvider.NxProviderServices, NexusDB.ADOProvider.Linq,
Culture=neutral, PublicKeyToken=86f660e26976e61e" />
  </providers>
</entityFramework>

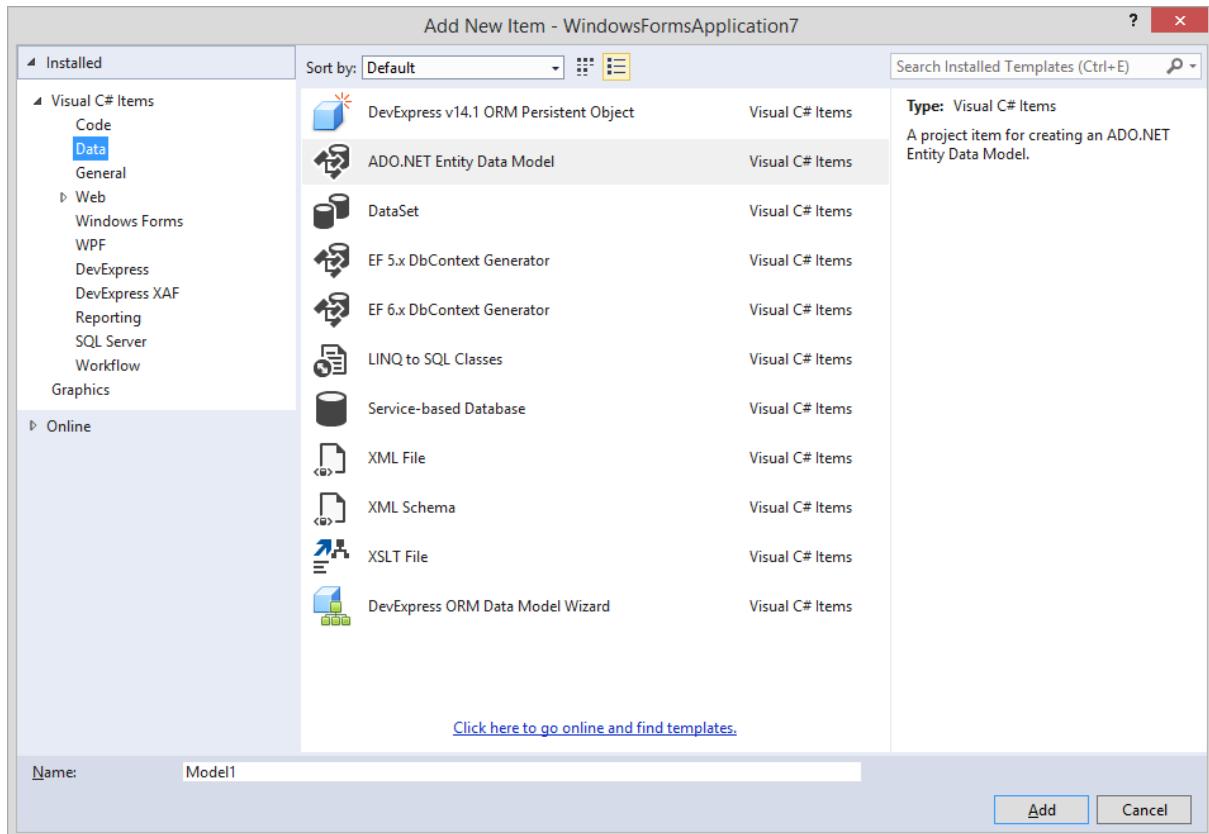
```

Make sure to save the changes to the App.Config file to disk before continuing.

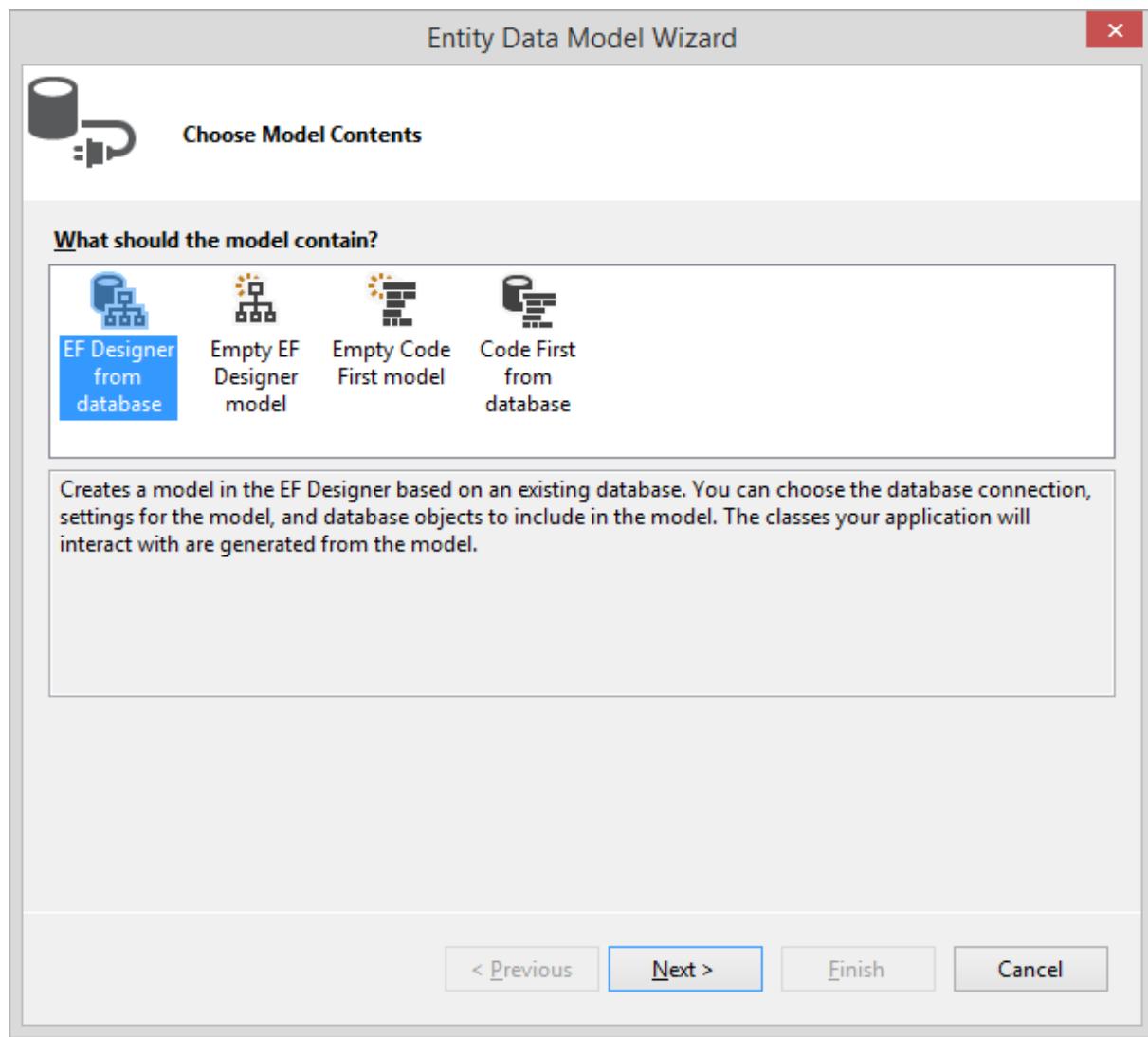
Test the Entity Framework support

After completing the above steps, with your project opened in VS, try the following:

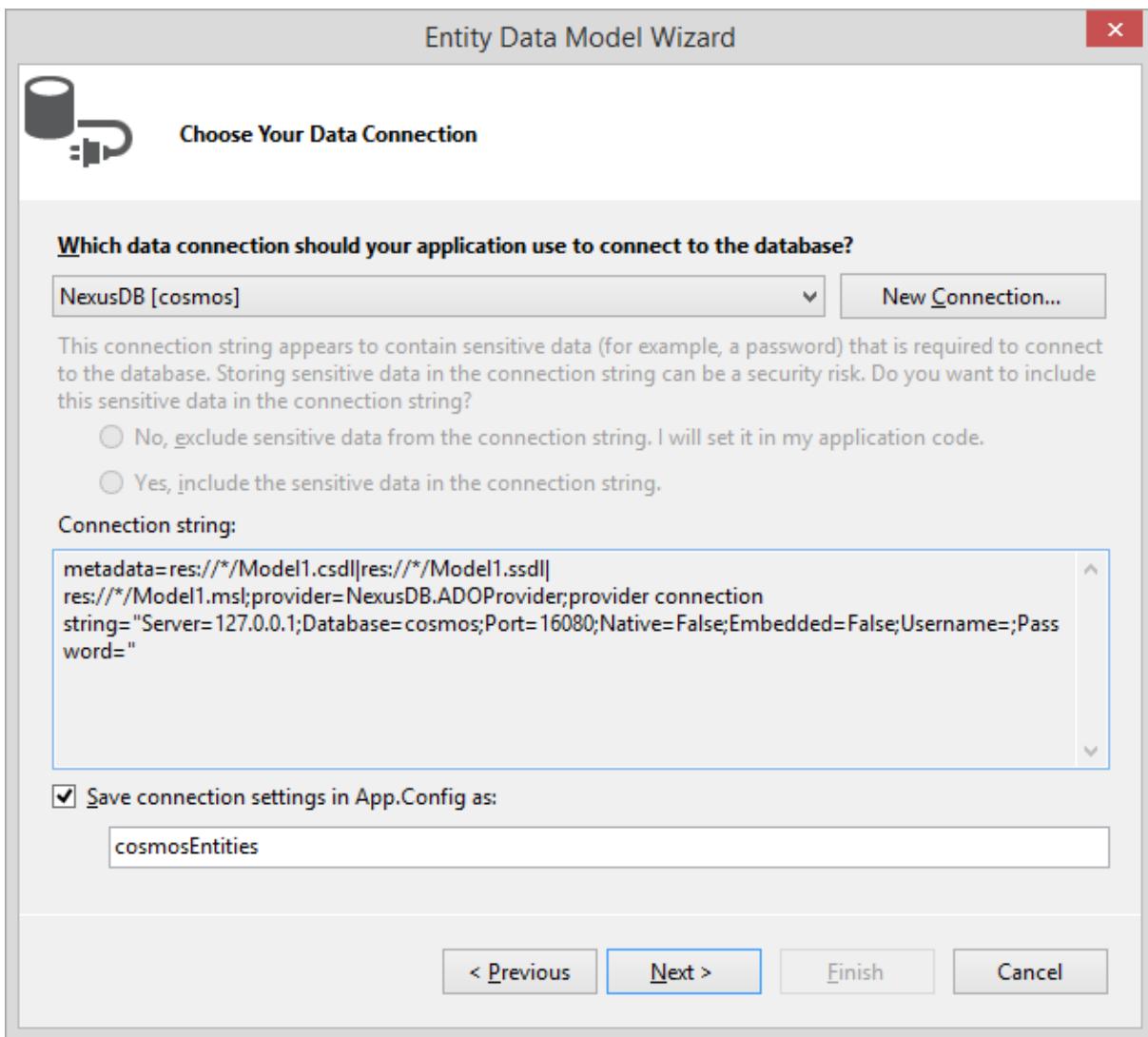
Open the Project | Add New Item... dialog. In that dialog, choose "Data" node under the "Visual C# Items" node on the left hand side.



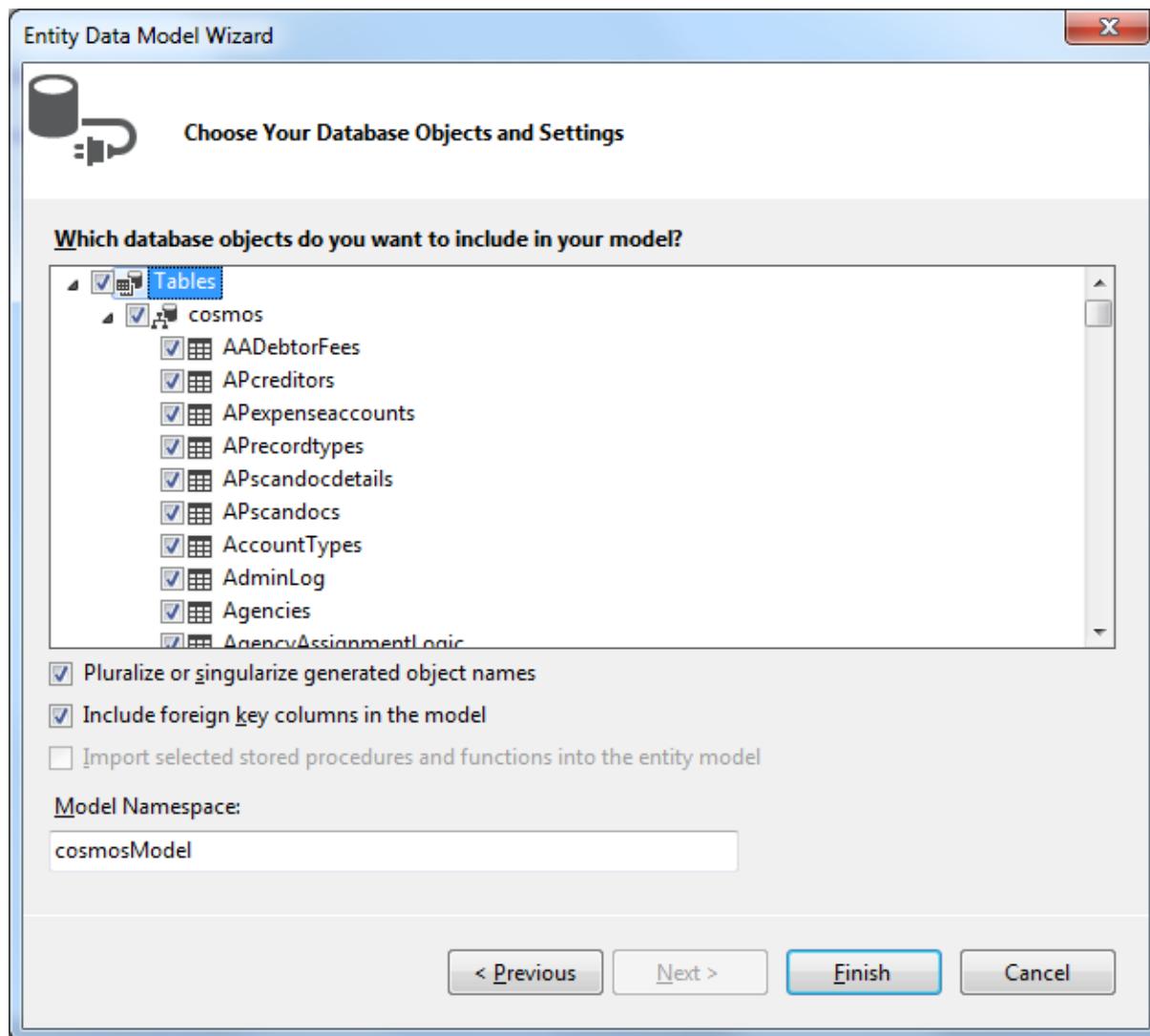
Double-click the "ADO.NET Entity Data Model" entry that becomes visible in the middle list in the dialog. The Entity Data Model Wizard will open.



Double-click the "EF Designer from database" entry or click Next. Choose an existing connection using the NexusDB ADO.NET Provider, or create a new connection.



The tables in your chosen database should then be displayed in the wizard:



Tick the boxes for the tables you want to use, and click Finish: The IDE will then display the data model.

32 Connection String Parameters

Connection String Parameters

Parameter Name	Example	Possible Values	Description
Native	False	True, False	False if the transport used for connecting to the server is the ADO.NET Connector, True if the transport used is the native TCP/IP transport
Embedded	False	True, False	If set to True the provider uses a built-in (Embedded) Server Engine instead of connecting to a remote server
Port	16000		Port for transport connection. Default value for Native connection: 16000, ADO Connector: 16080
Server	nexusdb@127.0.0.1		The name of the server to connect to in the form of IP address or machinename (DNS name).
Database	Northwind		The database to connect to. For Embedded mode this must be a locally accessible directory path (check user rights for ASP.Net!). For remote server mode, it is the name of the database alias.
Compression	9	0 to 10	Compression used for transferring data: 0 .. no compression; 1 to 9 .. zip compression (9 is highest compression); 10 .. RLE compression
BlockReadSize	512000		Number of bytes that are fetched from the server per message. Set to 0 to disable (not recommended!).
UserName			The username for connecting to the database.
PassWord			The password for connecting to the database.
Timeout	20000	1 to 4 billion	Timeout value for queries, in milliseconds. Default is 10000 (10 seconds).
NativeTransportType	TCP/IPV4	TCP/IPV4, DiffieHellman2048 AesCcm, DiffieHellman2048 AesGcm, DiffieHellman8192 AesCcm,	Internal identifier of the transport used when Native is true. Note: The encrypted DiffieHellman transports are only available in the Enterprise SKU.

		DiffieHellman8192 AesGcm, DiffieHellman8192 ShaCal	
MaxRam	-1	-1 to 2 billion	Only used if Embedded is True. Sets the amount of RAM used for the embedded server engine block cache, measured in MegaBytes. Default is -1, which is a special value meaning around 1.5GB in 32 bit and (system RAM - 2GB) in 64 bit.

33 Creating the Northwind sample database

Creating the Northwind sample database

All examples use the Northwind sample database.

The SQL script to create the database is installed in the "..\Sample Databases" subfolder of your Visual Studio Pack installation.

Create the database

Here's a step by step guide to running the SQL script from Enterprise manager.

- Create a new directory which will later hold the tables
- Make sure the NexusDB Server service and its tray icon is running
- Open the web configuration from the tray icon
- Select the Aliases node on the left
- Add a new alias called Northwind and point it to the newly created directory which will hold the database
- Make sure the TCP/IPv4 and Direct TCP/.NET Transports are set to active
- Open NexusDB Enterprise Manager
- You should see the server in the server window. Open one of the transport nodes.
- Right click the Northwind alias
- Select SQL in the popup menu and the SQL window opens on the right hand side
- Press CTRL-O or select Query/Open from the main menu
- Navigate to the sample databases directory and select the Northwind.sql file
- Press CTRL-E or select Query/Execute to run the script. This can take a few seconds.

That's it. You have created the full example database and it is ready for use.

34 Creating a simple NexusDB Application

Creating a simple C# Application

Here we will show how to build a very simple database application using NexusDB ADO.NET Provider with Microsoft Visual Studio 2015.

Hint

For this example we are using the Northwind sample database, if you haven't already created it, please refer to Creating the Northwind sample database to do so.

Step 1 - Creating a new project

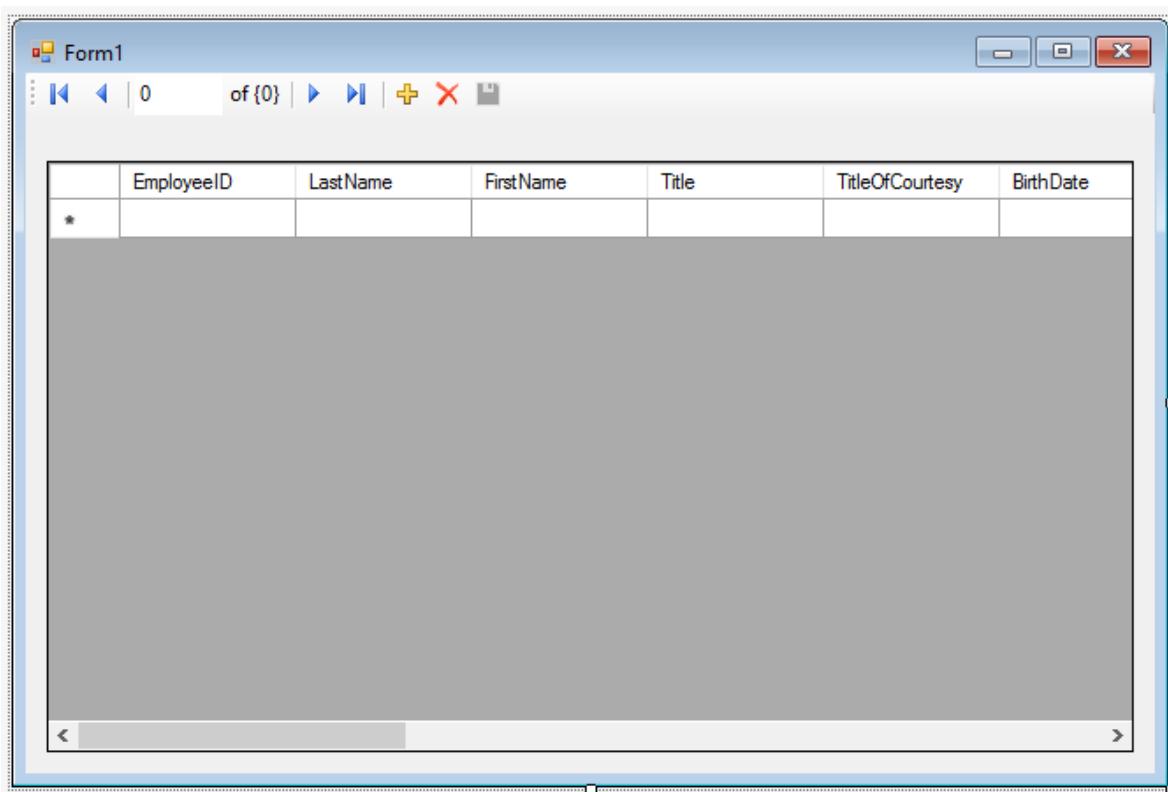
Start Visual Studio 2015 and create a new windows forms project using the **File \ New** menu item.

Step 2 - Setting up the connection

Open the Data Sources pane in Visual Studio, and click "Add new data source". Click "Database" and "Next", then "Dataset" and Next, then click "New connection". In the Add Connection dialog, if the default Data source is not the NexusDB Database Provider, click "Change" and choose the NexusDB Database Provider.

Set Server Name to the DNS name or IP address of your server, and set Database to "Northwind" (without quotes). Leave the rest of the settings at their default, and click Test Connection. If the NexusDB Server is running, you should see the "Test connection succeeded" message. Click Ok and Next to get to the "choose your database objects" dialog page. For this example, expand the "Tables" node and choose the Employees table. Click Finish.

In the Data Sources pane, the Employees table is now available. Click and drag the table to your form. Visual Studio will automatically create a grid and a navigator on your form:



Click Start in Visual Studio or press F5. Your new application should now start up and display the data:

The screenshot shows a Windows application window titled "Form1". At the top, there is a toolbar with icons for back, forward, search, and other operations. Below the toolbar is a status bar showing "1 of 9". The main area is a grid table with the following data:

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
▶	1	Davolio	Nancy	Sales Represent...	Ms.	8/12/1948
	2	Fuller	Andrew	Vice President, S...	Dr.	19/02/1952
	3	Leverling	Janet	Sales Represent...	Ms.	30/08/1963
	4	Peacock	Margaret	Sales Represent...	Mrs.	19/09/1937
	5	Buchanan	Steven	Sales Manager	Mr.	4/03/1955
	6	Suyama	Michael	Sales Represent...	Mr.	2/07/1963
	7	King	Robert	Sales Represent...	Mr.	29/05/1960
	8	Callahan	Laura	Inside Sales Coor...	Ms.	9/01/1958
	9	Dodsworth	Anne	Sales Represent...	Ms.	27/01/1966
*						

35 Deploying an Application

Deploying an Application

#todo



36 Online Support

Newsgroups

Our primary support channel is our newsgroups. To connect to our support newsgroups, please review the options here: [NexusDB Newsgroups](#).

For questions about the provider and connection methods, use the [nexusdb.public.support.dotNetProvider](#) newsgroup.

For questions about the NexusDB Server and Enterprise Manager, use the [nexusdb.public.support](#) newsgroup.

For questions about the PHP or ODBC drivers, use their respective newsgroups.

IRC

To connect to our live IRC chat channel, review the options on this link.

Support Level Agreement

If you have purchased an SLA extension to your subscription, please review the SLA document emailed to you, which contains your priority support contact information. Depending on your SLA level, you may have priority newsgroup access, priority email, and phone number options for contacting us.

If you require guaranteed support response times, and are considering becoming a member of our SLA program, please review the information on our [here](#).

Index

- - -

- (subtraction operator) 395

- # -

(temporary table) 515

(temporary table) 515

#L+ option 178

- (-

(### (temporary table) 515

- * -

* (asterisk) 551

* (multiplication operator) 395

- / -

/ (division operator) 395

- _ -

_DoNothingStub function 1185

_DoNothingStub2 function 1185

- | -

|| (concatenation operator) 395

- + -

+ (addition operator) 395

- < -

< (less than operator) 395

<= (ess than or equals operator) 395

<> (not equals operator) 395
<absolute value expression> 451
<add column definition> 541, 574
<add table constraint definition> 541, 574
<addition operator> 395
<aggregate function> 441
<alter column action> 541, 574
<alter column definition> 541, 574
<alter table action> 541, 574
<alter table statement> 541, 574
<approximate numeric literal> 399
<approximate numeric type> 435
<arctangent function> 458
<as subquery clause> 515
<assembly definition> 539
<assembly specifier> 539
<assert table statement> 591
<assignment operator> 395
<assignment source> 581
<assignment statement> 581
<assignment target> 581
<between predicate part 2> 483
<between predicate> 483
<binary large object string type> 431
<binary string literal> 401
<blob position expression> 446
<blob primary> 410
<blob switch> 513
<blob value expression> 410
<boolean AND operator> 395
<boolean factor> 413
<boolean literal> 406
<boolean NOT operator> 395
<boolean OR operator> 395
<boolean predicand> 413
<boolean primary> 413
<boolean term> 413
<boolean test> 413
<boolean type> 438
<boolean value expression> 413
<case abbreviation> 419
<case expression> 419
<case operand> 419
<case specification> 419
<cast operand> 422
<catch clause> 589
<catch statement block> 589
<ceiling function> 457

<char function> 472
<char length expression> 449
<char length units> 446
<character large object type> 431
<character like predicate part 2> 486
<character pattern> 486
<character primary> 410
<character representation> 401
<character set clause> 426
<character string literal> 401
<character string type> 431
<character substring function> 465
<character value expression> 410
<character value function> 465
<check constraint definition> 515
<collate clause> 426
<collation clause> 515
<column constraint definition> 515
<column constraint> 515
<column definition> 515
<commit statement> 570
<common value expression> 407
<comp op> 482
<compare flag> 426
<comparison predicate part 2> 482
<comparison predicate> 482
<compound statement> 578
<concatenation operator> 395
<constraint name definition> 515
<cosine function> 459
<cross join> 551
<current date value function> 475
<current timestamp value function> 476
<cursor specification> 551
<data type> 426
<date literal> 403
<date string> 403
<datetime literal> 403
<datetime primary> 412
<datetime type> 439
<datetime value expression> 412
<datetime value function> 474
<day-time interval> 404
<day-time literal> 404
<default clause> 515
<default option> 515
<delete rule> 515
<delete statement: searched> 564
<deleting predicate> 498
<derived column list> 551
<deterministic characteristic> 503
<digit> 399
<division operator> 395
<drop assembly statement> 550
<drop column definition> 541, 574
<drop index statement> 545
<drop routine statement> 548
<drop table constraint definition> 541, 574
<drop table statement> 544
<drop trigger statement> 547
<drop view statement> 546
<dynamic parameter specification> 424
<else clause> 419
<equals operator> 395
<equivalent predicate> 494
<exact numeric literal> 399
<exact numeric type> 435
<exists predicate> 491
<explicit row value constructor> 416
<exponent> 399
<exponential function> 454
<external body reference> 503
<extract expression> 448
<extract field> 448
<extract source> 448
<factor> 408
<floor function> 457
<fold> 470
<from clause> 551
<from constructor> 560
<from subquery> 560
<general literal> 398
<general logarithm> 453
<general set function> 441
<general value specification> 407
<greater than operator> 395
<greater than or equals operator> 395
<group by clause> 551
<guid literal> 401
<having clause> 551
<hexit> 401
<if statement elseif clause> 582
<if statement then clause> 582
<if statement> 582
<in predicate part 2> 484
<in predicate value> 484

<in predicate> 484
<index definition> 526
<index element> 526
<index switch> 513
<insert columns and source> 560
<insert statement> 560
<inserting predicate> 496
<interval literal> 404
<interval qualifier> 404
<interval string> 404
<iterate statement> 583
<join specification> 551
<join type> 551
<joined table> 551
<language clause> 503
<leave statement> 585
<length expression> 449
<less than operator> 395
<less than or equals operator> 395
<like clause> 515
<like predicate> 486
<literal> 398
<local declaration list> 578
<log switch> 513
<mantissa> 399
<match predicate part 2> 493
<match predicate> 493
<modulus expression> 451
<multiplication operator> 395
<national character large object type> 431
<national character string literal> 401
<national character string type> 431
<natural join> 551
<natural logarithm> 452
<nonparenthesized value expression primary> 407
<non-second primary datetime field> 404
<not equals operator> 395
<>null ordering> 526
<>null predicate part 2> 487
<>null predicate> 487
<numeric primary> 408
<numeric type> 435
<numeric value expression dividend> 451
<numeric value expression divisor> 451
<numeric value expression> 408
<numeric value function> 445
<octet length expression> 449
<odd predicate> 495
<order by clause> 551
<ordinal function> 461
<parameter mode> 503
<pi function> 462
<position expression> 446
<power function> 455
<predicate> 481
<primary datetime field> 404
<protected statement block> 589
<qualified join> 551
<quantified comparison predicate part 2> 488
<quantified comparison predicate> 488
<quantifier> 488
<query expression> 551
<query specification> 551
<query term> 551
<quote symbol> 401
<quote> 401
<random function> 463
<references specification> 515
<referential action> 515
<referential constraint definition> 515
<referential triggered action> 515
<repeat statement> 586
<result> 419
<return statement> 577
<return value> 577
<rollback statement> 571
<round function> 462
<routine body> 503
<routine characteristic> 503
<routine characteristics> 503
<routine invocation> 503
<row subquery> 417
<row value constructor predicand> 416
<row value constructor> 416
<row value element list> 416
<row value element> 416
<row value expression> 416
<row value predicand> 416
<scalar subquery> 417
<search condition> 481
<searched when clause> 419
<select list> 551
<select statement> 551
<select sublist> 551
<set clause list> 562
<set function type> 441

<set passwords statement> 572
 <signal statement> 588
 <signed integer> 399
 <signed numeric literal> 399
 <simple when clause> 419
 <simplify switch> 513
 <sine function> 460
 <sort direction> 526
 <sort options> 526
 <sort specification> 551
 <SQL argument list> 503
 <SQL argument> 503
 <SQL control statement> 509
 <SQL data change statement> 509
 <SQL data statement> 509
 <SQL parameter declaration list> 503
 <SQL parameter declaration> 503
 <SQL procedure statement> 509
 <SQL routine body> 503
 <SQL schema definition statement> 509
 <SQL schema manipulation statement> 509
 <SQL schema statement> 509
 <SQL session statement> 509
 <SQL statement list> 578
 <SQL transaction statement> 509
 <SQL variable declaration> 579
 <SQL variable name list> 579
 <SQL-data access indication> 503
 <SQL-invoked routine> 503
 <square root> 456
 <start transaction statement> 568
 <string position expression> 446
 <string value expression> 410
 <string value function> 465
 <subquery> 417
 <subtraction operator> 395
 <table constraint definition> 515
 <table constraint> 515
 <table contents source> 515
 <table definition> 515
 <table element> 515
 <table expression> 551
 <table factor> 551
 <table function derived table> 551
 <table reference> 551
 <table subquery> 417
 <table value constructor> 560
 <target specification> 407
 <term> 408
 <time interval> 404
 <time literal> 403
 <time string> 403
 <timeout switch> 513
 <timestamp literal> 403
 <timestamp string> 403
 <tostring function> 472
 <tostringlen function> 473
 <transaction mode> 568
 <transition variable> 529
 <trigger column list> 529
 <trigger definition> 529
 <trigger event> 529
 <triggered SQL statement> 529
 <trim character> 470
 <trim function> 470
 <trim source> 470
 <trim specification> 470
 <truth value> 413
 <try statement> 589
 <unique constraint definition> 515
 <unique predicate> 492
 <unsigned integer> 399
 <unsigned literal> 398
 <unsigned numeric literal> 399
 <unsigned value specification> 407
 <update rule> 515
 <update statement: searched> 562
 <updating predicate> 497
 <user-defined function> 536
 <user-defined procedure> 532
 <value expression primary> 407
 <value expression> 407
 <value specification> 407
 <view definition> 528
 <view specification> 528
 <when operand> 419
 <where clause> 551
 <while statement> 587
 <year-month literal> 404

- = -

= (assignment operator) 395
 = (equals operator) 395

- > -

> (greater than operator) 395
 >= (greater than or equals operator) 395

- A -

About NexusDB SQL Reference 383
 ABS 451
 Absolute Value Expression 451
 Active Query 183
 ADD 541, 572, 574
 Add column definition 541, 574
 Add table constraint definition 541, 574
 ADO 201
 ADO2FF 191
 ADO2NX 191
 Advanced settings 187
 AFTER 529
 Aggregate Functions 441
 Alias Name 106
 Alias names 196
 ALL 441, 488, 551
 ALTER 541, 574
 Alter column definition 541, 574
 ALTER TABLE statement 541, 574
 Always Commit 115
 Always Rollback 115
 AND 395, 413, 483
 ANY 488
 Appendices
 Additional SQL:2003 Features 600
 Core SQL:2003 Features 595
 Apply At 168
 ApplyOnInsert 168
 ApplyOnUpdate 168
 Approximate Numeric Literal 399
 Approximate Numeric Type 435
 Arctangent Function 458
 Arithmetic operators 395
 AS 422, 515, 528, 529, 532, 536, 551, 562, 564
 AS subquery clause 515
 ASC 526
 ASSEMBLY 539, 550
 ASSERT 591
 ASSERT TABLE statement 591

ATAN 458
 ATAN2 458
 ATN2 458
 ATOMIC 578
 Attach 152
 AUTHORIZATION 539
 AUTOINC 435
 Autolnc Engine 166
 AVG 441

- B -

BDE 201, 202
 BEFORE 529
 BEGIN 578
 BETWEEN 483
 Between Predicate 483
 BIGINT 435
 BINARY 431
 BINARY LARGE OBJECT 431
 Binary Large Object String Type 431
 Binary String Literal 401
 BLOB 431
 BLOB Fields 173, 177, 178, 181
 BLOB files 210
 BLOB Graphic 162
 BLOB Memo 162
 Blob Value Expression 410
 BLOCK 515
 Block Size 164
 BLOCKSIZE 515
 Blowfish/RC4 Secured Transport 107
 BOOL 438
 BOOLEAN 438
 Boolean Literals 406
 Boolean operators 395
 Boolean Types 438
 Boolean Value Expressions 413
 Borland Database Engine 191, 196
 BOTH 470
 Broadcast Thread Priority 111
 BROUND 462
 Browse Table 157
 browser window 177
 Browsing Table Data 173
 BY 551
 BYTE 435
 Byte Array 162

BYTEARRAY 426

- C -

C/S 103
 CALL 576
 CALL statement 576
 CALLED 503
 CALLED ON NULL INPUT 503
 Can't open tablename 164
 CASCADE 515
 CASE 419, 469, 486, 526
 Case Expressions 419
 CAST 422
 Cast Specification 422
 CATCH 589
 CEIL 457
 CEILING 457
 Ceiling Function 457
 CHAR 162, 431
 Char Function 472
 CHAR LARGE OBJECT 431
 CHAR VARYING 431
 CHAR_LENGTH 449
 CHARACTER 426, 431
 CHARACTER LARGE OBJECT 431
 Character Large Object Type 431
 CHARACTER SET 426
 Character sets and collations 426
 Character String Literal 401
 Character String Type 431
 Character Value Expression 410
 CHARACTER VARYING 431
 CHARACTER_LENGTH 449
 CHARACTERS 446, 449, 465, 469
 CHECK 515
 CHR 472
 Clear Range 173
 client/server 103, 168
 CLOB 431
 Clone Table 157
 Close Inactive Folders 115
 Close Inactive Tables 115
 CLR 503
 CLR routines 501
 COALESCE 419
 CODEPAGE 426
 COLLATE 426

COLLATION 515
 COLUMN 541, 574
 Column constraint 515
 Column definition 515
 column names 185, 187
 COM Transport 108
 command line parameters 192
 Comments 394
 COMMIT 570
 COMMIT statement 570
 Commit Transaction 155
 Comparison operators 395
 Comparison Predicate 482
 compound indexes 192
 Compound statement 578
 Compress Limit 110
 Concatenation operators 395
 Concurrent IOCP Threads 110
 configuration file 193
 connection to a server 152
 CONSTRAINT 515, 541, 574
 CONTAINS 500, 503
 CONTAINS SQL 503
 Control Statements
 About Control Statements 575
 CALL statement 576
 Compound statement 578
 DECLARE variable statement 579
 IF statement 582
 ITERATE statement 583
 LEAVE statement 585
 REPEAT statement 586
 RETURN statement 577
 SET statement 581
 SIGNAL statement 588
 TRY statement 589
 WHILE statement 587
 Copy To Table 176, 180
 COS 459
 Cosine Function 459
 COUNT 441
 CREATE
 CREATE ASSEMBLY statement 539
 CREATE FUNCTION statement 536
 CREATE INDEX statement 526
 CREATE PROCEDURE statement 532
 CREATE TABLE statement 515
 CREATE TRIGGER statement 529

CREATE
 CREATE VIEW statement 528
 create new Tables 160
 Create table 155
 CROSS 551
 CROSS JOIN 551
 CSV Import 155, 157
 CSV Import Wizard 185
 current date and time 168
 Current Date Function 475
 Current Time Function 475
 Current Timestamp Function 476
 CURRENT_DATE 475, 515
 CURRENT_TIME 475, 515
 CURRENT_TIMESTAMP 476, 515
 CURRENT_USER 407, 478, 515
 Cursor specification 551
 Cursors 551

- D -

DATA 503, 515
 Data Grid 172, 173, 178
 Data Loss Action 167
 Data Statements
 About Data Statements 551
 DELETE statement 564
 INSERT statement 560
 SELECT statement 551
 UPDATE statement 562

Data types 426
 Database List 152
 Database Popup Menu 155
 DATE 403, 439
 Date Literal 403
 date/time formats 187
 DATETIME 439
 DateTime Literals 403
 DateTime Types 439
 DateTime Value Expressions 412
 DateTime Value Functions

 About DateTime Value Functions 474
 Current Date Function 475
 Current Time Function 475
 Current Timestamp Function 476

DAY 404, 469, 477
 DAYOFYEAR 404
 Day-Time Interval 404

Day-Time Literal 404
 DEC 435
 DECIMAL 435
 DECLARE 579
 DECLARE variable statement 579
 DEFAULT 416, 515, 541, 560, 574, 581
 Default Descriptor 168
 Default Timeout 147, 148
 Default Value 168
 Default Values 168
 DELETE 515, 529, 564
 Delete all the records 157
 Delete Database Alias 155
 Delete Records 176
 DELETE statement 564
 Delete Table 157
 DELETING 498
 Deleting Predicate 498
 DESC 526
 DESCRIPTION 515, 529, 532, 536, 541, 574
 Design Report 176, 180, 189
 DETERMINISTIC 503
 Directory Mode 209
 disk fragmentation 164
 DISTINCT 441, 551
 DO 587
 DOUBLE 435
 DOUBLE PRECISION 435
 DROP 541, 574
 DROP ASSEMBLY statement 550
 DROP FUNCTION statement 548
 DROP INDEX statement 545
 DROP PROCEDURE statement 548
 DROP ROUTINE statement 548
 DROP TABLE statement 544
 DROP TRIGGER statement 547
 DROP VIEW statement 546
 Drop column definition 541, 574
 Drop table constraint definition 541, 574
 DWORD 435

- E -

EACH 529
 Edit Range 173
 ELSE 419, 582
 ELSEIF 582
 embedded server 147, 199, 203

EMPTY 515
Empty All Tables 155
Empty strings 206
ENCRYPT 515
encrypt tables 165
ENCRYPT WITH 515
ENCRYPTION 515
Encryption Engine 165
END 419, 578, 582, 586, 587, 589
END IF 582
END REPEAT 586
END WHILE 587
ENGINE 515
Engine Options 115
Engine Selection 195
Enterprise Manager 124, 196
EnxAbortStateChange class 603
EnxAdvAutoIncDescriptorException class 603
EnxBaseAutoIncDescriptorException class 603
EnxBaseBlobDescriptorException class 604
EnxBaseBlockHeapDescriptorException class 604
EnxBaseCustomDescriptorException class 605
EnxBaseDefaultValueDescriptorException class 605
EnxBaseDictionaryException class 605
EnxBaseFieldValidationDescriptorException class 606
EnxBaseHeapDescriptorException class 606
EnxBaseIndicesDescriptorException class 606
EnxBaseKeyDescriptorException class 607
EnxBaseLogException class 607
EnxBaseRecordCompressionDescriptorException class 608
EnxBaseRecordDescriptorException class 608
EnxBaseStreamDescriptorException class 608
EnxBaseTableDescriptorException class 609
EnxCommandHandlerException class 609
EnxCompKeyDescriptorException class 609
EnxComponentException class 610
 nxcCreate 610, 611, 612, 613
EnxComponentExceptionClass type 1214
EnxConstDefaultValueDescriptorException class 614
EnxCustomDescsDescriptorException class 614
EnxDataDictionaryException class 615
EnxDictionaryItemException class 615
EnxExtTextKeyFieldDescriptorException class 615
EnxFieldDescriptorException class 616
EnxFieldsDescriptorException class 616
EnxFieldValidationsDescriptorException class 616
EnxFileDescriptorException class 617
EnxFilesDescriptorException class 617
EnxHeapBlobDescriptorException class 618
EnxHeapRecordDescriptorException class 618
EnxIndexDescriptorException class 618
EnxKeyFieldDescriptorException class 619
EnxLocaleDescriptorException class 619
EnxLocalizedDictionaryItemException class 619
EnxLoggableComponentException class 620
EnxMainIndicesDescriptorException class 620
EnxMappedKeyDescriptorException class 621
EnxMinMaxValidationDescriptorException class 621
EnxNestedTableDescriptorException class 621
EnxNoChangeValidationDescriptorException class 622
EnxNoNullCompKeyDescriptorException class 622
EnxPluginCommandHandlerException class 622
EnxRefKeyDescriptorException class 623
EnxRegisterableComponentException class 623
EnxSqlCheckValidationDescriptorException class 623
EnxStateComponentException class 624
EnxTablesDescriptorException class 624
EnxTextKeyFieldDescriptorException class 625
EnxTransportException class 625
EQUIVALENT 494
Equivalent Predicate 494
ERROR_MESSAGE 478, 589
ESCAPE 486
Event Log 113, 208
Exact Numeric Literal 399
Exact Numeric Type 435
EXCEPT 551
execution plan 178, 181
EXISTS 491, 541, 544, 545, 546, 547, 548, 550, 574
Exists Predicate 491
EXP 454
Exponential Function 454
Expression indexes 192
EXTENDED 435
EXTERNAL 503
external server 199
EXTRACT 448
Extract Expression 448

- F -

FALSE 406, 413, 438
 FastReports 189
 Field Descriptors 167
 field map 167
 field Name 167
 field Type 167
 Field Types 162
 File Descriptors 164
 File Extension 164
 File Type 185
 Find Nearest 173
 FIRST 526
 fixed width formats 185
 FlashFiler 2 191, 199
 FLOAT 435
 floating point 192
 floating point fields 167
 FLOOR 457
 Floor Function 457
 Fold Function 470
 FOR 465, 469, 529
 FOR EACH 529
 Force Failsafe 115
 FOREIGN 515
 FOREIGN KEY 515
 formatting options 187
 FROM 448, 465, 470, 539, 551, 564
 FULL 493, 551
 FULL JOIN 551
 FUNCTION 536, 548
 Functions
 About Functions 441
 Aggregate Functions 441
 DateTime Value Functions 474
 Numeric Value Functions 445
 String Value Functions 465
 System Functions 478

- G -

General Logarithm 453
 Getting Started with NexusDB 102
 GLOBAL 515
 Global Defaults 183

Global Options 147, 148, 178
 GROUP 469, 551
 GROUP BY 551
 Grouping Data 172
 GROW 515
 Grow Blocks 164
 GROWSIZE 515
 GUI settings 115
 GUID 426
 GUID Literal 401

- H -

HAVING 551
 Heartbeat Frequency 147, 148
 HOUR 404, 469, 477
 How to add a new importer 213

- I -

Identifiers 391
 IDENTITY 464
 IF 541, 544, 545, 546, 547, 548, 550, 574, 582
 IF EXISTS 541, 544, 545, 546, 547, 548, 550, 574
 IF statement 582
 IGNORE 426, 469, 486, 526
 ignore all indexes 206
 IGNORE CASE 526
 IMAGE 431
 Implementation Details 210
 import operation 208
 Importer 191
 IN 446, 469, 477, 484, 503
 In Predicate 484
 INDEX
 CREATE INDEX statement 526
 DROP INDEX statement 545
 Index Descriptors 169
 index names 206
 Index Preferences 206
 indexed value 173
 Information Schema
 System Tables 593
 INITIAL 515
 Initial Blocks 164
 INITIALSIZE 515
 InMem only 115

INNER	551	GetMsgNumber	1179
INNER JOIN	551	GetPriority	1180
INOUT	503	GetUTCWhen	1180
INPUT	503	GetWhen	1180
INSERT	529, 560	GetWhenBias	1181
INSERT INTO	560	Msg	1181
INSERT statement	560	MsgNumber	1182
INSERTING	496	Priority	1182
Inserting Predicate	496	UTCWhen	1182
INT	435	When	1182
INTEGER	435	WhenBias	1183
Internal Server	147	InxSessionCallback interface	1183
INTERSECT	551	Post	1184
INTERVAL	404	Request	1184
Interval Literals	404	IS	413, 487
INTO	551, 560	ISODAYOFMONTH	404
Introduction	381	ISODAYOFWEEK	404
InxActiveBroadcast interface	1169	ISODAYOFYEAR	404
Resume	1169	ISOMONTH	404
Stop	1170	ISO_WEEK	404
Suspend	1170	ISOYEAR	404
InxBroadcastReply interface	1170	ITERATE	583
GetServerID	1171	ITERATE statement	583
GetServerName	1171	 - J -	
GetServerVersion	1171	Jet Engine	202
InxBroadcastReplyHandler interface	1172	JOIN	551
ReplyReceived	1172	Join specification	551
InxFieldDescriptor interface	1172	 - K -	
FieldDecPI	1174	KANA	426
FieldType	1174	KANA TYPE	426
FieldUnits	1174	KEY	515
GetFieldDecPI	1173	Keywords	386
GetFieldType	1173	Known Issues	192
GetFieldUnits	1173	 - L -	
InxFieldsSource interface	1175	LANGUAGE	503
BlobsSupported	1175	LARGE	431
FieldCount	1178	LARGEINT	435
FieldDescriptor	1178	LAST	526
GetCursor	1175	LASTAUTOINC	464
GetFieldAsVariant	1176	LastAutoinc Function	464
GetFieldCount	1176		
GetFieldDescriptor	1176		
GetFieldForFilter	1177		
GetFieldFromName	1177		
InxLogData interface	1178		
Category	1181		
GetCategory	1179		
GetMsg	1179		

LEADING 470
 LEAVE 585
 LEAVE statement 585
 LEFT 469, 551
 LEFT JOIN 551
 Length Expression 449
 LIKE 486
 LIKE clause 515
 Like Predicate 486
 LIST 441
 Listen Thread Priority 111
 Literals 404
 About Literals 398
 Boolean Literals 406
 DateTime Literals 403
 Numeric Literals 399
 String Literals 401
 Live Backup 155
 Live Dataset 183
 Live Datasets 172
 live queries 178
 LN 452
 LOCAL 515
 LOCALE 426
 Locale Descriptor 164
 LOCALTIME 475, 515
 LOCALTIMESTAMP 476, 515
 LOG 453
 log window 178
 Long File Problems 210
 Long Import Durations 210
 loss of data 167
 LOWER 470

- M -

maintenance 157
 Map From Field 167
 MATCH 493
 Match Predicate 493
 MAX 441
 Max RAM 115
 MED 441
 Microsoft Data Access Objects 191
 MIN 441
 MINUTE 404, 469, 477
 Misinterpretation of the Table Types 210
 MOD 451

MODIFIES 503
 MODIFIES SQL DATA 503
 Modulus Expression 451
 MONEY 435
 MONTH 404, 469, 477
 Multi-statement scripts 178

- N -

NAME 503
 Named Pipe Transport 110
 NATIONAL 431
 NATIONAL CHAR 431
 NATIONAL CHAR VARYING 431
 NATIONAL CHARACTER 431
 NATIONAL CHARACTER LARGE OBJECT 431
 National Character Large Object Type 431
 National Character String Literal 401
 National Character String Type 431
 NATIONAL CHARACTER VARYING 431
 NATURAL 551
 Natural Logarithm 452
 NCHAR 431
 NCHAR LARGE OBJECT 431
 NCHAR VARYING 431
 NCLOB 431
 NEW 529
 New Database Alias 152, 155
 New Table 155, 157
 NEWGUID 407, 478, 515
 Next Connection 182
 Next Table 176
 NO 503, 515
 NO DATA 515
 NO SQL 503
 non-null 167
 NONSPACE 426
 NOT 395, 413, 483, 484, 486, 487, 503, 515, 541, 574, 578
 NOT DETERMINISTIC 503
 NOT NULL 515, 541, 574
 NSINGLECHAR 431
 NULL 416, 419, 422, 487, 503, 515, 541, 574, 577, 581
 Null fields 206
 Null Predicate 487
 Null Values 397
 NULLIF 419

NULLS 526
NULLSTRING 431
NUMERIC 435
Numeric Literals 399
Numeric Types 435
Numeric Value Expressions 408
Numeric Value Functions
 About Numeric Value Functions 445
 Absolute Value Expression 451
 Arctangent Function 458
 Ceiling Function 457
 Cosine Function 459
 Exponential Function 454
 Extract Expression 448
 Floor Function 457
 General Logarithm 453
 LastAutoinc Function 464
 Length Expression 449
 Modulus Expression 451
 Natural Logarithm 452
 Ordinal Function 461
 PI Function 462
 Position Expression 446
 Power Function 455
 Random Function 463
 Round Function 462
 Sine Function 460
 Square Root Function 456
NVARCHAR 431
nxatAliasName constant 1249
nxatAliasPath constant 1250
nxBaseServerComp.pas 1255
nxc_MagicNumber_Cursor constant 1250
nxc_MagicNumber_Database constant 1250
nxc_MagicNumber_Session constant 1251
nxc_MagicNumber_Statement constant 1251
nxc_MagicNumber_TaskInfo constant 1251
nxc_MagicNumber_TransContext constant 1252
nxc_MainSettingsExtension constant 1252
nxc_SigHeaderBlockV1 constant 1252
nxc_SigHeaderBlockV2 constant 1253
nxcBlobTypes constant 1253
nxcDefaultServerSearchTimeout constant 1254
nxCheckValidAliasName function 1186
nxCheckValidRelativeTableName function 1186
nxCheckValidRootTableName function 1186
nxCheckValidStoredProcedureName function 1187
nxCheckValidTableName function 1187
nxConfigSettings.pas 1255
nxdb.pas 1255
nxDefaultFilterTimeout constant 1254
nxGetFailedFlag variable 1248
nxIComponent.pas 1257
nxITransport.pas 1257
nxLockTimeout variable 1248
nxMaxBlobChunk constant 1254
nxptBasePooledTransport.pas 1258
nxsdDataDictionary.pas 1259
nxsdServerEngine.pas 1260
nxsdTypes.pas 1262
nxServerComp.pas 1263
nxServerManager.pas 1264
nxSplitAddress function 1188
nxTableNamelsTempDatabase function 1188
nxTableNamelsTempGlobal function 1188
nxTableNamelsTempStatement function 1189
nxtwWinsockTransport.pas 1264
nxUnlockTimeout variable 1249

- O -

OBJECT 431
OCTET_LENGTH 449
OCTETS 446, 449, 465
ODD 495
Odd Predicate 495
OF 529
OLD 529
ON 503, 515, 526, 529, 551
Operator precedence 395
Operators 395
Optimistic Locks 147, 148
Optimistic Recordlocks 148
OR 395, 413
ORD 461
ORDER 551
ORDER BY 551
Ordinal Function 461
OUT 503
OUTER 551
OVERLAY 469, 477

- P -

Pack All Tables 155

Pack Table 157
 parameter "auto" 192
 parameter "nowait" 192
 Parameter values 180
 Parameters 424
 PARTIAL 493
 PASSWORDS 572
 PERCENT 551
 PI 462
 PI Function 462
 PLACING 469, 477
 PnxBaseHeader type 1214
 PnxBaseSession type 1215
 PnxBookmark type 1215
 PnxDataMessage type 1215
 PnxKeyBuffer type 1216
 PnxLockResult type 1216
 PnxSqlParamDesc type 1216
 PnxSqlParamList type 1217
 PnxTaskStatus type 1217
 Port 110
 POSITION 446
 Position Expression 446
 POWER 455
 Power Function 455
 PRECISION 435
 Predefined Data Types 429
 Predicates
 About Predicates 481
 Between Predicate 483
 Comparison Predicate 482
 Deleting Predicate 498
 Equivalent Predicate 494
 Exists Predicate 491
 In Predicate 484
 Inserting Predicate 496
 Like Predicate 486
 Match Predicate 493
 Null Predicate 487
 Odd Predicate 495
 Quantified Comparison Predicate 488
 Unique Predicate 492
 Updating Predicate 497
 Prepared Statements 512
 Previous Connection 182
 PRIMARY
 PRIMARY KEY 515
 primary index 206

Print Preview 176, 180, 189
 PROCEDURE 532, 548
 Procedure Language 503

- Q -

Quantified Comparison Predicate 488
 Query Connections Menu 182
 Query expression 551
 Query Menu 180
 query name 183
 Query Options Menu 183
 query property 183
 Query specification 551
 Query View Menu 181
 Query Window 178

- R -

RAND 463
 Random Function 463
 read only 115
 read-only 178
 Readonly Datasets 147, 148
 READS 503
 READS SQL DATA 503
 REAL 435
 Rearrange Columns 172
 RecNo Support 147, 148
 Record Count 177
 Record Engine 171
 Recover Records 157
 RECREV 426
 Redefine 157
 REFERENCES 515
 REFERENCING 529
 Referential constraint definition 515
 Referential Integrity 168
 Refresh Database List 152
 Refresh Server List 152
 Refresh Table List 155
 Register Server 152
 regular identifier 167
 Regular Identifiers 160
 REMOVE 572
 Rename Database Alias 155
 Rename Table 157

REPEAT 586
 REPEAT statement 586
 Reset Columns 176
 Resize Column Widths 172
 RESTRICT 515, 541, 544, 546, 548, 574
 restrict the data 173
 Result data types of aggregations 426
 RETURN 577
 RETURN statement 577
 RETURNS 503, 536
 RETURNS NULL ON NULL INPUT 503
 reusing values 166
 RIGHT 469, 551
 RIGHT JOIN 551
 ROLLBACK 571
 ROLLBACK statement 571
 Rollback Transaction 155
 ROUND 462
 Round Function 462
 ROUTINE 548
 Routine Characteristics 503
 Routine Invocation 503
 ROW 529
 Row subquery 417
 Row Value Expressions 416
 ROWSAffected 407, 478
 ROWSRead 407, 478
 Runtime Loaded Packages 147, 148

- S -

Scalar subquery 417
 Schema Statements
 About Schema Statements 514
 ALTER TABLE statement 541, 574
 CREATE ASSEMBLY statement 539
 CREATE FUNCTION statement 536
 CREATE INDEX statement 526
 CREATE PROCEDURE statement 532
 CREATE TABLE statement 515
 CREATE TRIGGER statement 529
 CREATE VIEW statement 528
 DROP ASSEMBLY statement 550
 DROP INDEX statement 545
 DROP ROUTINE statement 548
 DROP TABLE statement 544
 DROP TRIGGER statement 547
 DROP VIEW statement 546

search condition 481
 SECOND 404, 469, 477
 Secure Server 115
 SELECT 551
 SELECT statement 551
 Sequential Access Index 173
 SERIALIZABLE 568
 Server Addresses 110
 Server Engine Settings 115
 Server Engine statistics 123
 Server Info Plugin 117
 Server List 152
 Server Name 115
 Server Popup Menu 152
 Server Settings File 127
 Server Statistics 152
 Server Thread Priority 110
 Servers Window 150
 server-side resources 148
 Session Statements
 About Session Statements 572
 SET PASSWORDS statement 572
 SESSION_USER 407, 478, 515
 SET 515, 541, 562, 572, 574, 581
 Set Autolnc 157
 SET DEFAULT 515
 Set Field to Null 176
 SET NULL 515
 Set Password 155, 157
 SET PASSWORDS statement 572
 Set Range 173
 SET statement 581
 Set Timeout 178
 SHORTINT 435
 SHORTSTRING 431
 Show Blob Fields 177, 181
 Show Filter 177
 Show Range 177
 Show Record Count 177
 Side Effects 209
 SIGNAL 588
 SIGNAL statement 588
 SIMILAR 469, 499
 SIMPLE 493
 SIN 460
 Sine Function 460
 SINGLECHAR 431
 SMALLINT 435

SNAPSHOT 568
 SOME 488
 SORT 426
 Sorting Data 172
 Special Statements
 About Special Statements 591
 ASSERT TABLE statement 591
 SQL 503
 SQL Data Types
 About SQL Data Types 426
 Boolean Types 438
 DateTime Types 439
 Numeric Types 435
 Overview of Predefined Data Types 429
 String Types 431
 SQL engine 118
 SQL extensions 178
 SQL Language Elements
 About SQL Language Elements 386
 Comments 394
 Identifiers 391
 Keywords 386
 Literals 398
 Null Values 397
 Operators 395
 Parameters 424
 Value Expressions 407
 SQL Parameter Declaration 503
 SQL Procedure Statements 509
 SQL routines 501
 SQL script 178
 SQL scripts 180
 SQL Statements
 About SQL Statements 512
 Control Statements 575
 Data Statements 551
 Prepared Statements 512
 Schema Statements 514
 Session Statements 572
 Special Statements 591
 Statement Switches 513
 Transaction Statements 568
 SQL:2003 178
 SQL-Invoked Routines 503
 SQRT 456
 Square Root Function 456
 START 568
 Start Transaction 155
 START TRANSACTION statement 568
 STARTS WITH 500
 Statement Switches 513
 STD 441
 STORAGE 515
 STORAGE ENGINE 515
 Stored Procedures and Functions
 About Stored Procedures and Functions 501
 Procedure Language 503
 SQL Procedure Statements 509
 SQL-Invoked Routines 503
 STRING 426
 String Handling 206
 String Literals 401
 String Types 431
 String Value Expressions 410
 String Value Functions
 About String Value Functions 465
 Char Function 472
 Fold Function 470
 Substring Function 465
 ToString Function 472
 ToStringLen Function 473
 Trim Function 470
 strong encryption 165
 Subqueries 417
 subset of the available fields 196
 SUBSTRING 465
 Substring Function 465
 SUM 441
 SYMBOLS 426
 syntax highlighting 183
 System Functions 478
 SYSTEM_ROW# 407, 478

- T -

TABLE 536, 551, 591
 ALTER TABLE statement 541, 574
 CREATE TABLE statement 515
 DROP TABLE statement 544
 Table constraint 515
 Table expression 551
 Table List 155
 Table Menu 176
 Table name 163
 Table operators 551
 Table Options Menu 178

Table Popup Menu 157
Table reference 551
Table subquery 417
Table Types 210
Table View Menu 177
TCP/IP 124
TCP/IPv4 Transport 111
TEMPORARY 515
Temporary Storage 115
Temporary tables 515
TEXT 431
TFieldDefClass type 1218
TFormatSettings type 1218
THEN 419, 582
TIME 403, 439
Time Interval 404
Time Literal 403
TIMESTAMP 403, 439
Timestamp Literal 403
TINYINT 435
TNotifyErrorEvent type 1218
Txn1xFileType enumeration 1189
TnxAbstractCursor class 625
 .Dictionary 656
 .AfterConstruction 627
 .AutoIncGet 627
 .AutoIncSet 627
 .BeforeDestruction 628
 .BlobCreate 628
 .BlobCreateFile 628
 .BlobDelete 629
 .BlobFree 629
 .BlobGetLength 630
 .BlobModified 630
 .BlobRead 630
 .BlobTruncate 631
 .BlobWrite 631
 .BookmarkAsVariant 632
 .ChangePassword 632
 .CompareBookmarks 633
 .CompareKeys 633
 .CopyRecords 634
 .CopyRecordsEx 635
 .Create 626
 .Database 657
 .DatabaseNext 657
 .DatabasePrev 657
 .DeleteRecords 635
 .Destroy 626
 .Duplicate 635
 .FilterActivate 636
 .FilterAddCallback 636
 .FilterAddCustom 636
 .FilterAddExpression 637
 .FilterDeactivate 637
 .FilterRemove 638
 .FilterSetTimeout 638
 .FullName 658
 .GetBookmark 638
 .GetIndexID 639
 .GetRecordCount 639
 .GetRecordCountAsync 639
 .GetRecordCountEx 640
 .GetRecordCountExAsync 640
 .KeyFilterActivate 640
 .KeyFilterAddCallback 641
 .KeyFilterAddCustom 641
 .KeyFilterAddExpression 642
 .KeyFilterDeactivate 642
 .KeyFilterRemove 642
 .LookupByID 643
 .RangeReset 643
 .RangeSet 643
 .RecNoGet 644
 .RecNoSet 645
 .RecNoSupported 645
 .RecordDelete 646
 .RecordGet 646
 .RecordGetBatch 646
 .RecordGetForKey 647
 .RecordGetNext 648
 .RecordGetPrior 648
 .RecordInsert 648
 .RecordInsertBatch 649
 .RecordIsLocked 649
 .RecordLockRelease 650
 .RecordModify 650
 .RequestNestedTransactions 650
 .SetToBegin 651
 .SetToBookmark 651
 .SetToCursor 651
 .SetToEnd 652
 .SetToFilter 652
 .SetToKey 652
 .SwitchToIndex 653
 .TableDescriptor 658

TnxAbstractCursor class 625
TablesLocked 654
TableLockAcquire 654
TableLockRelease 655
TableStreamDelete 655
TableStreamGetList 655
TableStreamRead 656
TableStreamWriter 656
TnxAbstractDatabase class 658
AfterConstruction 660
BeforeDestruction 660
Create 659
CursorOpen 660
Destroy 659
GetChangedTables 661
GetFreespace 661
GetPath 662
LookupByID 662
Session 674
StatementAlloc 663
StatementExecDirect 663
TableAddIndex 664
TableAutoIncGet 664
TableBackup 665
TableBuild 665
TableChangePassword 666
TableDelete 666
Table DropIndex 667
TableEmpty 667
TableExists 668
TableGetDictionary 668
TableGetList 669
TablePack 669
TableRebuildIndex 670
TableRecover 670
TableRename 671
TableRestructure 671
TransactionCommit 672
TransactionCorrupted 672
TransactionGetLevel 672
TransactionRollback 673
TransactionStart 673
TransactionStartWith 673
TransContext 674
TnxAbstractSecurityMonitor class 675
HasAdmins 675
HasUsers 675
IsAdmin 676
UiStateVisible 676
TnxAbstractServerObject class 676
BeforeDestruction 677
Free 677
RemoteThreadPriority 677
TnxAbstractServerObjectClass type 1219
TnxAbstractSession class 678
AfterConstruction 679
AliasAdd 679
AliasDelete 680
AliasGetList 680
AliasGetPath 681
AliasModify 681
Authenticated 687
BeforeDestruction 682
CleanedUp 687
ClientVersion 687
CloseInactiveFolders 682
CloseInactiveTables 682
ConnectedFrom 688
Create 678
DatabaseOpen 683
Destroy 679
Failed 684
GetRegisteredClassList 684
GetUserInfo 684
LookupByID 685
Password 688
PasswordAdd 685
PasswordRemove 685
PasswordRemoveAll 686
ServerEngine 688
ServerVersion 689
TransContextCreate 686
UserName 689
TnxAbstractSessionClass type 1219
TnxAbstractStatement class 689
AfterConstruction 690
BeforeDestruction 691
Create 690
Database 693
Destroy 690
Exec 691
GetParams 692
LookupByID 692
Prepare 692
SetParams 693
TnxAbstractTaskInfo class 694

TnxAbstractTaskInfo class	694	AutoIncEngine	712
AfterConstruction	695	CheckValid	710
BeforeDestruction	695	Destroy	710
Cancel	696	FileNumber	712
Create	694	HeaderSection	712
Destroy	695	isEqual	711
GetStatus	696	LoadFromReader	711
LookupByID	696	SaveToWriter	711
Session	697		
TnxAbstractTimeoutObject class	697	TnxBaseAutoIncDescriptorClass type	1221
Create	698	TnxBaseBlobDescriptor class	713
OptionClear	698	BlobEngine	716
OptionGet	698	CheckValid	714
OptionGetEffective	699	Create	713
OptionList	699	Destroy	714
OptionListEffective	699	FileNumber	716
OptionSet	700	GetList	714
Timeout	700	HeaderSection	716
TnxAbstractTransContext class	701	isEqual	715
AfterConstruction	702	LoadFromReader	715
BeforeDestruction	702	SaveToWriter	715
Create	701	TnxBaseBlobDescriptorClass type	1221
Destroy	702	TnxBaseBlobStream class	717
Failed	703	Destroy	717
LookupByID	703	Truncate	718
Session	706	TnxBaseBlockHeapDescriptor class	718
SessionNext	706	BlockHeapEngine	720
SessionPrev	706	isEqual	718
TransactionCommit	703	LoadFromReader	719
TransactionCorrupted	704	SaveToWriter	719
TransactionGetLevel	704	TnxBaseBlockHeapDescriptorClass type	1221
TransactionRollback	704	TnxBaseCommandHandler class	720
TransactionStart	705	Create	720
TransactionStartWith	705	Destroy	721
TnxAddSessionEvent type	1219	UISettingsVisible	721
TnxAdvAutoIncDescriptor class	707	TnxBaseComponentConfiguration class	721
InitialValue	708	GetValue	722
isEqual	707	GetValueStream	722
LoadFromReader	707	SetValue	723
SaveToWriter	708	SetValueStream	723
Step	709	TnxBaseConnectionFilter class	723
TnxAliasDescriptor class	709	TnxBaseCustomDescriptor class	724
TnxApplyAt enumeration	1190	CheckValid	724
TnxApplyAtSet type	1220	isEqual	724
TnxAutoGuidDefaultValueDescriptor	168	Name	725
TnxAutoGuidDefaultValueDescriptor class	709	TnxBaseCustomDescriptorClass type	1222
TnxAutoIncStepRange type	1220	TnxBaseDefaultValueDescriptor class	725
TnxBaseAutoIncDescriptor class	710	ApplyAt	727
		ApplyOnInsert	728

TnxBaseDefaultValueDescriptor class	725	CheckValid 743 Clear 743 Create 741 Destroy 742 GetDescriptorFromName 744 GetIndexFromName 744 HeaderSection 748 InsertIndexAt 744 IsEqual 745 LoadFromReader 746 MoveIndex 746 RemoveIndex 746, 747 SaveToWriter 747
TnxBaseDefaultValueDescriptorClass type	1222	TnxBaseIndicesDescriptorClass type 1224
TnxBaseDirectTransport class	729	TnxBaseKeyDescriptor class 748 Destroy 748 KeyLen 749 LoadFromReader 749 SaveToWriter 749
TnxBaseEngineExtender class	729	TnxBaseLog class 750 Clear 750 Enabled 753 Flush 751 WriteBlock 751 WriteLogData 751 WriteString 752 WriteStrings 752
TnxBaseEngineExtenderClass type	1223	TnxBasePluginCommandHandler class 753 CommandHandler 754 UISettingsVisible 753
TnxBaseEngineMonitor class	731	TnxBasePooledTransport class 754 Broadcast 755 CallbackThreadCount 762 CallbackThreadPriority 762 CompressLimit 762 CompressType 763 ConcurrentIOPCThreads 763 ConnectionCount 756 Create 755 Destroy 755 EstablishConnection 756 GetConfigSettings 756 GetNetIdle 757 GetStatsCaptions 757 GetStatsValues 758 HeartbeatInterval 763 HeartbeatThreadPriority 764 IsConnected 758
TnxBaseFieldsValidationDescriptor class	734	
TnxBaseFieldsValidationDescriptorClass type	1223	
TnxBaseFieldValidationDescriptor class	734	
		ApplyAt 736 CheckValid 735 GetList 735 IsEqual 735 LoadFromReader 736 Name 737 SaveToWriter 736
TnxBaseFieldValidationDescriptorClass type	1223	
TnxBaseHeader record	1190	
TnxBaseHeapDescriptor class	737	
		AddBlockHeapDescriptor 738 BlockHeapDescriptor 740 CheckValid 738 Destroy 737 HeapEngine 740 IsEqual 739 LoadFromReader 739 RemoveBlockHeapDescriptor 739 SaveToWriter 740
TnxBaseHeapDescriptorClass type	1224	
TnxBaseIndicesDescriptor class	741	
		AddIndex 742

TnxBasePooledTransport class	754		
LoadConfig	758	Password	778
LoadSettingsFromStream	759	SaveSettings	774
OverlappedClient	764	SecureServer	778
Port	764	SecurityMonitor	779
Post	759	ServerConfiguration	779
Request	759	StartStoppedModules	774
SaveConfig	760	StopStartedModules	775
SaveSettingsToStream	760	Username	779
ServerThreadPriority	765		
SetNetIdle	761	TnxBaseServerConfiguration class	780
TerminateConnection	761	Create	780
WatchdogInterval	765	EnsureComponentConfiguration	781
WatchdogThreadPriority	765	GetComponentConfiguration	781
Write	761		
TnxBasePooledTransportClass type	1224	TnxBaseServerEngine class	781
TnxBaseRecordCompressionDescriptor class	766	ClearStats	783
IsEqual	766	Create	782
LoadFromReader	766	CursorCount	785
RecordCompressionEngine	767	DatabaseCount	785
SaveToWriter	767	Destroy	782
TnxBaseRecordCompressionDescriptorClass type	1225	GetStatsCaptions	783
TnxBaseRecordDescriptor class	768	GetStatsValues	783
CheckValid	768	ServerGuid	786
Destroy	768	SessionCount	786
FileNumber	770	SessionOpen	784
HeaderSection	770	SessionOpenEx	784
IsEqual	769	StatementCount	786
LoadFromReader	769	TransContextCount	787
RecordEngine	770		
SaveToWriter	769	TnxBaseSession class	787
TnxBaseRecordDescriptorClass type	1225	AbstractSession	802
TnxBaseServer class	771	AddAlias	788
ActivateAll	772	AddAliasEx	789
Authenticate	772	BeepOnLoginError	802
AutoSaveConfig	776	CancelProcessing	789
ClearAllStats	773	CloseInactiveFolders	790
Create	771	CloseInactiveTables	790
DeActivateAll	773	Create	787
Destroy	772	CurrentUserName	802
IsServerActive	776	DatabaseCount	803
LoadSettings	773	Databases	803
LoginTries	777	Default	804
MaxUserCount	777	DefaultDatabase	790
OnError	775	DefaultSession	791
OnLoginError	777	DefaultTransContext	791
OnShowLogin	775	DeleteAlias	791
		DeleteAliasEx	792
		Destroy	788
		FindDatabase	792
		GetAliasNames	793
		GetAliasNamesEx	793

TnxBaseSession class 787
 GetAliasPath 794
 GetAliasPathEx 794
 GetChangedTables 795
 GetRegisteredClassList 795
 GetStoredProcNames 796
 GetStoredProcParams 796
 GetTableNames 797
 IsAlias 797
 IsConnected 798
 ModifyAlias 798
 ModifyAliasEx 799
 OnLogin 801
 OpenDatabase 799
 Password 804
 PasswordAdd 800
 PasswordRemove 800
 PasswordRemoveAll 800
 PasswordRetries 804
 ServerEngine 805
 ServerVersion 805
 TableExists 801
 Timeout 805
 TransContextCount 806
 TransContexts 806
 UserName 806

TnxBaseSessionPool class 807
 AcquireSession 808
 Create 807
 Destroy 808
 Password 808
 ServerEngine 809
 Timeout 809
 UserName 809

TnxBaseSetting class 810
 DefaultValue 810
 EnforceValues 811
 Hint 811
 Max 811
 Min 811
 Name 812
 PropertyName 812
 SettingType 812
 ValueList 813

TnxBaseSettings class 813
 AddSetting 813
 Count 814
 Setting 814

TnxBaseStreamDescriptor class 814
 HeaderSection 816
 isEqual 815
 LoadFromReader 815
 SaveToWriter 816
 StreamEngine 816

TnxBaseStreamDescriptorClass type 1225

TnxBaseTableDescriptor class 817
 AddAutoIncDescriptor 818
 AddBlobDescriptor 818
 AddIndicesDescriptor 819
 AddRecordDescriptor 819
 AssignOnlyFields 820
 AutoIncDescriptor 826
 BlobDescriptor 827
 BookmarkSize 827
 CheckValid 820
 Clear 820
 Create 817
 CustomDescsDescriptor 827
 Destroy 817
 EnsureIndicesDescriptor 821
 FieldsDescriptor 828
 FindRelatedDescriptorOfType 821
 GetChild 821
 GetConstraintByName 822
 GetConstraints 822
 GetFieldFromName 822
 GetIndexFromName 823
 IndicesDescriptor 828
 isEqual 823
 IsIndexDescValid 824
 KeyLen 828
 LoadFromReader 824
 RecordDescriptor 829
 RemoveAutoIncDescriptor 824
 RemoveBlobDescriptor 825
 RemoveIndicesDescriptor 825
 RemoveRecordDescriptor 825
 SaveToWriter 826
 TablesDescriptor 829

TnxBaseTableDescriptorClass type 1226

TnxBaseTransport class 829
 ActualBytesReceived 844
 ActualBytesSent 844
 BeginBroadcast 831
 Broadcast 831
 btDoCallBack 832

TnxBaseTransport class	829	Timeout	850
ClearStats	832	UncompressedBytesReceived	850
CommandHandler	845	UncompressedBytesSent	851
CompressTime	845	UncompressTime	851
ConnectionCount	833	Write	841
Create	830	TnxBaseTransportClass type	1226
CurrentTransport	833	TnxBaseTransportWrapper class	851
Destroy	830	TnxBaseUpdateHandlerClass type	1227
EstablishConnection	833	TnxBaseUpdateObject class	852
EventLogOptions	845	TnxBatchAppendBlobStream class	852
GetBoundAddresses	834	Destroy	853
GetConfigSettings	834	Read	853
GetName	835	Seek	853
GetServerNames	835	Truncate	854
GetSessionCallback	835	Write	854
GetStatsCaptions	836	TnxBlobCopyMode enumeration	1191
GetStatsValues	836	TnxBlobStream class	855
IsConnected	836	ChunkSize	858
KeepStats	846	Destroy	855
LoadConfig	837	Read	856
LoadSettingsFromStream	837	Seek	856, 857
MaxBytesPerSecond	846	Truncate	857
MessagesReceived	846	Write	857
MessagesSent	847	TnxBlockModeBlobStream class	858
Mode	847	Read	859
MsgCount	847	Seek	859
OnAddSession	842	Truncate	860
OnChooseServer	843	Write	860
OnConnectionLost	843	TnxBlockSignature type	1227
OnFindServers	843	TnxBookmark record	1192
OnRemoveSession	844	TnxBroadcastReply class	861
PingTime	847	Create	861
Post	837	TnxCachedDataSet class	862
ProtocolName	838	AddIndexDefs	863
Reply	838	BatchSize	864
Request	839	Create	862
ResetMsgCount	839	Destroy	863
RespondToBroadcasts	848	FieldDefs	864
SaveConfig	840	IndexDefs	864
SaveSettingsToStream	840	IndexFieldCount	865
ServerGUID	848	IndexFieldNames	865
ServerName	848	IndexFields	865
ServerNameDesigntime	849	IndexName	866
ServerNameRuntime	849	KeyExclusive	866
Sleep	840	KeyFieldCount	866
TerminateConnection	841	KeyPartialLen	867
ThreadPriorityMax	849	MasterFields	867
ThreadPriorityMin	850	MasterSource	867

TnxCachedDataSet class 862
OnCreateTable 863
Options 868
ReadOnly 868
SourceDataSet 868
TableName 869
UpdateObject 869
UsedTableName 869
TnxCachedDataSetOption enumeration 1193
TnxCachedDataSetOptions type 1227
TnxChooseServerEvent type 1228
TnxClassListType enumeration 1193
TnxCompKeyDescriptor class 870
Add 870
CheckValid 871
Clear 871
Delete 871
Destroy 870
IsEqual 872
KeyFieldCount 873
KeyFields 873
LoadFromReader 872
SaveToWriter 873
TnxComponent class 874
BeforeDestruction 875
ClearStats 875
Create 874
Destroy 875
DisplayCategory 880
DisplayName 881
FindClassRecursive 876
FreeInstance 876
GetConfigSettings 876
GetStatsCaptions 877
GetStatsValues 877
IterateDependents 877
LoadConfig 878
LoadSettingsFromStream 878
NewInstance 878
SaveConfig 879
SaveSettingsToStream 879
Supported 879
UISettingsVisible 880
UIStatsVisible 880
Version 881
TnxConnectionLostEvent type 1228
TnxConstDefaultValueDescriptor 168
TnxConstDefaultValueDescriptor class 881
AsVariant 883
IsEqual 882
LoadFromReader 882
SaveToWriter 882
TnxCreateTableEvent type 1229
TnxCurrentDateTimeDefaultDescriptor 168
TnxCurrentDateTimeDefaultDescriptor class 883
TnxCurrentUserDefaultDescriptor class 883
TnxCursor class 884
Create 884
Database 885
TnxCursorClass type 1229
TnxCursorID type 1229
TnxCustomConnectionFilter class 885
OnAcceptConnection 885
TnxCustomDescsDescriptor class 886
AddCustom 887
CheckValid 887
Clear 888
Create 886
Destroy 887
GetCustomDescriptorFromName 888
IsEqual 888
LoadFromReader 889
RemoveCustom 889, 890
SaveToWriter 890
TnxCustomDescsDescriptorClass type 1230
TnxDataAccessStateComponent class 890
TnxDatabase class 891
AliasName 914
AliasPath 914
BackupTable 892
BackupTableEx 892
ChangePassword 893
ChangePasswordEx 893
Commit 894
Create 891
CreateTable 894
CreateTableEx 895
DataSetCount 915
DataSets 915
Default 915
DeleteTable 896
Destroy 892
EmptyTable 896
Exclusive 916
ExecQuery 897

TnxDatabase class	891	CheckValid	922
ExecStoredProcedure	897	Create	920
Exists	897	Destroy	920
FailSafe	916	FilesDescriptor	927
GetAutoIncValue	898	FindRelatedDescriptorOfType	923
GetChangedTables	898	GetFileFromExt	923
GetDataDictionary	899	IsEqual	923
GetDataDictionaryEx	899	MakeReadOnly	924
GetFreeDiskSpace	900	ReadFromReader	924
GetFreeDiskSpaceEx	900	ReadFromStream	925
GetPath	900	RemoveStreamDescriptor	925
GetStoredProcNames	901	StreamDescriptor	927
GetStoredProcParams	901	UsableBlockSize	925
GetTableNames	902	Version	927
Implicit	916	WriteToStream	926
InTransaction	917	WriteToWriter	926
OpenQuery	902	TnxDataMessage record	1194
OpenStoredProcedure	903	TnxDataMessageReader class	928
OpenTable	903	Create	928
PackTable	904	DataMessageStream	929
PackTableEx	904	Destroy	928
Path	917	TnxDataMessageStream class	929
Queries	917	Create	930
QueryCount	918	TnxDataset class	930
ReadOnly	918	_Dictionary	950
RecoverTable	905	AbstractCursor	950
RecoverTableEx	905	ActiveDesigntime	951
RelIndexTable	906	ActiveRuntime	951
RelIndexTableEx	907, 908	AddFileBlob	931
RenameTable	908	AddFileBlobEx	932
RestructureTable	909	AfterCancel	951
RestructureTableEx	910	AfterClose	952
Rollback	910	AfterConstruction	932
StartTransaction	911	AfterDelete	952
StartTransactionWith	911	AfterEdit	952
StartTransactionWithEx	912	AfterInsert	953
TableCount	918	AfterOpen	953
TableExists	913	AfterPost	953
Tables	919	AfterRefresh	954
TransactionCorrupted	913	AfterScroll	954
TransContext	919	AliasName	954
TryStartTransaction	913	AutoCalcFields	955
TnxDatabaseID type	1230	BatchPost	932
TnxDataDictionary class	920	BeforeCancel	955
AddStreamDescriptor	921	BeforeClose	955
Assign	921	BeforeDelete	955
AssignOnlyFields	922	BeforeEdit	956
CheckReadOnly	922	BeforeInsert	956

TnxDataset class 930
BeforeOpen 956
BeforePost 957
BeforeRefresh 957
BeforeScroll 957
BeginBatchAppend 933
BlockReadOptions 958
BookmarkValid 933
CompareBookmarks 934
CopyRecords 934
CopyRecordsEx 935
Create 930
CreateBlobStream 935
Database 958
DeleteRecords 936
DeleteStream 936
Destroy 931
EndBatchAppend 937
Exists 937
FieldsDescriptor 958
Filter 959
Filtered 959
FilterOptions 959
FilterResync 960
FilterTimeout 960
FilterType 960
FlipOrder 961
FlushBatchAppend 937
GetAutoIncValue 938
GetBookmark 938
GetCurrentRecord 939
GetFieldData 939
GetStreamList 939
GotoCurrent 940
InBatchAppend 940
IsRecordLocked 941
IsSequenced 941
IsTableLocked 941
KeyAsVariant 942
Locate 942
LockTable 942
Lookup 943
OnCalcFields 961
OnDeleteError 961
OnEditError 962
OnFilterRecord 962
OnNewRecord 962
OnPostError 962
OnServerFilterTimeout 950
Options 963
Post 943
PSEndTransaction 944
PSEExecuteStatement 944
PSGetQuoteChar 944
PSInTransaction 945
PSIsSqlBased 945
PSIsSqlSupported 945
PSReset 945
PSStartTransaction 946
ReadStream 946
ReadStreamEx 947
RecordCountAsync 947
Session 963
SetAutoIncValue 948
SimpleExpressionFilterClass 963
SqlFilterClass 964
TableDescriptor 964
Timeout 964
UnlockTable 948
UnlockTableAll 949
Version 965
WriteStream 949
TnxDataSetInternalState enumeration 1194
TnxDataSetInternalStates type 1230
TnxDataSetOption enumeration 1195
TnxDataSetOptions type 1231
TnxDictionaryItem class 965
 Assign 966
 CheckValid 966
 ConstraintName 970
 CustomStrings 970
 Destroy 965
 FindDescriptorClass 966
 FindParentOfType 967
 FindRelatedDescriptorOfType 967
 GetParentOfType 968
 GetRelatedDescriptorOfType 968
 HasCustomStrings 970
 ID 971
 isEqual 968
 LoadFromReader 969
 SaveToWriter 969
TnxDictionaryItemClass type 1231
TnxDirectKeySetFieldsMethod type 1232
TnxEmptyDefaultValueDescriptor 168
TnxEmptyDefaultValueDescriptor class 971

TnxEngineAction enumeration	1195	FieldByNameAsVariant	998
TnxEngineActions type	1232	fsdFieldChangeNotification	988
TnxExtendableServerObject class	971	GetFieldForFilter	988
AfterConstruction	973	GetFieldFromName	989
BeforeDestruction	973	GetFields	989
Create	972	GetRecordField	990
Destroy	972	GetRecordFieldForFilter	990, 991
esoOptionClear	973	HasAutoIncField	991
esoOptionGetEffective	974	HasBlobFields	992
esoOptionSet	974	HasRecRevField	992
GetExtenderOfType	974	HasSameFields	992
NotifyExtenders	975	InitRecord	993
OptionClear	975	InsertFieldAt	993
OptionGet	976	IsEqual	994
OptionGetEffective	976	IsRecordFieldNull	994
OptionList	976	LoadFromReader	995
OptionListEffective	977	LogicalRecordLength	999
OptionSet	977	MoveField	995
TnxExtTextKeyFieldDescriptor class	977	RecordLength	999
LoadFromReader	978	RemoveField	995
SaveToWriter	978	RemoveValidations	996
TnxFieldDescriptor class	979	SaveToWriter	996
AddDefaultValue	979	SetRecordField	996
AddValidations	980	SetRecordFieldNull	997
BufferAsVariant	984	UpdateLogRecLenAlign	997
CheckValid	980	UpdateSetupAndOffsets	998
Destroy	979	Validations	999
EnsureValidations	980	TnxFieldsDescriptorClass type	1233
FindRelatedDescriptorOfType	981	TnxFieldsValidationsDescriptor class	1000
IsEqual	981	AddValidation	1000
LoadFromReader	981	TnxFieldsValidationsDescriptorClass type	1233
Name	984	TnxFieldType enumeration	1197
Number	984	TnxFieldTypes type	1233
RemoveDefaultValue	982	TnxFieldValidationsDescriptor class	1000
RemoveValidations	982	AddValidation	1002
SaveToWriter	982	CheckValid	1002
SetupField	983	Clear	1002
UpdateSetup	983	Create	1001
TnxFieldDescriptorClass type	1232	Destroy	1001
TnxFieldsDescriptor class	985	GetValidationDescriptorFromName	1003
AddField	986	IsEqual	1003
AddValidations	987	LoadFromReader	1003
CheckValid	987	RemoveValidation	1004
Clear	987	SaveToWriter	1005
Create	985	TnxFieldValidationsDescriptorClass type	1234
Destroy	986	TnxFileDescriptor class	1005
EnsureValidations	988	BlockSize	1007
FieldByIndexAsVariant	998	BlockSizeBytes	1007

TnxFileDescriptor class 1005
 CheckValid 1005
 Desc 1008
 Extension 1008
 GrowSize 1008
 InitialSize 1009
 IsEqual 1006
 LoadFromReader 1006
 Number 1009
 SaveToWriter 1006
 UsableBlockSize 1009
TnxFileDescriptorClass type 1234
TnxFilesDescriptor class 1010
 AddFile 1011
 CheckValid 1011
 Clear 1012
 Create 1010
 Destroy 1010
 GetFileFromExt 1012
 InsertFileAt 1012
 IsEqual 1013
 LoadFromReader 1013
 MoveFile 1014
 RemoveFile 1014
 SaveToWriter 1015
TnxFilesDescriptorClass type 1234
TnxFilterID type 1235
TnxFilterType enumeration 1198
TnxFindServersEvent type 1235
TnxGetServerNamesHandler class 1015
 Create 1015
 Destroy 1016
 GetServerNames 1016
TnxGrowSize type 1236
TnxHeapBlobDescriptor class 1017
 AddHeapDescriptor 1018
 CheckValid 1018
 Create 1017
 Destroy 1017
 HeapDescriptor 1020
 IsEqual 1018
 LoadFromReader 1019
 RemoveHeapDescriptor 1019
 SaveToWriter 1020
TnxHeapBlobDescriptorClass type 1236
TnxHeapRecordDescriptor class 1020
 AddHeapDescriptor 1021
 AddRecordCompressionDescriptor 1022
 CheckValid 1022
 Destroy 1021
 HeapDescriptor 1024
 IsEqual 1022
 LoadFromReader 1023
 RecordCompressionDescriptor 1025
 RemoveHeapDescriptor 1023
 RemoveRecordCompressionDescriptor 1023
 SaveToWriter 1024
TnxHeapRecordDescriptorClass type 1236
TnxIndexDataSet class 1025
 ApplyRange 1026
 Cancel 1026
 CancelRange 1027
 Create 1025
 Destroy 1026
 EditKey 1027
 EditRangeEnd 1027
 EditRangeStart 1028
 FindKey 1028
 FindNearest 1029
 GetIndexNames 1030
 GotoKey 1030
 GotoNearest 1030
 GotoNearestBackward 1031
 Post 1031
 SetKey 1031
 SetRange 1032
 SetRangeEnd 1033
 SetRangeShared 1033, 1034
 SetRangeSimple 1034
 SetRangeStart 1034
TnxIndexDescriptor class 1035
 CheckValid 1036
 CreateStandalone 1035
 CreateStandaloneFromReader 1036
 Desc 1038
 Destroy 1036
 Dups 1038
 FileNumber 1039
 IndexEngine 1039
 IndexFile 1039
 IsEqual 1037
 KeyDescriptor 1040
 LoadFromReader 1037
 Name 1040
 Number 1040
 SaveStandalone 1037

TnxIndexDescriptor class	1035	CheckValid	1054
SaveToWriter	1038	CompareCodepage	1056
TnxIndexDescriptorClass type	1237	Flags	1056
TnxIndexPathPosition enumeration	1199	IgnoreKanaType	1056
TnxINIComponentConfiguration class	1041	IgnoreNonSpace	1057
Create	1041	IgnoreSymbols	1057
Destroy	1041	IgnoreWidth	1057
GetValue	1042	isEqual	1055
GetValueStream	1042	LoadFromReader	1055
SetValue	1042	Locale	1058
SetValueStream	1043	OverrideStorageCodepage	1058
TnxINIServerConfiguration class	1043	SaveToWriter	1055
Create	1044	StorageCodepage	1058
Destroy	1044	UseStringSort	1059
EnsureComponentConfiguration	1044	TnxLocaleDescriptorClass type	1238
Flush	1045	TnxLocalizedDictionaryItem class	1059
GetComponentConfiguration	1045	AddLocaleDescriptor	1060
TnxIPv4Transport class	1045	CheckValid	1060
ProtocolName	1046	Destroy	1059
TnxIterationList class	1046	FindRelatedDescriptorOfType	1061
Add	1047	isEqual	1061
LoadSettingsFromStream	1047	LocaleDescriptor	1062
SaveSettingsToStream	1047	RemoveLocaleDescriptor	1061
TnxIterationListClass type	1237	UsedLocaleDescriptor	1062
TnxKeyAsVariantField class	1048	UsedStorageCodePage	1063
Create	1048	TnxLockConflictType enumeration	1201
TnxKeyBuffer record	1199	TnxLockedOptionsExtendableServerObject class	
TnxKeyDescriptorClass type	1237	1063	
TnxKeyFieldDescriptor class	1048	Destroy	1063
Ascend	1051	esoOptionClear	1064
CheckValid	1049	esoOptionSet	1064
Desc	1051	TnxLockPresent enumeration	1201
Destroy	1049	TnxLockRequestType enumeration	1201
Field	1051	TnxLockResult enumeration	1202
FieldNumber	1052	TnxLockType enumeration	1202
isEqual	1049	TnxLoggableComponent class	1064
KeyChars	1052	EventLog	1066
KeyFieldEngine	1052	EventLogEnabled	1067
KeyLength	1053	IcLog	1065, 1066
LoadFromReader	1050	TnxLoginCallback type	1239
NullBehaviour	1053	TnxLoginEvent type	1239
OverrideLengthBytes	1053	TnxLogPriorities type	1240
OverrideLengthChars	1054	TnxLogPriority enumeration	1203
SaveToWriter	1050	TnxMainIndicesDescriptor class	1067
TnxKeyFieldDescriptorClass type	1238	CheckValid	1067
TnxKeyFilterID type	1238	Clear	1068
TnxKeyIndex enumeration	1200	DefaultIndex	1069
TnxLocaleDescriptor class	1054	isEqual	1068

TnxMainIndicesDescriptor class 1067
 LoadFromReader 1069
 SaveToWriter 1069
TnxMainIndicesDescriptorClass type 1240
TnxMappingMethod type 1240
TnxMemoBlobStream class 1070
 Destroy 1070
 Read 1071
 Seek 1071
 Truncate 1071
 Write 1072
TnxMemoField class 1072
 AsWideString 1073
TnxMemTable class 1073
 Create 1073
 Destroy 1074
 FieldDefs 1075
 IndexDefs 1075
 IndexFieldCount 1075
 IndexFieldNames 1076
 IndexFields 1076
 IndexName 1076
 KeyExclusive 1076
 KeyFieldCount 1077
 KeyPartialLen 1077
 MasterFields 1077
 MasterSource 1078
 OnCreateTable 1074
 StoreDefs 1078
 TableName 1078
 UsedTableName 1079
TnxMessageHeaderFlag enumeration 1203
TnxMessageHeaderFlags type 1241
TnxMinMaxValidationDescriptor class 1079
 AddKeyField 1080
 CheckValid 1080
 Destroy 1080
 IsEqual 1081
 KeyField 1082
 LoadFromReader 1081
 Max 1082
 MaxAsVariant 1083
 Min 1083
 MinAsVariant 1083
 RemoveKeyField 1081
 SaveToWriter 1082
TnxNestedTableDescriptor class 1084
 Create 1084
 Name 1084
TnxNestedTableDescriptorClass type 1241
TnxNetIdleEvent type 1242
TnxNoChangeValidationDescriptor class 1085
 AddKeyField 1086
 CheckValid 1086
 Destroy 1085
 IsEqual 1086
 KeyField 1088
 LoadFromReader 1087
 RemoveKeyField 1087
 SaveToWriter 1087
TnxNotNullCompKeyDescriptor class 1088
 CheckValid 1088
 LoadFromReader 1089
 SaveToWriter 1089
TnxNullActiveBroadcast class 1089
TnxNullBehaviour enumeration 1204
TnxOnAcceptConnection type 1242
TnxOpenMode enumeration 1204
TnxParamType enumeration 1205
TnxPersistent class 1090
 FreeInstance 1090
 NewInstance 1091
TnxPersistentClass type 1242
TnxPooledSession class 1091
 AddRef 1092
 Destroy 1091
 Release 1092
 SessionPool 1092
TnxQuery class 1093
 Create 1093
 Destroy 1093
 ExecSQL 1094
 ParamCheck 1094
 SQL 1094
 Text 1095
TnxRecordCountOption enumeration 1205
TnxRecordGetBatchExOption enumeration 1206
TnxRecordGetBatchExOptions type 1243
TnxRefKeyDescriptor class 1095
 LoadFromReader 1095
 SaveToWriter 1096
TnxRegisterableComponent class 1096
 FindRegisteredClass 1097
 GetRegisteredClasses 1097
TnxRegisterableComponentClass type 1243
TnxRemoveSessionEvent type 1243

TnxReplyCallback type	1244		ErrorMessage	1112
TnxSearchKeyAction enumeration	1206		isEqual	1110
TnxServer class	1097		LoadFromReader	1111
CommandHandler	1098		SaveToWriter	1111
EnabledModules	1098		TnxSqlParamDesc record	1209
EventLog	1098		TnxSqlParamList type	1246
EventLogEnabled	1099		TnxSqlUpdateObject class	1112
NamedPipeTransport	1099		Create	1113
SecurityMonitor	1099		DeleteSql	1113
ServerEngine	1100		Destroy	1113
ServerName	1100		InsertSql	1114
SQLEngine	1100		ModifySql	1114
WinsockTransport	1101		Params	1114
TnxServerFilterTimeoutEvent type	1244		Sql	1115
TnxServerManager class	1101		TnxState enumeration	1210
ServerEngine	1101		TnxStateComponent class	1115
TnxServerModule enumeration	1207		Active	1121
TnxServerModules type	1245		ActiveDesigntime	1121
TnxSession class	1102		ActiveRuntime	1121
Destroy	1102		BeforeDestruction	1116
ServerEngine	1102		Close	1117
TnxSessionOwnedDataAccessStateComponent class			Connect	1117
1103			Connected	1122
AfterConstruction	1104		Create	1116
Create	1103		Destroy	1116
Session	1104		Enabled	1122
Timeout	1104		GetConfigSettings	1117
TnxSessionPool class	1105		GetStatsCaptions	1118
Destroy	1105		GetStatsValues	1118
ServerEngine	1105		LoadConfig	1118
TnxSetKeyOption enumeration	1207		LoadSettingsFromStream	1119
TnxSetKeyOptions type	1245		OnStateChanged	1122
TnxSetting class	1106		OnStateTransChanged	1123
Assign	1106		Open	1119
TnxSettings class	1106		SaveConfig	1119
AddSetting	1108		SavedActive	1123
Create	1107		SaveSettingsToStream	1120
Destroy	1107		StartTime	1123
TnxSettingType enumeration	1208		State	1124
TnxShareMode enumeration	1208		StateTransition	1124
TnxSimpleLock class	1108		SwitchToSavedActive	1120
Create	1108		UIStateVisible	1120
Destroy	1109		TnxStatementDataSet class	1124
Failed	1109		Create	1125
Lock	1109		DataSource	1126
Unlock	1110		Destroy	1125
TnxSqlCheckValidationDescriptor class	1110		Log	1127
Constraint	1112		ParamByName	1125

TnxStatementDataSet class	1124	
ParamCount	1127	
Params	1127	
Prepare	1126	
Prepared	1128	
ReadOnly	1128	
RecordsRead	1128	
RequestLive	1129	
RowsAffected	1129	
Unprepare	1126	
TnxStatementExecDirectPhase enumeration	1210	
TnxStatementID type	1246	
TnxStatementType enumeration	1211	
TnxStateTransition enumeration	1211	
TnxStoredProc class	1129	
ExecProc	1130	
Prepare	1130	
RefreshParam	1130	
StoredProcName	1131	
TnxTable class	1131	
AddIndex	1132	
AddIndexEx	1133	
ChangePassword	1133	
Create	1131	
CreateTable	1134	
CreateTableEx	1134	
DeleteIndex	1134	
DeleteTable	1135	
Destroy	1132	
EmptyTable	1135	
Exclusive	1140	
Exists	1135	
FieldDefs	1140	
IndexDefs	1140	
IndexFieldCount	1141	
IndexFieldNames	1141	
IndexFields	1141	
IndexName	1142	
KeyExclusive	1142	
KeyFieldCount	1142	
KeyPartialLen	1143	
MasterFields	1143	
MasterSource	1143	
OnCreateTable	1139	
PackTable	1136	
PackTableEx	1136	
Password	1143	
ReadOnly	1144	
RelIndexTable	1136, 1137	
RelIndexTableEx	1137, 1138	
RenameTable	1138	
RestructureTable	1139	
RestructureTableEx	1139	
StoreDefs	1144	
TableName	1144	
WriteOnly	1145	
TnxTableDescriptor class	1145	
TnxTableScope enumeration	1212	
TnxTablesDescriptor class	1145	
AddTable	1147	
CheckValid	1147	
Clear	1147	
Create	1146	
Destroy	1146	
GetTableDescriptorFromName	1148	
GetTableIndexFromName	1148	
GetTables	1148	
IsEqual	1149	
LoadFromReader	1149	
RemoveTable	1149, 1150	
SaveToWriter	1150	
TnxTablesDescriptorClass type	1246	
TnxTaskID type	1247	
TnxTaskStatus record	1212	
TnxTextKeyFieldDescriptor class	1151	
IgnoreCase	1152	
IsEqual	1151	
LoadFromReader	1151	
SaveToWriter	1152	
TnxThreadWithDatabase class	1153	
Create	1153	
Database	1153	
TnxThreadWithSession class	1154	
Create	1154	
Session	1155	
TnxThreadWithTable class	1155	
Create	1155	
Table	1156	
TnxTransContext class	1156	
Commit	1157	
Create	1157	
DatabaseCount	1161	
Databases	1161	
Default	1162	
Destroy	1157	
FailSafe	1162	

TnxTransContext class 1156
 InTransaction 1162
 Rollback 1158
 StartTransaction 1158
 StartTransactionWith 1159
 StartTransactionWithEx 1159
 TransactionCorrupted 1160
 TryStartTransaction 1160
 TnxTransContextID type 1247
 TnxTransID type 1247
 TnxTransportLogOption enumeration 1213
 TnxTransportLogOptions type 1248
 TnxTransportMode enumeration 1213
 TnxWinsockTransport class 1163
 BeginBroadcast 1164
 BroadcastThreadPriority 1167
 CallbackThreadCount 1167
 CallbackThreadPriority 1167
 ConnectionFilter 1168
 Create 1163
 Destroy 1164
 GetBoundAddresses 1164
 GetConfigSettings 1165
 GetName 1165
 ListenAddresses 1168
 ListenThreadPriority 1168
 LoadConfig 1165
 LoadSettingsFromStream 1166
 SaveConfig 1166
 SaveSettingsToStream 1166
 TO 404, 469
 TOP 551
 TOSTRING 472
 ToString Function 472
 TOSTRINGLEN 473
 ToStringlen Function 473
 TRAILING 470
 TRANSACTION 568
 Transaction Statements
 About Transaction Statements 568
 COMMIT statement 570
 ROLLBACK statement 571
 START TRANSACTION statement 568
 Transfer Type 206
 transport statistics 122
 transport type 150
 TRIGGER
 CREATE TRIGGER statement 529

DROP TRIGGER statement 547
 TRIM 470
 Trim Function 470
 TRUE 406, 413, 438
 TRY 589
 TRY statement 589
 TYPE 426

- U -

UDF 118
 UNION 551
 UNIQUE 492, 493, 515, 526
 Unique Predicate 492
 UNKNOWN 406, 413, 438
 UNTIL 586
 UPDATE 515, 529, 562
 UPDATE statement 562
 UPDATE statements 178
 UPDATING 497
 Updating Predicate 497
 UPPER 470
 USE 426
 USE STRING SORT 426
 USER 407, 478, 515
 User Defined Functions 118
 User-defined function 536
 User-defined functions 501
 User-defined procedure 532
 User-defined procedures 501
 USING 446, 449, 465, 469, 551
 Using SQL 178
 Using the Importer Classes in own applications 214

- V -

Value Expressions

About Value Expressions 407
 Boolean Value Expressions 413
 Case Expressions 419
 Cast Specification 422
 DateTime Value Expressions 412
 Numeric Value Expressions 408
 Row Value Expressions 416
 String Value Expressions 410
 Subqueries 417
 VALUES 560

VARCHAR 431
variable length 171
VARYING 431
VIEW
 CREATE VIEW statement 528
 DROP VIEW statement 546
View Dictionary 157
View Log 181
Visual Style 147, 148

- W -

WHEN 419, 529
WHERE 551, 562, 564
WHILE 587
WHILE statement 587
WIDTH 426
WinNT Service 125
WITH 515
WITH DATA 515
WORD 435
WORK 570, 571

- Y -

YEAR 404, 469, 477
Year-Month Literal 404

Endnotes 2... (after index)

Back Cover