TopSpeed Database Recovery Utility

The TopSpeed file system is designed to automatically repair most errors. If the data file is physically damaged during a system malfunction, the TopSpeed Database Recovery Utility can recover the undamaged portions of your data.

> **Note: The TopSpeed Database Recovery Utility is an emergency repair tool and should not be used on a regular basis. Use it only when a file has been damaged.**

The TopSpeed Database Recovery Utility reads the damaged file and writes the recovered records to a new file. It uses the information stored in the file's header or scans the file recovering undamaged portions. Optionally, you can provide an example file containing table (individual file) and key layout.

The TopSpeed Database Recovery Utility is a freely distributable utility designed to enable your end users to recover damaged files.

The recovery utility is designed to work interactively or transparently via command line parameters. Interactively, you can use the utility to recover damaged files and provide the parameters via two wizard dialogs. Using the command line parameters, you can incorporate it in your application using a RUN() statement or create a shortcut (in Windows 95) or Program Manager Icon (in Windows 3.1x) with the parameters to enable end users to recover data files.

Using the TopSpeed Database Recovery Utility Interactively

1. Start the utility by double-clicking on the TopSpeed Database Recovery Utility Icon In the Clarion for Windows 1.5 Program Group.

The **TopSpeed Database Recovery Utility** dialog appears. The utility consists of two wizard dialogs.

2. In the **Source** (file to recover) section, specify the file name or press the **Browse** button to select it from a standard file open dialog.

3. If the file has a password, type it in the **Password** entry box.

If the database file contains multiple tables (data files), each table *must* have the same password.

4. Optionally, in the **Destination** (result file) section, specify the file name for the target file or press the **Browse** button to select it from a standard file open dialog.

By default the .TPR extension is added to the source file name. This parameter is optional. If omitted, the original (source file) is overwritten and a backup file is created. The source file is renamed to *filename.*TPx, where x is automatically incremented from 1 to 9 each time a new file is created. If all nine numbers are used, any subsequent files created are given the extension .TP$ and are overwritten.

5. If the result file is to have a different password, type it in the **Password** entry box. If omitted, the password is removed.

6. Press the **Next** button.

The second wizard dialog for the TopSpeed Database Recovery Utility appears.

7. Optionally, specify the **Example File** file name or press the **Browse** button to select it from a standard file open dialog.

The utility uses the Example File to determine table layouts and key definitions in the event those areas of the source file are damaged. The default extension is.TPE, but if you choose, you may use any valid DOS extension

Tip: We recommend shipping an example file when you deploy your application. This improves data recovery from a damaged file.

8. If the example file has a password, type it in the **Password** entry box.

9. If you want the utility to rebuild Keys, check the **Build Keys** box.

If omitted, the keys are rebuilt by the original application when it attempts to open it.

10. If you want to use the Header Information in the source file, check the **Use Header** box.

Utilizing Header Information optimizes the utility's performance, but should not be used if the file header is corrupt. If omitted, the utility searches the entire data file and restores all undamaged pages.

11. If the application uses a Locale (.ENV) File for an alternate collating sequence, specify the .ENV file or press the **Browse** button to select it from a standard file open dialog.

12. If the file is using the OEM attribute to control the collating sequence, Check the **Use OEM** box.

This enables the OEMTOANSI and ANSITOOEM conversion.

13. Press the **Start** button to begin the recovery process.

If the utility does not find any errors, a message appears informing you that "No Errors Detected in <fliename.ext>" and asks if you want to continue with recovery.

Command Line Parameters

The utility can also accept command line parameters which enables you to execute it from an application or Program Manager Icon (or Shortcut in Windows 95).

TPSFIX sourcepath[?password] [destpath[?password]] [/E:examplepath[?password]] [/L:localepath] [/H] [/K] [/P] [/O]

sourcepath	The file name and path of the source (damaged) database file.
[?password]	The database file's password.
destpath	The file name and path of the recovered database file.
[?password]	The recovered database file's password.
/E:examplepath	The file name and path of the example database file.
[?password]	The example database file's password
/L:localepath	The Locale (.ENV) file used to specify an alternate collating sequence.
/H	If specified, the utility uses the header information in the source file.
/K	If specified, the utility rebuilds all keys for the database.
/P	If specified, the user is prompted for each parameter even if they are supplied on the command line.
/O	If specified, the file uses OEMTOANSI and ANSITOOEM to determine the collating sequence. See Internationalization in Chapter 10 of the Language Reference.

Using the Utility in your Application

There are some issues to consider before allowing the utility to run.

- The database file should NOT be open when running the utility. Ensure that the file is closed before allowing the user to start the utility.
- To prevent access during the recovery process is completed, the utility locks the file automatically.
- It is more efficient and safer to allow the application to rebuild the KEYs (by omitting the /K parameter in the recovery). It is also a good way to check the status of a recovery.

Running the TopSpeed Database Recovery Utility

There are basically two methods you can use from a RUN() statement: Using the first method, you omit the *destpath* parameter so the original (source) file is overwritten. This requires an Example file.

In the Application Generator:

1. In the **Actions** dialog for a button or menu item, choose *Run a Program* from the drop down list.
2. In the **Program Name** entry box, specify *TPSFIX.EXE.*
3. In the parameters entry box, specify the parameters (see *Command Line Parameters* above).

For Example:

```
TPSFIX.EXE Datafile.TPS /E:Example.TPE /H
```

In Embedded Source Code:

```
RUN('TPSFIX.EXE Datafile.TPS /E:Example.TPE /H')
```

This recovers the "datafile.TPS" file using the "Example.TPE" file as an example for the table and key layouts, does not rebuild the keys, and uses the header information in the original file. The original file is saved to a backup file with an extension of TP1 through TP9. Each time the utility is executed, the numeric portion of the extension is incremented.

The second method requires two lines of embedded source code but gives you control over the renaming process. You insert the source code in the Accepted Embed point for the Menu Item or button.

For example:

```
COPY(datafilelabel, 'Datafile.OLD')      !  copies the original file
                                       !  to Datafile.OLD
RUN(TPSFIX Datafile.OLD Datafile.tps /H) !  Runs the utility using
                                       !  renamed file as
                                       !  the source and the original
                                       !  name as the target
```

This copies the datafilelabel file to DATAFILE.OLD, recovers the file and writes it to DATAFILE.TPS using the header information in the original file.