

## Px7Table unit

This unit contains a descendant of TTable used with a number of routines in [BDEDoRxS](#). For every application that uses BDEDoRxS it is mandatory that the unit Px7Table is in the search path. For every application using BDEDoRxS routines together with descending indices it is also mandatory that you use Px7Table for all restructuring / index creation routines.

Of course this component makes sense even without BDEDoRxS... probably the greatest advantage compared to standard TTable is it's ability to [AutoSaveChanges](#) with local tables.

### Components

[TPx7Table](#)

### Types

[TPDXOpenFailType](#)

[TPDXOpenFailure](#)



## TPx7Table Component

[Properties](#)

[Methods](#)

[Events](#)

### Unit


Px7Table

### Description

Px7Table is a TTable descendant with an added method to make use of the new Paradox level 7 descending indices. With this type of index, you can determine the sortorder for every single field in the index.

It also adds a new event OnOpenFailure which will be triggered when opening the table fails for one of these reasons: index out of date, blob file, lookup table, master table or detail table missing.


## Methods


 Key methods

 AddPx7Index    GetLevel

 GetDescFields    SetLevel

## Events

 Key events  
OnOpenFailure

 Key Property  
AutoChangeLevel

AutoSaveChanges

## AddPx7Index method

### Applies To

TPx7Table

### Declaration

```
procedure AddPx7Index(const Name, Fields, DescFields: string;  
                     Options: TIndexOptions);
```

### Description

AddPx7Index works just like TTable.Addindex except for an additional string parameter named DescFields:

```
procedure AddPx7Index(const Name,  
                     Fields,  
                     DescFields: string;  
                     Options: TIndexOptions);
```

With ixDescending in Options DescFields holds the list of fields that will actually be indexed descending, i.e. for a full descending index it equals the Fields parameter.

The following example creates an index that is descending on the 1st and 3rd and ascending on the 2nd field:

```
Px7Table1.AddPx7Index('AscDescMix',  
                     'Field1;Field2;Field3',  
                     'Field1;Field3',  
                     [ixDescending]);
```

---

Note on the internals of setting desc indices...:

The sortorder of a field in the index is determined by setting the correspondent entry in IDXDesc.iUnUsed to 1 for descending or to 0 for ascending. There is one entry in iUnUsed for every field in IDXDesc.aiKeyFld (i.e. for every field in the key...). The arrays for the above example would look like this:

```
aiKeyFld = (1,2,3);  
iUnUsed  = (1,0,1);
```

NOTE WIN32: abDescending is used instead of iUnUsed!

See the code for EncodePx7IndexDesc for more details.7bdf7bdf7bdf7bdf7bd

## GetDescFields method

### Applies To

TPx7Table

### Declaration

```
function GetDescFields(const IxName: string): string;
```

### Description

This one can be used to retrieve a list of descending fields in in index (useful for recreation of indices in code). It returns a string holding the fieldnames seperated by ';', just like the string used for describing index fields with TTable.AddIndex.

## GetLevel method

### Applies To

TPx7Table

### Declaration

```
function GetLevel: string;
```

### Description

A GetLevel method was added for completeness. Both use strings for input/output. See [SetLevel](#) as well.

## SetLevel method

### Applies To

TPx7Table

### Declaration

```
procedure SetLevel(ALevel: string);
```

### Description

As it turned out that TUTILITY.DLL v2.52 (the one that ships with Pfw7 for Win3.1x) not only drops indices when rebuilding a table but sets the table level back to 5, I added a method to set the table level. This call makes your Paradox table a level 7 table:

```
Px7Table1.SetLevel('7');
```

NOTE the table has to be opened exclusive for this call.



## OnOpenFailure event

### Applies To

TPx7Table

### Declaration

property OnOpenFailure: TPDXOpenFailure;

### Description

This event will be triggered when opening the table fails for one of these reasons: index out of date, blob file, lookup table, master table or detail table missing. You will be informed of the failure cause through type TPDXOpenFailType.

## TPDXOpenFailType type

### Unit

Px7Table

### Declaration

```
TPDXOpenFailType = (ofNone, ofIndexOutOfDate,  
                    ofMBMissing, ofLookUpMissing,  
                    ofDetailMissing, ofMasterMissing);
```

### Description

This type is used with event OnOpenFailure to determine the cause why the opening failed.

## TPDXOpenFailure type

### Unit

Px7Table

### Declaration

```
TPDXOpenFailure = procedure(Sender: TObject;  
                             FailType: TPDXOpenFailType) of object;
```

### Description

This type is used as the prototype of the event OnOpenFailure of Px7Table.

## AutoSaveChanges Property

Unit

[Px7Table](#)

### Declaration

```
property AutoSaveChanges: boolean;
```

### Description

Determines whether DBISaveChanges is called after any post or delete actions. DBISaveChanges flushes the buffer to the next instance, i.e. either disk or cache. BTW with local databases switching off the write behind cache is recommended as well as having this property true.

## AutoChangeLevel Property

### Unit

[Px7Table](#)

### Declaration

```
property AutoChangeLevel: boolean;
```

### Description

Determines whether Px7Table automatically changes the table level to level 7 if you're trying to set up a descending index. If false and with table level < 7 an exception will be raised when trying to set up a descending index.

