Enhanced TTable Control

The enhanced ttable control (TTableEnhanced) which ships with BDE2.51, is an example of how the BDE can be used to create a Delphi control. This control offers functionality beyond the ttable control.

Please select one of the following:

TTableEnhanced.BlockSize Property

Example

Function: Sets the block size in number of records.

Description: This number is used in conjunction with WriteBlock to determine the amount of records inserted per call to WriteBlock. i.e. If BlockSize is set to 100 and BlockTotal is set to 1000, WriteBlock should be called 10 times to insert the amount of records determined by BlockTotal. BlockSize should be set to 1 then using the InsertFast or AppendFast method for speed and memory consumption reasons. The BlockTotal setting is disregarded.

TTableEnhanced.BlockTotal Property

Example

Function: Sets the total amount of records to be inserted using WriteBlock

Description: This number is used in conjunction with BlockSize to determine how WriteBlock should insert records. Set BlockTotal to the total amount of records to be placed into the table. Set BlockSize to the amount of records to be placed into the table with each WriteBlock call. For example: If an application was to write 10,000 records to a table, 500 records at a time, Set BlockTotal to 10000 and BlockSize to 500.

TTableEnhanced Overview

TTableEnhanced offers an extended set of functions beyond the normal TTable. **NOTE**: TTableEnhanced can replace a TTable without any code change to an existing program. All the original TTable functionality still exists because TTableEnhanced is derived from TTable. TTableEnhanced offers greatly improved exception handling.

With this control you can get IDAPI system information, network information, session information, and much more using the following methods:
GetSysVersion, GetSysInfo, GetSysConfig, IsTableShared, IsTableLocked, and IsRecordLocked

The InsertFast method enables the user to insert a single record with almost twice the speed of the TTable's Insert method. It also allows you to lock the record, using the InsertMode property, that is being inserted.until the application unlocks it using the ReleaseRecordLock method. The AppendFast method also gives a performance increase over the Append method. The WriteBlock method lets the application transfer up to 64k of record information at one time. For example, by setting the BlockSize and BlockTotal properties, an application can fill up a buffer that holds 100 records worth of information and then transfer this to the table. This is good for tables that need to insert multiple record at one time. It gives an increase in performance over the InsertFast method.

The Pack method removes records in a dBase or Paradox table marked for deletion. This reduces the size of the table and updates the indexes giving increased performance.

The CopyTable method is different than the BatchMove method. It performs a DOS copy on local tables and indexes rather than a record by record move of the table and indexes. Performance is greatly improved. It can only be used to copy tables of identical type. If the table is remote, indexes are not copied. Setting the Overwrite property gives the CopyTable method rights to overwrite an existing table.

Other enhancements include: SaveToDisk and SaveWhenIdle methods. SaveToDisk saves the record to disk after every dbi operation. Usually records are placed into a buffer that is written when the buffer is full or the table is closed. This option will greatly decrease the performance of table operations, but will make sure that the table is updated instantly. SaveWhenIdle saves the buffer to disk when the application is idle. This is recommended. If a power failure ocurred, most or all of the table's operations would be saved to disk.

TTableEnhanced Function Reference

The following methods and properties are enhancements over TTable which are included in TTableEnhanced:

METHODS:

- AppendFast
- WriteBlock
- CopyTable
- GetSysConfig
- GetSysInfo
- GetSysVersion
- InitBooleanField
- InitCurrencyField
- InitDateField
- InitFloatField
- InitIntegerField
- InitSmallintField
- InitStringField
- InitTimeField
- InitTimeStampField
- InitializeBuffer
- InsertFast
- IsRecordLocked
- IsTableLocked
- IsTableShared
- NextRecord
- Pack
- ReleaseRecordLock

PROPERTIES:

- BlockTotal
- BlockSize
- InsertMode
- Overwrite
- SaveToDisk
- SaveWhenIdle

TTableEnhanced.WriteBlock Method

Example

Function: Writes a block of records to a table.

Description: This method give the application functionality to write a set of records to a table with one WriteBlock call. This is different from AppendFast or InsertFast methods because these methods only insert or append one record. WriteBlock will insert up to 64k of records into a table. It employes greater exception handling, along with faster record insertion capabilities. Set BlockTotal to the total amount of records to be inserted. Set BlockSize to the amount of records to insert with each WriteBlock call. Initialize the record buffer before calling WriteBlock. After each record buffer is fiilled with record information, use the method NextRecord to move the record buffer pointer to the next buffer space.

TTableEnhanced WriteBlock Example

```
procedure TMainForm.InsertRecords;
var
  W, RecordCount: Word;
  MoveDone: Boolean;

begin
  RecordCount := 0;
  BlockTotal := 1000;
  BlockSize := 250;
  MoveDone := False;
  { Continue to call the WriteBlock method until it returns True }
  with TableEnhanced1 do
  begin
    repeat
      { Allocate and Initialize the record buffer }
      InitializeBuffer;
      { Fill the record buffer }
      for W := 1 to BlockSize do
      begin
        Inc(RecCount);
        InitIntegerField(FieldByName('IntField').Index, RecCount);
        InitStringField(FieldByName('StrField').Index,
                        Format('Record %d', [RecCount]));
        InitTimeField(FieldByName('TimeField').Index, SysUtils.Time);
        { After each record in the record buffer has been filled, move the
          record buffer ponter to the next record in the record buffer }
        NextRecord;
      end;
      { Place the block of records into the table }
      MoveDone := WriteBlock;
    { Check to see if all records have been placed into the table }
    until MoveDone = True;
  end
end;
```

TTableEnhanced.InsertFast Method

Example

Function: Inserts a record to a table

Description: InsertFast is an alternative to the TTable Insert method. It employes greater exception handling, along with faster insertion capabilities. This method should be used over the AppendFast method when a table is opened on an index. Setting the InsertMode property allows the application to set the record lock type. ReleaseRecordLock method can remove record locks on a table. Set BlockSize to 1 and initialize the record buffer before calling InsertFast.

TTableEnhanced.AppendFast Method

Example

Function: Appends a record to a table

Description: AppendFast is an alternative to the TTable Append method. It employs greater exception handling, along with faster appending capabilities. This method should be used over the InsertFast method when a table has no active index.Set BlockSize to 1 and initialize the record buffer before calling AppendFast.

TTableEnhanced AppendFast Example

```
procedure TMainForm.FastAppendBtnClick(Sender: TObject);
begin
  Inc(RecCount);
  { Set the BlockSize to one so extra record buffers are not allocated.
    NOTE: It would not effect the behavior of AppendFast to have BlockSize
    greater than one;  it would just be a waste of memory and slightly slower. }
  TableEnhanced1.BlockSize := 1;
  try
    with TableEnhanced1 do
    begin
      { Allocate and Initialize the record buffer }
      InitializeBuffer;
      { Fill the record buffer }
      InitIntegerField(FieldByName('IntField').Index, RecCount);
      InitStringField(FieldByName('StrField').Index, Format('Record %d',
                                  [RecCount]));
      InitTimeField(FieldByName('TimeField').Index, SysUtils.Time);
      { Append the record }
      AppendFast;
      { Make sure the Delphi controls are aware of the new record }
      Refresh;
    end;
  except
    on E:EDatabaseError do
    begin
      Dec(RecCount);
      MessageDlg(E.Message, mtError, [mbOk], 0);
    end;
  end;
end;
```

TTableEnhanced Exception Handling

The following exceptions can be handled by an application:

```
EEnhDBError = class(EDatabaseError);
EEnhDBBufferTooBig = class(EEnhDBError);
EEnhDBWrongType = class(EEnhDBError);
EEnhDBIncorrectType = class(EEnhDBError);
EEnhDBTblNotOpen = class(EEnhDBError);
{ dbiPutField }
EEnhDBInvalidHndl = class(EEnhDBError);
EEnhDBOutOfRange = class(EEnhDBError);
EEnhDBInvalidXLation = class(EEnhDBError);
{ dbiInsertRecord }
EEnhDBInvalidParam = class(EEnhDBError);
EEnhDBMinValErr = class(EEnhDBError);
EEnhDBMaxValErr = class(EEnhDBError);
EEnhDBReqdErr = class(EEnhDBError);
EEnhDBLookupTableErr = class(EEnhDBError);
EEnhDBKeyViol = class(EEnhDBError);
EEnhDBFileLocked = class(EEnhDBError);
EEnhDBErrorReadOnly = class(EEnhDBError);
EEnhDBNotSuffTableRights = class(EEnhDBError);
EEnhDBNotSuffSQLRights = class(EEnhDBError);
EEnhDBNoDiskSpace = class(EEnhDBError);
EEnhDBRecLockFailed = class(EEnhDBError);
EEnhDBForiegnKeyErr = class(EEnhDBError);
EEnhDBTableReadOnly = class(EEnhDBError);
{ dbiSaveChanges }
EEnhDBNotSupported = class(EEnhDBError);
{ dbiCopyTable }
EEnhDBInvalidFileName = class(EEnhDBError);
EEnhDBFileExists = class(EEnhDBError);
EEnhDBFamFileInvalid = class(EEnhDBError);
EEnhDBNoSuchTable = class(EEnhDBError);
EEnhDBNotSuffFamilyRights = class(EEnhDBError);
EEnhDBLocked = class(EEnhDBError);
{ dbiIsRecordLocked }
EEnhDBNoCurrRec = class(EEnhDBError);
EEnhDBBOF = class(EEnhDBError);
EEnhDBEOF = class(EEnhDBError);
EEnhDBKeyOrRecDeleted = class(EEnhDBError);
{ dbiTimeEncode }
EEnhDBInvalidTime = class(EEnhDBError);
{ dbiOpenDatabase }
EEnhDBUnknownDB = class(EEnhDBError);
EEnhDBNoConfigFile = class(EEnhDBError);
EEnhDBInvalidDBSpec = class(EEnhDBError);
EEnhDBDBLimit = class(EEnhDBError);
{ dbiRelRecordLock }
EEnhDBNotLocked = class(EEnhDBError);
```

NOTE: Each exception corrilates to the dbi error. i.e. The dbi error: dbiErr_InvalidHndl is translated into the exception EEnhDBInvalidHndl.

TTableEnhanced.InitializeBuffer Method

Example

Function: Initializes the record buffer.

Description: This method is used in conjunction with AppendFast, InsertFast and WriteBock methods. It allocates and clears a memory area to prepare for record tranfers. This method **must** be called before any of the record transfer routines.

TTableEnhanced InsertFast Example

```
procedure TMainForm.FastInsertBtnClick(Sender: TObject);
begin
  Inc(RecCount);
  TableEnhanced1.InsertMode := imNoLock;
  TableEnhanced1.BlockSize := 1;
  try
    with TableEnhanced1 do
    begin
      { Allocate and Initialize the record buffer }
      InitializeBuffer;
      { Fill the record buffer }
      InitIntegerField(FieldByName('IntField').Index, RecCount);
      InitStringField(FieldByName('StrField').Index, Format('Record %d',
                                  [RecCount]));
      InitTimeField(FieldByName('TimeField').Index, SysUtils.Time);
      { Insert the record }
      InsertFast;
      { Make sure the Delphi controls are aware of the new record }
      Refresh;
    end;
  except
    on E:EDatabaseError do
    begin
      Dec(RecCount);
      MessageDlg(E.Message, mtError, [mbOk], 0);
    end;
  end;
end;
```

TTableEnhanced.InsertMode Property

Example

Function: Determines how and if the new record will be locked.

Description: This property can be set to imNoLock: No locking; imReadLock: Read Lock; or imWriteLock: Write Lock. This property only works with the InsertFast method. For more information on record lockeing, refer to the BDE User's Guide.

TTableEnhanced.ReleaseRecordLock Method

Function: Release a Read or Write lock on the table..

Description: If the parameter passed to ReleaseRecordLock is False, only the lock on the current record is released. If the parameter is True, all record locks on the table are released. For more information, refer to DbiRelRecordLock in the BDE User's Guide.

Example:

```
TableEnhanced1.ReleaseRecordLock(True); { Release all record locks }
```

TTableEnhanced.Pack Method

Example

Function: Packs the currently opened table.

Description: Deletes unused records from a Paradox or dBase table. NOTE: This function does not work with SQL databases. For more information, refer to DbiPackTable and DbiDoRestructure in the BDE User's Guide.

TTableEnhanced.CopyTable Method

Example

Function: Copies a table to another table of same type.

Description: Using CopyTable on a local table copies the table, indexes and BLOb files to the name specified. NOTE: The source table must be currently opened to provide the CopyTable method with a valid databse handle. For more information, refer to DbiCopyTable in the BDE User's Guide.

TTableEnhanced.Overwrite Property

Example

Function: Determines if the CopyTable method will automatically overwrite an existing table.

Description: If overwrite is False and the table already exists, the EEnhDBFileExists exception will be raised. If overwrite is True, the table is overwritten without error.

TTableEnhanced.GetSysVersion Method

Example

Function: Retrieves IDAPI system version information.

Description: Please refer to DbiGetSysVersion in the BDE User's Guide for more information.

TTableEnhanced.GetSysInfo Method

Example

Function: Retrieves IDAPI system information.

Description: Please refer to DbiGetSysInfo in the BDE User's Guide for more information.

TTableEnhanced.GetSysConfig Method

Example

Function: Retrieves IDAPI system configuration information.

Description: Please refer to DbiGetSysConfig in the BDE User's Guide for more information.

TTableEnhanced.IsTableShared Method

Example

Function: Determines if the currently opened table is shared.

Description: Please refer to DbiIsTableShared in the BDE User's Guide for more information.

TTableEnhanced.IsTableLocked Method

Example

Function: Determines the number of locks on the currently opened table. This function can determine a read lock or a write lock.

Description: Please refer to DbiIsTableLocked in the BDE User's Guide for more information.

TTableEnhanced.IsRecordLocked Method

Example

Function: Determines the current record is locked

Description: Please refer to DbiIsRecordLocked in the BDE User's Guide for more information.

TTableEnhanced.SaveToDisk Property

Function: Force IDAPI to write each table altering operaton to disk.

Description: For more information, refer to DbiSaveChanges.

Example:

```
TableEnhanced1.SaveToDisk := True;
```

TTableEnhanced.SaveWhenIdle Property

Function: When the application is idle, write the IDAPI buffer to disk.

Description: For more information, refer to DbiUseIdleTime.

Example:

```
TableEnhanced1.SaveWhenIdle := True;
```

TTableEnhanced IDAPI Information Example

Refer to the BDE User's Guide for information on SYSConfig, SYSInfo, and SYSVersion structures.

```
procedure TMainForm.IDAPIInformationBtnClick(Sender: TObject);
var
  Version: SysVersion;
  Info: SysInfo;
  Config: SysConfig;

begin
  Version := TableEnhanced1.GetSysVersion;
  Info := TableEnhanced1.GetSysInfo;
  Config := TableEnhanced1.GetSysConfig;
  { Show appropriate information from the structures}
end;
```

TTableEnhanced CopyTable Example

```
procedure TMainForm.CopyTableBtnClick(Sender: TObject);
begin
  Screen.Cursor := crHourGlass;
  Application.ProcessMessages;
  try
    if TableEnhanced1.Acitve = False then
      TableEnhanced1.Open;
    TableEnhanced1.Overwrite := False;
    { Copy the table }
    TableEnhanced1.CopyTable('NEWTABLE');
    Screen.Cursor := crDefault;
  except
    on EEnhDBFileExists do
    begin
      Screen.Cursor := crDefault;
      MessageDlg('Table already exists.', mtInformation, [mbOk], 0);
    end
    on E:EDatabaseError do
    begin
      Screen.Cursor := crDefault;
      MessageDlg(E.Message, mtError, [mbOk], 0);
    end;
  end;
end;
```

TTableEnhanced Pack Example

```
procedure TMainForm.PackTableBtnClick(Sender: TObject);
begin
  Screen.Cursor := crHourGlass;
  Application.ProcessMessages;
  try
    { Pack table }
    TableEnhanced1.Pack;
    Screen.Cursor := crDefault;
  except
    on E:EEnhDBError do
    begin
      Screen.Cursor := crDefault;
      MessageDlg(E.Message, mtError, [mbOk], 0);
    end;
  end;
end;
```

TTableEnhanced IsRecordLocked, IsTableLocked, IsTableShared Example

```
procedure TMainForm.InformationBtnClick(Sender: TObject);
begin
  { Is the table sharable }
  if TableEnhanced1.IsTableShared = True then
    Label1.Caption := 'Table is shared'
  else
    Label1.Caption := 'Table is NOT shared';

  { Check the amount of Write Locks on the table }
  Label2.Caption := Format('Table has %d table locks',
                      [TableEnhanced1.IsTableLocked(dbiWriteLock)]);

  { Is the current record locked }
  if TableEnhanced1.IsRecordLocked = True then
    Label3.Caption := 'Current record is locked'
  else
    Label3.Caption := 'Current record is NOT locked';
end;
```

TTableEnhanced.NextRecord Method

Example

Function: Moves the record buffer pointer to the next record to be filled.

Description: This method is only used when using WriteBlock. If BlockSize is set to 50, NextRecord will be called 50 times to move the pointer to each new record in the buffer.

TTableEnhanced.InitBooleanFIeld Property

Function: Sets a boolean value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.InsertIt;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitBooleanField(FieldByName('BooleanFld').Index, True);
    InsertFast;
  end;
end;
```

TTableEnhanced.InitCurrencyFIeld Property

Function: Sets a currency value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
var
  Money: Double;

begin
  { Set the Money variable }
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitCurrencyField(FieldByName('CurrencyFld').Index, Money);
    InsertFast;
  end;
end;
```

TTableEnhanced.InitDateFIeld Property

Function: Sets a date value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitDateField(FieldByName('DateFld').Index, SysUtils.Date);
    InsertFast;
  end;
end;
```

TTableEnhanced.InitTimeFIeld Property

Function: Sets a time value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitTimeField(FieldByName('TimeFld').Index, SysUtils.Time);
    InsertFast;
  end;
end;
```

TTableEnhanced.InitTimeStampFIeld Property

Function: Sets a time stamp value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitTimeStampField(FieldByName('TimeStampFld').Index, SysUtils.Now);
    InsertFast;
  end;
end;
```

TTableEnhanced.InitStringFIeld Property

Function: Sets a string value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitStringField(FieldByName('StringFld').Index, 'Hello there');
    InsertFast;
  end;
end;
```

TTableEnhanced.InitFloatFIeld Property

Function: Sets a float value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitFloatField(FieldByName('FloatFId').Index, 283.23);
    InsertFast;
  end;
end;
```

TTableEnhanced.InitIntegerField Property

Function: Sets an integer value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitIntegerField(FieldByName('IntFld').Index, 1923843);
    InsertFast;
  end;
end;
```

TTableEnhanced.InitSmallintFIeld Property

Function: Sets a smallint value to a field in the record buffer

Description: For more information, refer to DbiPutField in the BDE User's Guide.

Example:

```
procedure TMain.SetField;
begin
  with TableEnhanced1 do
  begin
    InitializeBuffer;
    InitSmallintField(FieldByName('SmallFld').Index, 283);
    InsertFast;
  end;
end;
```