

## DBFTable unit

This unit holds a TTable descendant that adds some methods to handle missing or out of date MDX and DBT files with dBase tables. As it uses some declarations from [BDEDoRxS](#) that unit has to be in the search path as well when using TDBFTable.

### Components

[TDBFTable](#)

### Types

[TDBFAtDBTMissing](#)

[TDBFAtMDXMissing](#)

[TDBFOpenFailType](#)

[TDBFOpenFailure](#)

## TDBFTable Component

[Properties](#)

[Events](#)

**Unit**

[DBFTable](#)

### Description

TDBFTable installs an InputRequest callback with the BDE before trying to open a table. The callback is triggered when either the MDX file or the DBT file is missing or out of date. By setting the [AtDBTMissing](#) and [AtMDXMissing](#) properties you can determine how to handle these errors. In any case the event [OnOpenFailure](#) will be triggered to inform you of the error. For a short notice on the reliability of dBase indices cf [BDEDoRxS](#) help file.

## Properties

▶ Run-time only

🔑 Key properties

AtDBTMissing

AtMDXMissing

AutoSaveChanges

## Events

 Key events

[OnOpenFailure](#)

## AtDBTMissing property

### Applies to

[TDBFTable](#)

### Declaration

property AtDBTMissing: [TDBFAtDBTMissing](#);

### Description

This property tells TDBFTable how to handle missing DBT files.

## AtMDXMissing property

### Applies to

TDBFTable

### Declaration

```
property AtMDXMissing: TDBFAtMDXMissing;
```

### Description

This property tells TDBFTable how to handle missing MDX files.

## OnOpenFailure event

### Applies To

TDBFTable

### Declaration

```
property OnOpenFailure: TDBFOpenFailure;
```

### Description

This event will be triggered if either the DBT or the MDX file associated with the table is missing or out of date. You can then take appropriate measures like rebuilding the index file.

## TDBFAtDBTMissing type

### Unit

[DBFTable](#)

### Declaration

TDBFAtDBTMissing = (atdOpenWithout,atdOpenError,atdRemoveFields);

### Description

Type of the AtDBTMissing property. Meaning of the different values are:

atdOpenWithout: the table will be opened without the DBT file, i.e. any memo fields are not visible.

atdOpenError: an exception will be raised by the BDE after trying to open the table.

atdRemoveFields: the fields held in the DBT file will be removed from the table.

## TDBFAtMDXMissing type

### Unit

[DBFTable](#)

### Declaration

```
TDBFAtMDXMissing = (atmOpenReadOnly, atmOpenError, atmDetachMDX);
```

### Description

Type of the AtMDXMissing property. Meaning of the different values are:

atmReadOnly: the table will be opened in read only mode.

atmOpenError: an exception will be raised by the BDE after trying to open the table.

atmDetachMDX: the link between the DBF file and the MDX file will be cleared and the table will be opened. This is IMO the best choice to recreate the index on the fly.



## TDBFOpenFailType type

### Unit

[DBFTable](#)

### Declaration

```
TDBFOpenFailType = (ofNone, ofMDXMissing, ofDBTMissing);
```

### Description

This type is used by [OnOpenFailure](#) to inform of the kind of error that occurred when trying to open the table.

## TDBFOpenFailure type

### Unit

[DBFTable](#)

### Declaration

```
TDBFOpenFailure = procedure(Sender: TObject;  
                           FailType: TDBFOpenFailType) of object;
```

### Description

This is the prototype of the [OnOpenFailure](#) event.

## AutoSaveChanges Property

Unit

[DBFTable](#)

### 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. With local databases switching off the write behind cache is recommended.

Please note that the BDE reacts differently here and as far as my tests indicated the buffer will not be flushed directly to disk as with Paradox tables. It's in here in case a later BDE version will implement this differently.

