1 FileStorage component

1.1 TFileStorage - Overview

Overview

The FileStorage component capable to upload and hold any data files within your Delphi/BCB forms (within the body of the EXE-program). If your software requires any additional files (.DLL's, WAV's, .TXT's etc), these files could be uploaded straight onto your form and be extracted from executable file at run-time (AutoExtract property). Also you can access to stored files directly from memory without extracting them to disk (see example).

You can use this component if you would like to make high integration of your app with stored files and get access to them at run-time or, if you would like to supply your customers with just one, single executable file!.

 To upload files onto the form, or manage already stored files at design-time - invoke the **FileStorage Designer** (double click on component or click on Files property).

See also

WavPlayer component.

1.2 Properties

1.2.1 AutoExtract

Applies to

FileStorage component.

Declaration

```
type
  TAutoExtract = class
  published
    property Enabled: Boolean;
    property ExtractTo: TExtractTo;
    property Overwrite: TFileStorageOverwrite;
    property RemoveOnTerminate: Boolean;
  end;

property AutoExtract: TAutoExtract;
```

Description

The AutoExtract is the TPersistent property which contains the list of sub-properties.

You may use these auto extracting features if your would like to extract your files on every start of your application. Make Enabled property True and specify the path where files should be stored. If you don't want to keep extracted files permanenetly on the hard drive and would like to kill all extracted files after closing of your program - set RemoveOnTerminate to True.

If you would NOT extract files automatically on program startup, set Enabled property to False and ignore other auto extracting features. Anyway you may use Extract method at any time.

Note

Since auto extracting features used by FileStorage on program startup only, you don't need to change anything from AutoExtract subproperties at run-time. You may set these properies at

© 1998-2001, UtilMind Solutions®

design-time only, all run-time modifications will senseless.

1.2.1.1 Enabled

Applies to
FileStorage component, as subproperty AutoExtract structure.

Declaration
```
property Enabled: Boolean;
```

Description
Controls whether FileStorage component should automatically extract all files on program startup.

If AutoExtract.Enabled is TRUE, FileStorage will automatically extract all files on program startup, to location specified in ExtractTo property.

If AutoExtract.Enabled is FALSE then FileStorage will NOT extract files on startup and all other subproperties of the AutoExtract property will ignored.

Note
You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by FileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

1.2.1.2 ExtractTo

Applies to
FileStorage component, as subproperty AutoExtract structure.

Declaration
```
type
  TExtractTo = class(TPersistent)
  published
    property CreatePath: Boolean;
    property ToplevelDir: TToplevelDir;
    property Subdirectory: String;
  end;

property ExtractTo: TExtractTo;
```

Examples of usage
1. If you would like to extract stored files to the Desktop - just make TopLevelDir property "tdDesktop". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and exctract files there - set ToplevelDir property as "tdCurrentDir" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own database\") then you don't need to use any toplevel directories, just set ToplevelDir property to "tdNone" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Note

© 1998-2001, UtilMind Solutions®

You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by FileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

1.2.1.2.1 CreatePath

Applies to
FileStorage component as subproperty of AutoExtract.ExtractTo.

Declaration
```
property CreatePath: Boolean;
```

Description
Controls whether FileStorage should automatically create path for exctractable files.

Examples of usage
1. If you would like to extract stored files to the Desktop - just make TopLevelDir property "tdDesktop". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and exctract files there - set ToplevelDir property as "tdCurrentDir" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own database\") then you don't need to use any toplevel directories, just set ToplevelDir property to "tdNone" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Note
You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by FileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

1.2.1.2.2 SubDirectory

Applies to
FileStorage component as subproperty of AutoExtract.ExtractTo.

Declaration
```
property SubDirectory: String;
```

Description
The value of the Directory property points the directory or subdirectory where will be stored your files on program startup.

Examples of usage
1. If you would like to extract stored files to the Desktop - just make TopLevelDir property "tdDesktop". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and exctract files there - set ToplevelDir property as "tdCurrentDir" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own

© 1998-2001, UtilMind Solutions®

database\") then you don't need to use any toplevel directories, just set ToplevelDir property to "tdNone" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Note

You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by FileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

1.2.1.2.3 TopLevelDir

Applies to

FileStorage component as subproperty of AutoExtract.ExtractTo.

Declaration

```
type
  TacToplevelDir = (tdCurrentDir, tdWindowsDir,
                    tdTempDir, tdSystemDir,
                    tdMediaDir, tdCursorsDir,
                    tdHelpDir, tdSamplesDir,
                    tdDesktop, tdProgramFiles,
                    tdMyDocuments, tdMyPictures,
                    tdAppData, tdNone);

property ToplevelDir: TToplevelDir;
```

Description

The ToplevelDir property specifies the directory where you would like to automatically extract files on program startup.

Examples of usage

1. If you would like to extract stored files to the Desktop - just make ToplevelDir property "tdDesktop". Leave SubDirectory string clean.

2. If you would like to extract files to current directory, create subdirectory named, let's say, "plugins" and exctract files there - set ToplevelDir property as "tdCurrentDir" and assign SubDirectory property as "plugins" (`SubDirectory := 'plugins'`). If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

3. If you would like to extract your files to non-system folder (for example, to "h:\databases\my own database\") then you don't need to use any toplevel directories, just set ToplevelDir property to "tdNone" and assign to SubDirectory "h:\databases\my own database" path. If you would like to create this subdirectory automatically even if it's does not exists - set CreatePath property to True.

Note

1. You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by FileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

2. Current directory (tdCurrentDir) is the same as ExtractFilePath(Application.ExeName).

3. If TopLevelDir = tdNone, the file will be extracted to absolute path specified in SubDirectory property.

© 1998-2001, UtilMind Solutions®

1.2.1.3 Overwrite

Applies to
FileStorage component, as subproperty AutoExtract structure.

Declaration
```
type
  TacFileStorageOverwrite = (owIfSizeDifferent,
                             owNever, owAlways);

property Overwrite: TFileStorageOverwrite;
```

Description
Controls whether FileStorage could overwrite already extracted files on program startup.

Note
You don't need to change anything from AutoExtract subproperties at run-time because of these auto extracting features used by FileStorage on program startup only. You may set this property at design-time only, all run-time modifications will senseless.

1.2.1.4 RemoveOnTerminate

Applies to
FileStorage component, as subproperty AutoExtract structure.

Declaration
```
property RemoveOnTerminate: Boolean;
```

Description
If RemoveOnTerminate property is True then all automatically extracted files will be deleted on program termination. If the target Subdirectory is specified, the FileStorage will also try to delete the subdirectory, if it is empty.

Remark
It will remove only files extracted per current application session, and will not delete files that was exist before extracting. If you wish to be sure that ALL files are stored on start and definitely deleted on termination — set Overwrite property to `owAlways`.

See also
Enabled and Overwrite properties; SubDirectory property of ExtractTo structure.

1.2.2 Count

Applies to
FileStorage component.

Declaration
```
property Count: Integer; // Read only !
```

Description
The Count is read-only property which holds the number of currently stored files.

© 1998-2001, UtilMind Solutions®

1.2.3 DataSize

Applies to
FileStorage component.

Declaration
```
property DataSize: Integer; // Read only !
```

Description
The DataSize is read-only property which holds the total size of all currently stored files.

1.2.4 Files

Applies to
FileStorage component.

Declaration
```
property Files: TStoredFiles; { Successor of TList. Contains the list of
                                TStoredFile objects }
```

Description
The Files property is the List which contains all currently stored files. Every item of this list is the file uploaded onto your form at design-time. All files in this list are TStoredFile objects.

If you would like to access stored file at run-time directly from memory you may use following **example**:

```
var
  StoredFile: TStoredFile;
  DataStream: TMemoryStream;
begin
  StoredFile := FileStorage1.Files[1];
  DataStream := StoredFile.Data; // Data: TMemoryStream
```

Another example (demonstrates how to retrieve the content of text file to Memo control):
```
begin
   TStoredFile(FileStorage1.Files[0]).Data.Position := 1; // Since Data
is the stream we must reset its position
   Memo1.Lines.LoadFromStream(TStoredFile(FileStorage1.Files[0]).Data);
end;
```

© 1998-2001, UtilMind Solutions®

1.3 Methods

1.3.1 Extract

Applies to
FileStorage component.

Declaration
```
function Extract(FileName, OutputFile: String): Boolean;
```

Description
The Extract method is intended to extract stored files from FileStorage component to disk.

FileName parameter specifies the exact (though case sensitive) file name of stored file.
OutputFile parameter specifies the location (directory) and the file name of output file.

Returns True if file has been successfully extracted.

Example
```
procedure TForm1.ExtractBtnClick(Sender: TObject);
begin
  Extract('MyFile.exe', 'c:\MyFolder\MyFile.exe');
end;
```

1.4 Events

1.4.1 OnExtract

Applies to
FileStorage component.

Declaration
```
procedure OnExtract(Sender: TObject; FileName: String; FileSize:
  Integer; var AllowExtracting: Boolean);
```

Description
The OnExtract event occurs when StoredFile is ready to be extracted.

Write an OnExtract event handler to determinate whether this file could be extracted. Set *AllowExtracting* parameter to False to pervent file from extracting.

Parameters
FileName file name of the file to extract.
FileSize size (in bytes) of the file to extract.
AllowExtracting determinees whether the file (specified by *FileName* parameter) could be extracted.

Example
```
procedure TForm1.FileStorageExtract(Sender: TObject;
   FileName: String; FileSize: Integer;
   var AllowExtracting: Boolean);
begin
  if UpperCase(ExtractFileExt(FileName)) = '.TXT' then
    AllowExtracting := False;
end;
```

© 1998-2001, UtilMind Solutions®

1.4.2 OnExtracted

Applies to

FileStorage component.

Declaration

```
procedure OnExtracted(Sender: TObject; FileName: String; FileSize:
  Integer; Successfully: Boolean);
```

Description

The OnExtracted event occurs once StoredFile has extracted.

Write an OnExtracted event handler to determinate whether this file has successfully extracted from FileStorage. If file has been sucessfully extracted (without write errors) then *Successfully* parameter will be True.

Parameters

FileName	file name of the extracted file.
FileSize	size (in bytes) of the extracted file.
Successfully	determinees whether the file (specified by *FileName* parameter) has successfully extracted.

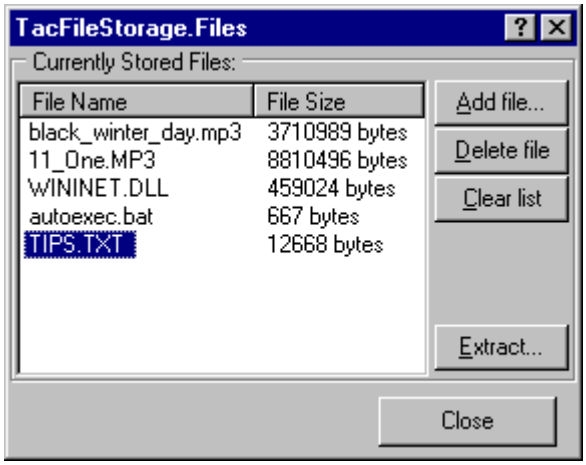
Example

```
procedure TForm1.FileStorageExtracted(Sender: TObject;
  FileName: String; FileSize: Integer; Successfully: Boolean);
begin
  if not Successfully then
   ShowMessage('Error !!');
end;
```

1.5 FileStorage designer

The FileStorage designer will help you to upload and organize your files at design-time.

Screenshoot



Yeah, all these files (>12 MB) currently uploaded on Delphi form!

© 1998-2001, UtilMind Solutions®

1.6 Playing .WAVs (example)

Following code demonstrates how to access stored files at run-time without extracting them to hard disk. This sample shows how to play stored .WAV file.

Delphi:

```
uses MMSystem;

procedure TForm1.Button1Click(Sender: TObject);
var
  StoredFile: TStoredFile;
  Data: TMemoryStream;
begin
  StoredFile := FileStorage1.Files[0];
  Data := StoredFile.Data;

  // the stored data could be retrieved from Data.Memory
  sndPlaySound(Data.Memory, SND_MEMORY or SND_ASYNC);
end;
```

C++ Builder:

```
#include <mmsystem.h>

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TStoredFile *StoredFile = (TStoredFile*)acFileStorage1->Files-
>Items[0];
  TMemoryStream *Data = StoredFile->Data;

  // the stored data could be retrieved from Data->Memory
  sndPlaySound((LPCTSTR)Data->Memory, SND_MEMORY|SND_ASYNC);
}
```

See also

Files property.

2 WavPlayer component

2.1 TWavPlayer - Overview

Overview

The acWavPlayer component could be used for playing the standard or custom Wave-Audio (*.wav) files.

The type of sound controlled by SoundType property. It could be a custom (stCustom) or standard sound.

Custom sounds can be stored (uploaded) onto your form in WaveSound property editor at design-time and does not requires additional file in the supply package of your software. WavPlayer component stores WAV's onto your form same way as it does the FileStorage component.

© 1998-2001, UtilMind Solutions®

Standard sounds is the sounds which associated with some system events (such as Windows Start or Program Minimize) and playing when any standard system event occurs.

How to use ?
To start playing the sound - call **Play** method. To terminate playing - call Stop method.

 To upload a custom wave-audio file onto form - click on WaveSound property or click right mouse button on the component image and select "Upload WAV sound..." menu item.
 To test the sound - double click on the component image.

See also
Other components which stores (uploads) files onto Delphi/BCB form at design-time:
FileStorage, WavPlayer.

2.2 Properties

2.2.1 Asynchroneous

Applies to
WavPlayer component.

Declaration
```
property Asynchronous: Boolean;
```

Description
The Asynchronous property controls whether sound will playing synchronously (if False) or asynchronously (if True, defauly), relatively to the main application thread. If Asynchronous is True, the sound will played asynchronously and the function returns immediately after beginning the sound. To terminate an asynchronously played sound, call Stop method.

2.2.2 Looped

Applies to
WavPlayer component.

Declaration
```
property Looped: Boolean;
```

Description
The Looped property controls whether sound sould be played once (if False) or constantly looped. To terminate looped sound use Stop method.

2.2.3 SoundType

Applies to
WavPlayer component.

Declaration
```
procedure SoundType: TWavPlayerSndType;
```

Description
The SoundType property specifies the type of a sound to play. When SoundType is stCustom, WavPlayer will play custom sound specified in (uploaded to) WaveSound property (.WAV file uploaded onto form). When SoundType is any other value (*stAsterisk, stCloseProgram, stCriticalStop, stDefaultSound, stExclamation, stExitWindows, stMaximize, stMenuCommand,*

© 1998-2001, UtilMind Solutions®

stMenuPopup, stMinimize, stNewMailNotification, stOpenProgram, stProgramError, stQuestion, stRestoreDown, stRestoreUp, stStartWindows), WavPlayer will play sound associated with enumerated system events.

Note
⚠ After selecting of any system sound, content of WaveSound property will be cleared.

See also
WaveSound property and TWavPlayerSndType type.

2.2.4 WaveSound

Applies to
WavPlayer component.

Declaration
```
property WaveSound: TStoredFile;
```

Description
The WaveSound property is the storage for custom wave-audio file. You may upload sound file at design-time, double clicking on WaveSound property in Object Inspector.

To test uploaded sound - double click on component image.

See also
FileStorage component.

2.3 Methods

2.3.1 Play

Applies to
WavPlayer component.

Declaration
```
procedure Play;
```

Description
The Play method plays a waveform sound specified either by uploaded onto form in WaveSound property or by system sound specified in SoundType property.

If Asynchronous property is True, sound will plays asynchronously and and the function returns immediately after beginning the sound.

Usually Play method initiates playing of the sound only once. However, If Looped property is True, sound will playing continuously. To terminate playing - call Stop method.

See also
Stop method.

2.3.2 Stop

Applies to
WavPlayer component.

Declaration

© 1998-2001, UtilMind Solutions®

```
procedure Stop;
```

Description
Call Stop method to terminate playing the sound.

See also
Play method.

2.4 Events

2.4.1 OnAfterPlay

Applies to
WavPlayer component.

Declaration
```
procedure OnAfterPlay: TNotifyEvent;
```

Description
The OnAfterPlay event occurs after playing the sound.

See also
OnBeforePlay event

2.4.2 OnBeforePlay

Applies to
WavPlayer component.

Declaration
```
procedure OnBeforePlay: TNotifyEvent;
```

Description
The OnBeforePlay event occurs before playing the sound.

See also
OnAfterPlay event

© 1998-2001, UtilMind Solutions®

3 Installation Instructions

Package without source code

to Delphi 2

1. Unzip files from "Delphi2" directory to your "Delphi 2\Lib" directory.
2. Start Delphi 2 IDE.
3. Select "Component \ Install..." menu item.
4. Press "Add" button and select "_FSReg.pas" file.
5. Rebuild library.

to Delphi 3

1. Unzip files from "Delphi3" directory and copy them to "Delphi 3\Lib".
2. Start Delphi 3 IDE.
3. Open "FileStorageD3.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 4

1. Unzip files from "Delphi4" directory and copy them to "Delphi 4\Lib".
2. Start Delphi 4 IDE.
3. Open "FileStorageD4.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 5

1. Unzip files from "Delphi5" directory and copy them to "Delphi 5\Lib".
2. Start Delphi 5 IDE.
3. Open "FileStorageD5.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 6

1. Unzip files from "Delphi 6" directory and copy them to "Delphi 6\Lib".
2. Start Delphi 6 IDE.
3. Open "FileStorageD6.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 7

1. Unzip files from "Delphi 7" directory and copy them to "Delphi 7\Lib".
2. Start Delphi 7 IDE.
3. Open "FileStorageD7.dpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 3

1. Unzip files from "BCB3" directory and copy them to "CBuilder3\Lib".
2. Start C++ Builder 3 IDE.
3. Open "FileStorageCB3.bpk" file.
6. Select "Project \ Make FileStorageCB3" menu item.
7. Select "Component \ InstallPackages" menu item.
8. Press "Add" button and select "FileStorageCB3.bpl" file.

to C++ Builder 4

1. Unzip files from "BCB4" directory and copy them to "CBuilder4\Lib".
2. Start C++ Builder 4 IDE.
3. Open "FileStorageCB4.bpk" file.
4. Install package to the components palette ("Install" button).

© 1998-2001, UtilMind Solutions®

to C++ Builder 5
1. Unzip files from "BCB5" directory and copy them to "CBuilder5\Lib".
2. Start C++ Builder 5 IDE.
3. Open "FileStorageCB5.bpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 6
1. Unzip files from "BCB6" directory and copy them to "CBuilder6\Lib".
2. Start C++ Builder 6 IDE.
3. Open "FileStorageCB6.bpk" file.
4. Install package to the components palette ("Install" button).

Source code
1. Uninstall / delete all previous (trial) instances of FileStorage.
2. Unzip files from "Sources" directory and copy them to "..\Lib" directory.
3. Run Delphi or ++ Builder IDE.
4. Select "Component \ Install..." menu item.
5. Press "Add" button and select "_FSReg.pas" file.
6. Rebuild library.

4 License Agreement

Copyright

The FileStorage component (software) is Copyright © 1998-2002, by Utilmind Solutions® (Utilmind). All rights reserved.
The authors - Utilmind Solutions® and Aleksey Kuznetsov (founder of Utilmind), exclusively own all copyrights to the Advanced Application Controls (AppControls) and all other products distributed by Utilmind Solutions®.

Liability disclaimer

THIS SOFTWARE IS DISTRIBUTED "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. YOU USE IT AT YOUR OWN RISK. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

Restrictions

You may not attempt to reverse compile, modify, translate or disassemble the software in whole or in part. You may not remove or modify any copyright notice or the method by which it may be invoked.

Operating license

Unregistered version

You may distribute the unregistered version of software freely, provided that all files are included and remain unmodified and that no extra files have been added to the package. You may not ask any money for the distribution. You may use the unregistered version of software free of charge for testing purposes, but if you want to use it for other purposes than testing - you have to register it with the author.

Registered version (single user license)

Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend.

You have the non-exclusive right to use registered version of the software only by a single person, on a single computer at a time. You may physically transfer the software from one computer to another, provided that the software is used only by a single person, on a single computer at a time. In group projects where multiple persons will use the software, you must purchase an individual license for each member of the group or purchase site license. Use over a "local area network" (within the same locale) is permitted provided that the software is used only by a single person, on a single computer at a time. Use over a "wide area network" (outside the same locale) is strictly prohibited under any and all circumstances.

Registered version (site/team license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team only in one location (building complex). If you purchase a site license, you may use the program in an unlimited number of your company's computers within this area.

Registered version (Educational site license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your educational organisation (school/college/university etc) in one location (building complex). If you buy a educational site license, you may use the program in an unlimited number of your edicational organisation's computers within this area.

Registered version (World-wide license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team world-wide. If your company has many branches even with thouthands of computers, world wide license covers them all.

Notes (clarification)
"Single-user license" means "single-developer license". "Site license" means that it can be used by any number of software developers within your company.
You can use purchased components in ANY number of your projects and deploy the "end-user" software to ANY number of your users/customers without any additional royalty fees. However you are not permitted to distribute the component itself (the source code or .dcu files of components).

Back-up and transfer
You may make one copy of the software solely for "back-up" purposes, as prescribed by international copyright laws. You must reproduce and include the copyright notice on the back-up copy.

Terms
This license is effective until terminated. You may terminate it by destroying the program, the documentation and copies thereof. This license will also terminate if you fail to comply with any terms or conditions of this agreement. You agree upon such termination to destroy all copies of the program and of the documentation, or return them to author.

Other rights and restrictions
All other rights and restrictions not specifically granted in this license are reserved by authors.

© 1998-2001, UtilMind Solutions®

5 Registration Information

FileStorage component is SHAREWARE. This means that you can try it out for free, but if you like it and want to use it you have to register it with the author. Before continue read and accept license agreement please.

The only difference between the unregistered and registered versions is that the registered one has not message box with remind to register and works without Delphi (C++ Builder) running. You can also purchase the source code, if you would like to have it, and be able to compile or modify the FileStorage on any 32bit version of Delphi or C++ Builder.

If you would like to use the FileStorage and receive full, unrestricted version, priority support or even source code — you have to purchase proper license.

All prices in US dollars. Registering entitles you to unlimited support via E-Mail, minor version updates indefinitely and major version updates for 6 month from date of purchase.

Registration types:

Full, unrestricted version without source code:

Single user license:

- **https://secure.element5.com/register.html?productid=140745 - $19,95**

Site license:

- https://secure.element5.com/register.html?productid=140746 **- $79,95**

Full version including 100% Source Code:

Single user license:

- **https://secure.element5.com/register.html?productid=140747 - $29,95**

Site license:

- https://secure.element5.com/register.html?productid=140748 **- $119,95**

Comments

1. ***Site license*** covers a single organisation in one location (building complex). If you buy a site license, you may use the software in unlimited number of your company's computers withing this area. Site license is very cost-effective if you have many computers (many software developers).

See license agreement for more details.

© 1998-2001, UtilMind Solutions®