Maelstrom TDBOutline v1.5 Help Contents

Click a folder icon to expand or collapse a list of topics for an item.   You can also click the underlined text to see a list of topics.   To expand or collapse all folders, click the icon in the title.

To get Help on Help, press F1

Overview provides information you need to know before using TDBOutline.
Using the DBOutline Component.
About Maelstrom TDBOutline provides information on how to order TDBOutline, future enhancements etc.

Overview

The following topics provide basic information you should know before you start using TDBOutline.
What is TDBOutline?
What's new in version 1.5?
What can TDBOutline be used for?
Installation Instructions.

What can TDBOutline be used for?

TDBOutline can be used to display, manipulate and save the hierarchic structure of recursively related (self-referencing) data.   Recursively related tables are used for many purposes, including the following:

- Corporate personnel hierarchy
- Genealogy (Family trees)
- Notes/subnotes (document management)
- GL account structures
- Business channel/segment structures
- Project tasks/subtasks
- Decision trees
- Process flows
- Universal address master files
- PIMs

TDBOutline contains an algorithm that can load hierarchic data into an outline, thus saving the programmer many hours of coding.

Events
In addition to the events inherited from TOutline, TDBOutline includes the following:

Key events

OnAutoDragDrop OnLoadFromDataSet

What is TDBOutline?

Welcome to release 1.5 of the Maelstrom TDBOutline component.

TDBOutline is a native VCL Delphi component designed to allow the loading, navigation, and manipulation of hierarchically organized data contained in a database.

TDBOutline is a descendant of the TOutline class, inheriting all TOutline's properties, methods and events.

TDBOutline has additional properties, methods and events that provide the following:

- Automatic loading of recursively related data into the outline, preserving the hierarchic structure. One call to method LoadFromDataSet is all it takes to populate the outline with all the table data.

- Automatic drag and drop facilities that move a node and all its children to be children of the dropped-on node. Dragging a node off the outline control will cause the outline to scroll, enabling drag and drop to a node not visible on the same page as the from-node. A default Move confirmation message can be enabled, disabled or overridden with your own custom message.

- Automatic record-pointer synchronization -- as you navigate through the outline, TDBOutline will move the record pointer in the table to the associated record. It will even search for the appropriate index to enable FindKey synchronization.

- Automatic outline synchronization -- as you navigate through the dataset, you can cause the selected outline to synchronize with the selected dataset record by calling a single method.

- Automatic update of recursive field upon drag and drop -- when a node is dropped on another node, the dropped-on node's key field will be written to the dropped node's recursive field.

- Mass update or mass cancel of drag-drop changes with a single method call.

Methods

In addition to the methods inherited from TOutline, TDBOutline includes the following:

🔑 Key methods

🔑 AutoDrop	🔑 LoadFromDataSet
🔑 AddDBRecord	🔑 SynchOutline
🔑 ChangeDBRecord	🔑 UpdateDraggedNodes

ChangeDBRecord Method

Applies to
TDBOutline

Declaration
`procedure` ChangeDBRecord(FromDataField: string; ToDataField: string; ToDataFieldDisplay: string);

Description

The ChangeDBRecord method is designed to change the contents of a node and its data to reflect a change in the underlying dataset record.   This method takes care of the pointers required to maintain DBOutline <-> dataset synchronization through DataAutoSynch and SynchOutline.

The FromDataField parameter corresponds to the DataField field of the record prior to the change. ToDataField corresponds to the DataField field of the record after the change. ToDataFieldDisplay corresponds to the DataFieldDisplay field of the record after the change. DataFieldRecursive is not updated via this method.   Use AutoDrop to modify DataFieldRecursive.

Note: Changing the DataField value of a node by any other method will disable synchronization for the changed node.   Modifying the attached data of an outline node via any means will also disable synchronization for the modified node, and may prevent synchronization for all nodes.

DataAutoSynch Property

Applies to
TDBOutline

Declaration
```
property DataAutoSynch: Boolean;
```

Description
The DataAutoSynch property determines whether the record pointer of the dataset loaded into the DBOutline control will move to the record corresponding to the selected outline node whenever the selected outline node changes. Also, DataAutoSynch must be True to allow DataAutoUpdate to be True, and to allow use of method UpdateDraggedNodes. These are the possible values:

Value	Meaning
True	If True is selected, and a new outline node is selected, the record pointer of the outline's underlying dataset will be moved to the record represented by the outline node.
False	If False is selected the record pointer of the outline's underlying dataset will not be synchronized with the selected node.

DataAutoDrag Property

Applies to
TDBOutline

Declaration
```
property DataAutoDrag: Boolean;
```

Description
The DataAutoDrag property determines whether the DBOutline control, when dragging, will automatically scroll when a node is dragged outside the control's boundaries. These are the possible values:

Value	Meaning
True	If True is selected, and drag mode is entered, then when a node is dragged outside the boundaries of the control, the control will scroll in the appropriate direction.
False	If False is selected and drag mode is entered, then scrolling will not be automatic.

About Maelstrom TDBOutline

How to Order TDBOutline
How to get Support
What to look forward to

Maelstrom TDBOutline Help Contents

Click a folder icon to expand or collapse a list of topics for an item.   You can also click the underlined text to see a list of topics.   To expand or collapse all folders, click the icon in the title.

To get Help on Help, press F1

Maelstrom TDBOutline Help Contents

Click a folder icon to expand or collapse a list of topics for an item.   You can also click the underlined text to see a list of topics.   To expand or collapse all folders, click the icon in the title.

To get Help on Help, press F1

Maelstrom TDBOutline Help Contents

Click a folder icon to expand or collapse a list of topics for an item.   You can also click the underlined text to see a list of topics.   To expand or collapse all folders, click the icon in the title.

To get Help on Help, press F1

Maelstrom TDBOutline Help Contents

Click a folder icon to expand or collapse a list of topics for an item.   You can also click the underlined text to see a list of topics.   To expand or collapse all folders, click the icon in the title.

To get Help on Help, press F1

Overview provides information you need to know before using TDBOutline.
Using the DBOutline Component.

About Maelstrom TDBOutline provides information on how to order TDBOutline, future enhancements etc.

- How to Order TDBOutline
- How to get Support
- What to look forward to

How to Order TDBOutline

Maelstrom TDBOutline v1.5 Order Form

Maelstrom TDBOutline v1.5 may be ordered through the Shareware Registration System on CompuServe.

CompuServe: SWREG   ID: 8635

Please note that Maelstrom Software does not accept Credit Card Orders. To order direct, please mail a copy of this order form, along with your check or money order to:

Maelstrom Software
85 Fernhill Blvd
Oshawa, Ontario
Canada L1J 5J1

Please type or print clearly:
Name: ____

Company (if applicable): ____

Mailing Address: ____

City: ________________________   State: ____________   Postal Code: __________

Country: ________________________

A copy of TDBOutline must be ordered for each machine on which the program will be used.   Your check or money order should be made payable to "Maelstrom Software."

# of Copies:	_________	x USD $39.00 =	$__________
TOTAL:			$__________

If you would like to receive notice of future upgrades and new Maelstrom product releases via e-mail, please include your Internet or CompuServe e-mail address: ________________________

How did you hear about Maelstrom TDBOutline? ______________________________

You will receive a 3.5" disk through the Mail upon receipt of order.

Please write or send Internet e-mail to 71431.62@compuserve.com to check status or ask questions. Thank you for ordering the Maelstrom TDBOutline component for Delphi.

If you have a Compuserve account and would prefer to receive TDBOutline through e-mail, Go SWREG # 8635

How to get Support

Support is available to registered users through e-mail:

Maelstrom Software
85 Fernhill Blvd.
Oshawa, Ontario
Canada L1J 5J1

CompuServe: 71431,62
Internet: 71431.62@compuserve.com

Registered users will be notified of bug fixes, enhancements and new products via e-mail or postal mail--depending on the method by which the user registered.

Bug reports and enhancement requests are always welcome.

What to look forward to

Upcoming Enhancements

The following enhancements are in the planning/development stages:

- Method to load data from one-many-many... relationships.   Property editor to allow the user to define these relationships.
- Glyph & Font definition at outline levels and by dataset status code.
- Semi-virtualization -- allow loading of much larger datasets by loading only the top two levels in LoadFromDataSet, and loading child nodes as nodes are expanded.

Please let us know of any other improvements you would like to see.

A recursive relation, also referred to as a *self-referencing* relation, is one in which a record in a table refers to another record in the same table, as follows:

Supervisor refers to EmpNo.

In this example, EmpNo is said to be the *Key Field*, and Supervisor the *Recursive* or *Self-Referencing* field.
Within the context of TDBOutline, EmpNo would be the DataField, and Supervisor the DataFieldRecursive.

Adding new records to a DBOutline

Example

The AddDBRecord method should be used to add new nodes to the DBOutline. AddDBRecord ensures that the proper information is stored in the node's attached data. If a node is added using the inherited Add or AddChild methods, DataAutoSynch and SynchOutline will not function for that node.

DataField Property

Applies to
TDBOutline

Declaration
```
property DataField: String;
```

Description

The DataField property identifies the field from which TDBOutline controls the loading and display of its hierarchical data.
DataField represents the key field of the recursive relation defined for the dataset.
During drag-and-drop operations, the DataField of the dropped-on node will be used to update the field of the dragged node's record identified by the DataFieldRecursive property.
The dataset the field is located in is specified by DataSource.

DataFieldDisplay Property

Applies to
TDBOutline

Declaration
```
property DataFieldDisplay: String;
```

Description

The DataFieldDisplay property identifies the field which TDBOutline will display after loading its hierarchical data.
DataFieldDisplay can be a field in TDBOutline's underlying dataset, or a calculated field defined in the TFields editor.
After TDBOutline's LoadFromDataSet method is called, DataFieldDisplay will be displayed as the text value of each outline node.
The dataset the field is located in is specified by DataSource.

LoadFromDataSet Method

Applies to
TDBOutline

Declaration
`procedure` LoadFromDataSet;

Description

The LoadFromDataSet method executes an algorithm that loads recursively-related data from a dataset into the DBOutline control.
The dataset that the data is loaded from is specified by the DataSource property.
The hierarchy is defined to LoadFromDataSet via the DataField and DataFieldRecursive properties of TDBOutline.   The data that will be displayed is specified by DataFieldDisplay.   LoadFromDataSet will first clear the DBOutline of all nodes, then add the dataset records.

DataFieldRecursive Property

Applies to
TDBOutline

Declaration
`property DataFieldRecursive: String;`

Description

The DataFieldRecursive property identifies the field which defines the recursive relation from which TDBOutline controls the loading and display of its hierarchical data.
DataFieldRecursive represents the referencing field of the recursive relation defined for the dataset.
DataFieldRecursive references the field specified in the DataField property of TDBOutine.
During drag-and-drop operations, the field in the from-record specified by DataFieldRecursive will be updated with the value of the field in the to-record specified by the DataField property.
The dataset the field is located in is specified by DataSource.

DataSource Property

Applies to
TDBOutline

Declaration
`property` DataSource: TDataSource;

Description

The DataSource property determines where the component obtains the data to display. Specify the data source component that identifies the dataset the data is found in.

MasterParent Property

Applies to
TDBOutline

Declaration
```
property MasterParent: Boolean;
```

Description
The MasterParent property determines whether method LoadFromDataSet will load top-level records as children of a master node, or as top-level nodes. These are the possible values:

Value	Meaning
True	If True is selected, LoadFromDataSet will load all top-level parent records (i.e., the field represented by DataFieldRecursive is blank) as children of a master parent node. The text of the master node is specified by property MasterParentText.
False	If False is selected, LoadFromDataSet will load all top-level parent records as top-level outline nodes.

MasterParentText Property

Applies to
TDBOutline

Declaration
```
property MasterParentText: String;
```

Description
The MasterParentText property specifies the text of the master parent node that will be used if the MasterParent property is set to True.

AutoDrop Method

Applies to
TDBOutline

Declaration
`procedure` AutoDrop(X, Y: integer);

Description

The AutoDrop method moves the selected DBOutline node to be a child of the node at screen pixel coordinates X and Y.   All children of the selected node will be moved with it.
If the DataAutoSynch and DataAutoUpdate properties are set to True, AutoDrop will also update the field specified by DataFieldRecursive of the record represented by the selected node, with the content of the field specified by DataField of the record represented by the node at X,Y.   If DataAutoSynch is True and DataAutoUpdate is False, the record will not be updated automatically, but may be updated by calling method UpdateDraggedNodes.
Typically, AutoDrop should be called in the OnDragDrop event of the TDBOutline control to enable drag-and-drop manipulation of the hierarchy of the dataset that was loaded into TDBOutline using the LoadFromDataSet method.
See Manipulating the hierarchy via drag-and-drop for details on the use of AutoDrop.

OnLoadFromDataSet Event

Applies to
TDBOutline

Declaration
`property` OnLoadFromDataSet: TLoadFromDataSetEvent;

Description

The OnLoadFromDataSet event occurs whenever the LoadFromDataSet method evaluates a new record for load to TDBOutline. Use the OnLoadFromDataSet event handler to evaluate whether a record should be loaded, and set the value of Accept appropriately. If Accept is set to false, LoadFromDataSet will skip the record. If true, it will add the record as a node.
The Sender parameter of the OnLoadFromDataSet event is the object calling the event handler.
The Accept parameter is a boolean value indicating whether the record should be loaded.
The ToNode parameter is the DataField value of the record being evaluated.

Installation Instructions

To install the TDBOutline component, copy all distributed files into a directory of your choice.
You may wish to install the following files to a component library (example: \delphi\lib).

DBOUTLN.DCU
DBOUTLN.RES

From Delphi's menu, invoke OPTIONS | INSTALL COMPONENTS and install component DBOUTLN.DCU.
The DBOutline component will appear on the Data Controls pallet.

To run the demonstration program, load and compile DEMO.DPR in Delphi.

To integrate TDBOutline's help file with Delphi, install the KWF file:

1. Make a backup copy if Delphi's master index file, \delphi\bin\delphi.hdx.
2. Copy file TDBOUTLI.HLP to \delphi\bin.
3. Copy file TDBOUTLI.KWF to \delphi\help
4. Run the program HELPINST (installed with Delphi).
5. Open keyword index file \delphi\bin\delphi.hdx.
6. Add keyword file \delphi\help\tdboutli.kwf.
7. Save.

OnAutoDragDrop Event

Applies to
TDBOutline

Declaration
`property` OnAutoDragDrop: TAutoDragDropEvent;

Description

The OnAutoDragDrop event occurs whenever the AutoDrop method is called, prior to the execution of the drop. Use the OnAutoDragDrop event handler to specify how TDBOutline should handle the drop of one node onto another.
If no event handler exists, TDBOutline will send a default message asking the user if the drop should be performed.
If an event handler exists, TDBOutline will only proceed with the drop if the value of Accept is true.
Hence, simply setting Accept to true will override the default message, and perform the drop without confirmation from the user.

Tip: replace the default confirmation message with one of your own in the OnAutoDragDrop event handler.   If the user confirms, set Accept to true; otherwise set Accept to false.

The Sender parameter of the OnAutoDragDrop event is the object calling the event handler.
The Accept parameter is a boolean value indicating whether the record should be dropped.
The ToNode parameter is the text value of the node being dropped on.
The FromNode parameter is the text value of the node being dropped.

TDBOutline Component

Properties Methods Events Tasks

Unit
DBOutln

Description

The TDBOutline component inherits all properties, methods and events of TOutline.   Perform a search on TOutline for more information.

The TDBOutline component is used to display and manipulate multilevel outlines of dataset data. Use a DBOutline to visually organize information in a hierarchical tree. Each item in a DBOutline is contained in a TOutlineNode object.

An item in a DBOutline can be accessed by the Items property. The items are indexed from 1 to the number of items. For example, Items[1] refers to the first (topmost) item. Since Items is the default array property of TDBOutline, an item can also be accessed immediately following the outline name. For example, DBOutline1.Items[1] and DBOutline1[1] refer to the same outline item.

Use the LoadFromDataSet method to load dataset records into the DBOutline.

Use the AddDBRecord method to add a new dataset record to a DBOutline. Use ChangeDBRecord to change the dataset information contained in a DBOutline node.   Use Delete to remove items.

The currently selected item is specified by the SelectedItem property. When the user selects a new item of the outline (by clicking with the mouse or pressing an Arrow key), the newly selected item is specified by SelectedItem.

Properties

In addition to the properties inherited from TOutline, TDBOutline includes the following:

▶ Run-time only	🔑 Key properties

🔑 DataAutoDrag	IgnoreCyclicalDrops
🔑 DataAutoSynch	MasterParent
🔑 DataAutoUpdate	MasterParentText
🔑 DataField	▶ 🔑 SynchSuccess
🔑 DataFieldDisplay	
🔑 DataFieldRecursive	
🔑 DataSource	

DataAutoUpdate

Applies to
TDBOutline

Declaration
```
property DataAutoUpdate: Boolean;
```

Description
The DataAutoUpdate property may only be True if the DataAutoSynch property is also True. DataAutoUpdate determines whether the underlying dataset will be automatically updated when method AutoDrop is called. AutoDrop is typically called after a drag-drop move of a node to another node. Hence DataAutoUpdate determines whether the dragged node's underlying dataset record will have its DataFieldRecursive field automatically updated to contain the value of the dropped-on node's underlying dataset record's DataField field.

If DataAutoUpdate is false, method UpdateDraggedNodes can be called to perform the update of all dragged nodes that have not yet been updated (provided DataAutoSynch is True).

Value	Meaning
True	If True is selected, and method AutoDrop is called in the OnDrop event of the DBOutline, the dragged node's underlying dataset record's DataFieldRecursive field will be updated to contain the value of the dropped-on node's underlying dataset record's DataField field.
False	If False is selected, the above update will not automatically occur when AutoDrop is called. Instead, method UpdateDraggedNodes will have to be called to perform the update.

IgnoreCyclicalDrops Property

Applies to
TDBOutline

Declaration
```
property IgnoreCyclicalDrops: Boolean;
```

Description
The IgnoreCyclicalDrops property indicates whether a default error message will be displayed if a cyclical drag-drop operation is attempted.   A cyclical drag-drop is one in which a node is dropped onto itself, or one of its child nodes.   A cyclical drop will not be performed regardless of the value of IgnoreCyclicalDrops -- the property only controls whether or not the error message is displayed.

Value	Meaning
True	If True is selected, and a cyclical drag-drop occurs, no error message will be displayed.
False	If False is selected, and a cyclical drag-drop occurs, a default error message will display.

SynchSuccess

Applies to
TDBOutline

Declaration
```
property SynchSuccess: Boolean;
```

Description
The SynchSuccess property is a Read-only property that indicates if the last attempt by TDBOutline to synchronize the dataset pointer to the selected DBOutline node was successful.   If SynchSuccess is True, the DBOutline component was able to synchronize the dataset, otherwise it was unable to synchronize.

SynchSuccess can be evaluated at runtime to determine if the record corresponding to the selected outline was found.   SynchSuccess can only be True if DataAutoSynch is True.   SynchSuccess will always be False when a MasterParent node is selected, because there is no corresponding dataset record.

Using the DBOutline Component

TDBOutline Reference

Purpose

Use the DBOutline component when you want to load dataset data into an outline to display and manipulate the hierarchical relationship between the data records. Each record in the dataset is represented by a node in the outline.

The DataSource property defines which table or query will be loaded into the outline. The DataFieldDisplay property specifies the dataset field that will be displayed as the text of the outline node. DataFieldDisplay can be a calculated field defined with the TField editor.

The DataField and DataFieldRecursive properies define the recursive relation that will be loaded into the outline--the LoadFromDataSet method uses these properies to drive the algorithm that iterates through the dataset, loading each record into the outline. DataField and DataFieldDisplay must be of the same type, and DataField must be a unique key of the dataset. In addition, if the datasource's dataset is a TTable, there must be an index, primary or secondary, that has DataField as the first key in order to take advantage of DataAutoSynch. If the datasource's dataset is a TQuery, TDBOutline will move through the dataset to synchronize without an index.

Tasks

Loading a TDBOutline component

Manipulating the hierarchy via drag-and-drop

Synchronizing the DBOutline to reflect changes to the underlying dataset

Adding new records to a DBOutline

Editing existing records in a DBOutline

Deleting records from a DBOutline

Notes

- If DataAutoSynch is True and the dataset is indexed on a primary key that is not DataField, TDBOutline will change to an index based on DataField to perform dataset synchronization, then switch back to the original index. If the dataset originally had a Range set, the range will be lost after dataset synchronization. Thus, if a filtered dataset is required a TQuery must be used.
- If DataSource is pointing to a TQuery, the TQuery's RequestLive property must be true and the SQL statement must meet Delphi's syntax requirements for a live result set in order to take advantage of methods that update the dataset.
- If DataSource is pointing to a TTable, the TTable's ReadOnly property must be false to take advantage of methods that update the dataset.

Manipulating the hierarchy via drag-and-drop

Example

Changes can be made to the DBOutline hierarchy via drag-and-drop.
If the DataAutoDrag property is set to True, TDBOutline will provide automatic drag-and-drop scrolling of the outline. Calling the AutoDrop method in the OnDragDrop event of the DBOutline will automatically move the dragged node and make it a child of the dropped-on node.

Hint: if your outline contains more than one root (top-level) node, set property MasterParent to True and assign a string to MasterParentText. This will create a single root with text MasterParentText and make your top-level nodes children of it. You will then be able to drop a node on the MasterParent node to make it top-level. If a node is dropped on the MasterParent node, the DataFieldRecursive value will be replaced with a null value. This indicates that the node has become a top-level parent.

If the DataAutoSynch and DataAutoUpdate properties are set to True, a drag-drop operation will automatically update the underlying dataset with the change (i.e., post the DataField value of the dropped-on node to the DataFieldRecursive field of the dragged node).

If the DataAutoSynch property is set to True, and the DataAutoUpdate property is set to **False**, then drag-drop changes are **not** automatically posted to the underlying dataset. In this case the method UpdateDraggedNodes can be called to post all drag-drop changes to the dataset.

If DataAutoSynch is false, drag-drop changes cannot be posted to the dataset.

If the MasterParent property is set to True, and a node is dropped on the MasterParent node, the DataFieldRecursive value will be replaced with a null value. This indicates that the node has become a top-level parent.

Example 1
The following code illustrates the implementation of drag-and-drop manipulation of TDBOutline nodes.

In the first example, DataAutoDrag = True, DataAutoSynch = True, DataAutoUpdate = True

```
  procedure TMainForm.DBOutline1DragOver(Sender, Source: TObject;
    X, Y: Integer; State: TDragState; var Accept: Boolean);
  begin
    if Source = DBOutline1 then Accept := True;
  end;

  procedure TMainForm.DBOutline1DragDrop(Sender, Source: TObject; X,
    Y: Integer);
  begin
    if Source = DBOutline1 then
      DBOutline1.AutoDrop(x, y);
  end;
```

To override TDBOutline's default confirmation message (presented during AutoDrop processing), add code similar to the following in the OnAutoDrop event handler:

```
  procedure TMainForm.DBOutline1AutoDragDrop(Sender: TObject;
    var Accept: Boolean; var FromNode, ToNode: OpenString);
  begin
    if MessageDlg(' Have employee ['+ FromNode +'] report to ['+ ToNode +
']?', mtConfirmation, mbOkCancel,0) = mrOk
      then Accept := True
    else Accept := False
  end;
```

To prevent any confirmation message, simply set Accept := True in the OnAutoDragDrop event hander.

Because DataAutoSynch = True and DataAutoUpdate = True, method AutoDrop will immediately post the change to the dataset record.

Example 2
In the second example, DataAutoDrag := True, DataAutoSynch := True, DataAutoUpdate = False

Given the same code as above, because DataAutoSynch = True and DataAutoUpdate = False, method AutoDrop will not post the change to the dataset record.
To post drag-drop changes given DataAutoUpdate = False, method UpdateDraggedNodes must be called. UpdateDraggedNodes is used to post all drag-drop changes made to the DBOutline that are outstanding (i.e., not posted to the dataset). The method can be used in conjunction with property DataAutoUpdate = False to allow the user to manipulate the hierarchy without saving, then save or abandon.

```
  procedure TMainForm.btnSaveDragDropChangesClick(Sender: TObject);
  begin
    DBOutline1.UpdateDraggedNodes(Self);
  end;
```

Example 3
In the third example, DataAutoDrag := True, DataAutoSynch := False, DataAutoUpdate = False

In this example, the code for example 1 applies.   Because DataAutoSynch =False, however, neither AutoDrop nor UpdateDraggedNodes will be able to update the dataset to reflect drag-drop changes to the nodes.

UpdateDraggedNodes Method

Applies to
TDBOutline

Declaration
`procedure` UpdateDraggedNodes(Sender: TObject);

Description

The UpdateDraggedNodes method is used to post all outstanding drag-drop changes to the dataset. Drag-drop changes are outstanding if they have not been saved to the dataset.   UpdateDraggedNodes is only functional if DataAutoSynch is True.   If DataAutoUpdate is also True, drag-drop changes will always be posted to the dataset immediately, and UpdateDraggedNodes will have no effect (as there will be no outstanding changes to update).   See Manipulating the hierarchy via drag-and-drop for more details.

Loading a TDBOutline component

Example

To use a TDBOutline component,

1. Place the TDBOutline component on the form.
2. Set the DataSource property to the name of a TTable or TQuery component already on the form.
3. Set the DataField property to the key field of the DataSource.
4. Set the DataFieldRecursive property to the field that references DataField for the recursive relation.
5. Set the DataFieldDisplay property to the field you wish displayed in the outline node. DataFieldDisplay can be a calculated field.
6. In the FormCreate event handler of the form, call the LoadFromDataSet method.

TDBOutline considers a node to be a top-level parent, or root, if its DataFieldRecursive field is empty, zero, or blank.

Example
The following code illustrates the use of method SynchOutline to synchronize the selected DBOutine node to the selected dataset record.

In this example the OnDataChange event of the DataSource is used. This ensures the SynchOutline will be called whenever the record pointer moves.

```
procedure TMainForm.DataSource1(Sender: TObject; Field: TField);
begin
  if StartSynching then
    DBOutline1.SynchOutline(Self);
end;
```

StartSynching is a boolean variable used to control whether SynchOutline is triggered. In some instances (for example, appending a new record), you would not want SynchOutline to execute, because the DBOutline node corresponding to the record being appended does not yet exist. See Adding new records to a DBOutline for more information on the use of a boolean variable to enable/disable SynchOutline.

Example 1
The following code illustrates the loading of a TDBOutline from a dataset

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  Table1.Open;
  DBOutline1.LoadFromDataSet;

  {Expand the MasterParent node, and select the first top-level node }
  if DBOutline1.ItemCount > 1 then
  begin
    DBOutline1.Items[1].Expand;
    DBOutline1.SelectedItem := 2;
  end;
end;
```

Synchronizing the DBOutline to reflect changes to the underlying dataset

Example

If property DataAutoSynch is set to True and its conditions are met, the record pointer of the dataset will move to the record that corresponds to the currently selected DBOutline node.   Each time a new DBOutline node is selected, the record pointer will automatically move.

It remains for the programmer to call method SynchOutline at appropriate points in order to synchronize the currently selected DBOutline node to the currently selected dataset record.

SynchOutline Method

Applies to
TDBOutline

Declaration
`procedure` SynchOutline(Sender: TObject);

Description

The SynchOutline method changes the selected DBOutline node to match the currently selected dataset record.
See Synchronizing the DBOutline for details on usage.

AddDBRecord

Applies to
TDBOutline

Declaration
procedure AddDBRecord(NewDataField: string; NewDataFieldDisplay: string; NewDataFieldRecursive: string);

Description

The AddDBRecord method is designed to add a new dataset record to the DBOutline. This method takes care of the pointers required to maintain DBOutline <-> dataset synchronization through DataAutoSynch and SynchOutline. The new node is added as a child node of the currently selected DBOutline node.

The NewDataField parameter corresponds to the DataField field of the record to be added. NewDataFieldDisplay corresponds to the DataFieldDisplay field of the record to be added. NewDataFieldRecursive corresponds to the DataFieldRecursive field of the record to be added.

Note: Adding nodes to a DBOutline by any other method (i.e. by the inherited Add or AddChild methods) will disable synchronization for the added node. Modifying the attached data of an outline node via any means will also disable synchronization for the modified node, and may prevent synchronization for all nodes.

Example
The following code illustrates how to add a new dataset record to TDBOutline.

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  StartSynching := False;
  Table1.Append;
  Table1.FieldByName('EmpNo').AsString := Edit1.Text;
  Table1.FieldByName('LastName').AsString := Edit2.Text;
  Table1.FieldByName('Supervisor').AsString := Edit3.Text;
  Table1.Post;
  DBOutline1.AddDBRecord(Edit1.Text, Edit2.Text, Edit3.Text);
  StartSynching := True;
end;
```

The boolean variable StartSynching is used as per Synchronizing the DBOutline to reflect changes to the underlying dataset.

Editing existing records in a DBOutline

Example

The ChangeDBRecord method should be used to modify existing nodes in a DBOutline. ChangeDBRecord ensures that the proper information is modified in the node's attached data. If a node's DataField value is modified by any other method, DataAutoSynch and SynchOutline will not function for that node.

Deleting records from a DBOutline

Example
To delete a node from a DBOutline control, use the Delete method inherited from TOutline. Particular attention must be paid to the deletion of records / nodes, as DataAutoSynch and SynchOutline can move the record pointer / selected node during the operation. See the example for details.

Example

The following code illustrates how to change a dataset record in a DBOutline control.

```
  procedure TForm1.SpeedButton1Click(Sender: TObject);
  begin
    StartSynching := False;
    Table1.Edit;
    Table1.FieldByName('EmpNo').AsString := Edit1.Text;
    Table1.FieldByName('LastName').AsString := Edit2.Text;
    Table1.Post;

DBOutline1.ChangeDBRecord(IntToStr(DBOutline1.Items[DBOutline1.SelectedItem].I
ndex)
     , Edit1.Text, Edit2.Text);
    StartSynching := True;
  end;
```

For an explanation of the use of boolean variable StartSynching, see Synchronizing the DBOutline to reflect changes to the underlying dataset.

Example
The following code illustrates how to delete a dataset record from TDBOutline.

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
  StartSynching := False;
  Table1.Delete;
  DBOutline1.Delete(DBOutline1.SelectedItem);
  StartSynching := True;
end;
```

Note that the record in the table was deleted first. If DataAutoSynch is set to True, and the DBOutline node is deleted first, the new selected node will cause the dataset to resynchronize, and the incorrect record would be deleted.

Also, note the use of a boolean variable, StartSynching. In this example, it is assumed that the OnDataChange event of the DataSource calls method SynchOutline if StartSynching is True. This gives the programmer the ability to prevent the table from resynchronizing the DBOutline when the dataset record is deleted, thus preventing the incorrect node from being deleted. For more information on the use of boolean variable StartSynching, see Synchronizing the DBOutline to reflect changes to the underlying dataset.

What's new in version 1.5?

• Version 1.5 has been modified to prevent the display of DataField within the node text. Resulting from this modification, the Data element of each node is now occupied with information TDBOutline needs to navigate and synchronize the DBOutline.
IMPORTANT: the attached data (pointer) of each DBOutline node is now used by TDBOutline, and should not be modified by the programmer. If modification of the attached data is required, version 1.0 should be used. Version 2.0 is planned to include a new data element that the programmer can use.

• Several new methods have been added to facilitate the addition/editing of outline nodes after the initial call to LoadFromDataSet. These methods are AddDBRecord and ChangeDBRecord.

• A new method has been added to allow the developer to synchronize the DBOutline's selected node to changes in the underlying dataset's selected record: SynchOutline.

• A new property has been added, DataAutoUpdate, that specifies if drag-drop changes should be immediately posted to the dataset or not. In conjunction with this new property, a method has been added, UpdateDraggedNodes, that will post all un-posted drag-drop changes to the dataset. This allows the manipulation of the DBOutline's structure without changing the dataset. The modifications to the hierarchy can be saved with a call to UpdateDraggedNodes, or abandoned with a call to LoadFromDataSet.

• In keeping with the above change, AutoDrop has been modified to post drag-drop changes to the dataset only if DataAutoUpdate is True.

• Improved support for TQueries has been added. DataAutoSynch will now work with a TQuery. All other functionality also works with TQuery, provided the TQuery is capable of being updated.

• A new property has been added to prevent the display of the default error message if a cyclical drag-drop is attempted: IgnoreCyclicalDrops.

• Improved index support has been provided. The DataAutoSynch mechanism will now maintain the initial index of the dataset, switching to an index with primary key DataField as needed, then switching back.