Table of Contents

Foreword 0

Part I Introduction 10

Part II Components Overview 10

Part III Version History 13

Part IV Installation Instructions 16

Part V Registration Information 18

Part VI License Agreement 19

Part VII Contact 21

Part VIII Components 21

1 dcDiskScanner 21

TdcDiskScanner 21
Properties 22
Busy 22
FindAllFolders 22
Folder 23
Files 23
IncludeHiddenSubfolders 23
IncludeSubfolders 24
Matches 24
SearchAttributes 24
SearchMask 26
SearchSize 26
SearchTime 26
Suspended 27
ThreadPriority 27
UseIcons 28
WaitThread 28
Methods 29
Execute 29
Stop 29
Events 30
OnFileFound 30
OnScanDone 32
OnScanFolder 32
OnStopped 33
OnFileFound example 33

2 dcMultiDiskScanner 34

TdcMultiDiskScanner 34
Properties 35

© 1999-2001, UtilMind Solutions®

Busy ... 35
Files ... 35
ExcludeList ... 35
IncludeList ... 36
Matches ... 37
SearchAttributes ... 38
SearchSize ... 39
SearchTime ... 39
Suspended ... 40
ThreadPriority ... 40
UseIcons ... 41
WaitThread ... 41
Methods ... 42
Execute ... 42
Stop ... 42
Events ... 42
OnExcludingBegin ... 42
OnExcludingEnd ... 43
OnFileFound ... 43
OnFolderNotExist ... 45
OnScanDone ... 45
OnScanFolder ... 46
OnStopped ... 46
Example (IncludeList/ExcludeList) ... 47
Search Rules designers ... 47
TDiskScanList class ... 48
TDiskScannerPath class ... 49

3 dcFolderMonitor ... 50

TdcFolderMonitor ... 50
Properties ... 50
Active ... 50
Filter ... 51
Folder ... 51
ThreadPriority ... 52
Events ... 52
OnChange ... 52

4 dcFileOperations ... 53

TdcFileOperations ... 53
Properties ... 53
DestFolder ... 53
FileList ... 54
Operation ... 54
Options ... 55
ProgressTitle ... 56
Methods ... 56
Execute ... 56
Copy ... 56
Delete ... 56
Move ... 57
Rename ... 57
Events ... 58
OnError ... 58
Example ... 58

© 1999-2001, UtilMind Solutions®

5 dcFileAssociation 58

TdcFileAssociation **58**
Properties **60**
AccessMode 60
DefCommandName 60
ExecutableFile 61
ExtDescription 61
EXTENSION 61
FileDescription 62
IconIndex 62
IconLocation 62
LargeIcon 63
MIMEType 63
ParamString 63
SmallIcon 64
Methods **64**
InstallFileType 64
UninstallFileType 65
dcFileAssociation example **65**

6 dcVirtualDrives 66

TdcVirtualDrives **66**
Properties **67**
Drive 67
Target 67
UnusedDrives 67
VirtualDrives 68
Methods **68**
AddDrive 68
DeleteDrive 69
GetTargetByDrive 69

7 dcDiskInfo 70

TdcDiskInfo **70**
Properties **70**
AvailableDrives 70
BytesPerSector 71
Disk 71
DiskFree 71
DiskSize 72
DriveType 72
FileSystem 72
FreeClusters 73
LargeIcon 73
SectorsPerCluster 73
SerialNumber 73
SerialNumberStr 74
SmallIcon 74
SysImageIndex 74
TotalClusters 75
VolumeLabel 75
Example **75**

8 dcFileInfo 76

TdcFileInfo **76**
Properties **76**

© 1999-2001, UtilMind Solutions®

DisplayName ... 76
FileAttributes ... 77
FILENAME ... 77
FileSize ... 78
FileType ... 78
LargeIcon ... 78
SmallIcon ... 79
SysImageIndex ... 79
TimeCreated ... 79
TimeLastAccessed ... 80
TimeLastModified ... 80

9 dcFileCRC ... 80

TdcFileCRC ... 80
Properties ... 81
CRC ... 81
FILENAME ... 81
Method ... 81

10 dcFileTail ... 82

TdcFileTail ... 82
Properties ... 82
FileName ... 82
FileSize ... 83
LastLines ... 83
LastLinesCount ... 83
MonitorType ... 83
ThreadPriority ... 84
Methods ... 85
CheckChanges ... 85
Events ... 85
OnFileAppended ... 85
OnFileChanged ... 86

11 dcVersionInfo ... 86

TdcVersionInfo ... 86
Properties ... 87
Comments ... 87
CompanyName ... 87
FileDescription ... 88
FileFlags ... 88
FileFlagsMask ... 88
FileMajorVersion ... 88
FileMinorVersion ... 89
FILENAME ... 89
FileVerBuild ... 90
FileVerMajor ... 90
FileVerMinor ... 90
FileVerRelease ... 91
FileVersion ... 91
FileVersionFloat ... 91
InternalName ... 92
LanguageName ... 92
LegalCopyright ... 92
LegalTrademark ... 92
OriginalFilename ... 92

© 1999-2001, UtilMind Solutions®

ProductMajorVersion ... 93
ProductMinorVersion ... 93
ProductName ... 93
ProductVersion ... 93
ProductVersionFloat ... 94
TranslationValue ... 94

12 dcShellIcon ... 94

TdcShellIcon ... 94
Properties ... 95
DisplayName ... 95
FILENAME ... 95
FileType ... 95
LargeIcon ... 96
Options ... 96
SmallIcon ... 96
SysImageIndex ... 97

13 dcShellLink ... 97

TdcShellLink ... 97
Properties ... 98
FILENAME ... 98
HotKey ... 98
IconIndex ... 99
IconLocation ... 99
LargeIcon ... 99
LinkTarget ... 99
Parameters ... 100
RunAs ... 100
ShortDescription ... 100
SmallIcon ... 101
WorkingDirectory ... 101
Methods ... 101
LoadFromFile ... 101
SaveToFile ... 102
ShellLink example ... 102

14 dcShellProperties ... 102

TdcShellProperties ... 102
Properties ... 103
FileName ... 103
MenuAlignment ... 104
MenuOptions ... 104
Methods ... 105
ShowContextMenu ... 105
ShowContextMenuByFile ... 106
ShowProperties ... 107
ShowPropertiesByFile ... 108
Events ... 108
OnAfterMenuItemClick ... 108
OnBeforeMenuItemClick ... 108
OnDelete ... 109
OnRename ... 109

15 dcFolderListView ... 110

TdcFolderListView ... 110
Properties ... 111

© 1999-2001, UtilMind Solutions®

Columns ... 111
Attributes ... 111
Alignment ... 112
Caption ... 112
Visible ... 112
Width ... 113
FileType ... 113
Modified ... 113
Name ... 114
Size ... 114
Cursors ... 115
Border ... 115
Divider ... 115
Header ... 116
ScrollHorz ... 116
ScrollVert ... 116
FileAttributes ... 116
FileMask ... 117
Folder ... 118
FolderMonitor ... 118
HeaderMenu ... 119
AscendingItem ... 119
DescendingItem ... 120
PopupMenu ... 120
HeaderStyle ... 120
KBStr ... 121
Options ... 121
RegistrySaver ... 122
Enabled ... 122
RegKey ... 123
RegLocation ... 123
ShowAllFolders ... 123
ShowFileExtensions ... 124
ShowHeaderMenu ... 124
ShowSortMark ... 124
ShowToolTips ... 125
SortColumn ... 125
SortDirection ... 125
SortMarkAlign ... 126
SortMarkKind ... 127
ToolTipOptions ... 127
Methods ... 128
Back ... 128
CreateFolder ... 128
CreateShortcut ... 129
DeleteSelectedFiles ... 129
FileAttributesByListItem ... 130
FileNameByListItem ... 130
FileSizeByListItem ... 131
FileTimeByListItem ... 131
FileTypeByListItem ... 131
GetColumnAt ... 132
GetItemAt ... 132
IsBackPossible ... 133

© 1999-2001, UtilMind Solutions®

IsScrollBarsVisible ... 133
ListItemByFileName ... 133
Refresh ... 134
ShowFolderProperties ... 134
ShowItemProperties ... 135
SelectAll ... 135
UnselectAll ... 135
Events ... 135
OnBeginUpdate ... 135
OnDrawHeaderSection ... 136
OnEndUpdate ... 136
OnFolderChanged ... 137
OnMouseEnter ... 138
OnMouseLeave ... 138
OnSettingsChanged ... 138
OnToolTip ... 139
16 dcFileEdit ... 139
TdcFileEdit ... 139
Properties ... 139
Align ... 139
Alignment ... 140
AutoSelect ... 141
Button ... 141
Cursor ... 141
Flat ... 142
Glyph ... 142
NumGlyphs ... 143
Hint ... 143
Width ... 143
Visible ... 144
Color ... 144
ColorDisabled ... 144
CursorBorder ... 144
DlgDefaultExt ... 145
DlgFilter ... 145
DlgOptions ... 145
DlgTitle ... 145
DlgType ... 146
Events ... 146
OnButtonClick ... 146
OnDlgOk ... 146
OnDlgCancel ... 147
OnMouseEnter ... 147
OnMouseLeave ... 147
17 dcFolderEdit ... 147
TdcFolderEdit ... 147
Properties ... 148
Align ... 148
Alignment ... 149
AutoSelect ... 149
Button ... 149
Cursor ... 150
Flat ... 150

© 1999-2001, UtilMind Solutions®

Glyph ... 150
NumGlyphs ... 151
Hint ... 152
Width ... 152
Visible ... 152
Color ... 152
ColorDisabled ... 153
CursorBorder ... 153
DlgOptions ... 153
DlgSpecialLocation ... 154
DlgStatusText ... 154
DlgTitle ... 155
Events ... 155
OnButtonClick ... 155
OnDlgOk ... 155
OnDlgCancel ... 156
OnMouseEnter ... 156
OnMouseLeave ... 156
18 dcOpenDialog ... 156
TdcOpenDialog ... 156
Properties ... 157
DefaultExt ... 157
FileName ... 157
Files ... 157
Filter ... 158
FilterIndex ... 158
InitialDir ... 159
InitialDirSaver ... 159
Options ... 160
Title ... 161
Methods ... 161
Execute ... 161
19 dcSaveDialog ... 161
TdcSaveDialog ... 161
Properties ... 161
DefaultExt ... 161
FileName ... 162
Files ... 162
Filter ... 163
FilterIndex ... 164
InitialDir ... 165
InitialDirSaver ... 166
Options ... 167
Title ... 168
Methods ... 168
Execute ... 168
20 dcBrowseDialog ... 168
TdcBrowseDialog ... 168
Properties ... 169
BrowseFor ... 169
Folder ... 170
Options ... 170
NewFolder ... 170

© 1999-2001, UtilMind Solutions®

Caption ... 171
Prompt ... 172
Visible ... 172
Position ... 173
SpecialLocation ... 173
StatusText ... 174
Title ... 174
Methods ... 174
Execute ... 174
Events ... 175
OnChanged ... 175
OnInitialized ... 175
OnSelected ... 175
21 dcSystemImageList ... 176
TdcSystemImageList ... 176
Properties ... 177
ImageSize ... 177
22 dcIcon2Bitmap ... 177
TdcIcon2Bitmap ... 177
Properties ... 177
Bitmap ... 177
Icon ... 178
Index 179

© 1999-2001, UtilMind Solutions®

1 Introduction

Disk Controls is set of 22+ top quality components which can make your life much easier if you developing software that works with disks, shell and file system.

DiskControls pack contains:

- Two advanced search engines, which will find any file or folder by specified criterias (*DiskScanner* and *MultiDiskScanner* components);
- The *FolderListView* component which let's you to display filtered contents of any directory and looks and feels like Windows Explorer. Supports big number of features such like automatical sorting of the list items with arrow-style sort mark on the header section, individual context menus for every shell object, possibility to hide or show some list columns, automatic drag'n'drop support via OLE and so on…
- The *FolderMonitor*, which will check your folders and sub-folders for any changes;
- The *VirtualDrives* utility, which can define, redefine or delete the virtual drives from the file system;
- The *FileTail* utility, which works like tail command in Unix, and can monitor the file for its changes;
- Several components that provide you with detailed information about any shell objects (files, folders or drives), can retrieve the version information from executable files and dynamic-link libraries;
- Component which can install or uninstall new file types into the shell (*dcFileAssociation*);
- Component which performs the operations with files and folders on shell level (*dcFileOperations*);
- Several advanced edit controls and dialogs for browsing and selecting the files and folders;
- *much more...* see **Components Overview** for more details.

Compatibility

DiskControls compatible with Delphi 3/4/5/6//7/2005 and C++ Builder 3/4/5/6 and has been tested on Win95, Win95OSR2, Win98, WinME, NT4, Win2K and WinXP.

Improvements

Actually the DiskControls pack contains 22 components. However, we contantly updating the package and adding more useful utilities and new great features in existing components. If you have any suggestions to help us to improve the Disk Controls — please **contact us** and never hesitate to do it for any question.

Updates

Registering of the DiskControls entitles you to unlimited support (via email or personal panel at AppControls.Com website), minor updates indefinitely and major version updates for 1 year from date of purchase.

WWW

Please visit the DiskControls homepage at **http://www.appcontrols.com/diskcontrols.html**

Disk Controls pack (http://www.appcontrols.com/diskcontrols.html)
Copyright © 1999-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual** help authoring tool.

2 Components Overview

1. **dcDiskScanner** - extremely easy to use and powerful threaded search engine that scans the

drives and folders for files (specified by wildcarded mask) and / or directories that matches to specified searching criteria. To find files — just point the starting folder + wildcarded searching mask, and DiskScanner will return the matches by specified criteria with lightning-fast speed. You may also specify desired file attributes, size and time range, search or not the files in subfolders and hidden subfolders. Besides file names it can return the file type, date and time of last modification, file icons (32x32 and 16x16).

2. **dcMultiDiskScanner** - The dcMultiDiskScanner component is enhancement of the dcDiskScanner component. Used to search files in multiple locations with multiple searching masks. MultiDiskScanner operates with two lists — *IncludeList* and *ExcludeList* that describes desired and unwanted searching paths and file masks.
For example, you would like to find all files in the "c:\windows\" directory, including all subfolders except "c:\windows\system". Just put the "c:\windows*.*" path to the *IncludeList*, point that this path should include subfolders, and put "c:\windows\systems*.*" to the *ExcludeList*.

3. **dcFileOperations** - allows to perform the shell operations with files and folders, like copying, moving / renaming and deleting the shell object(s). Every process execute the shell dialog box which amuses users displaying an progress dialog.with according animated picture. These operations uses high level functions of the Windows Explorer so, for example, "Delete" command moves files to the "Recycled bin" instead of erasing them permanently.

4. **dcFolderMonitor** - watch specified folder for changes and fires special notification event if any changed occurs in any file or the directory structure.

5. **dcVirtualDrives** - lets an application to determinate, define, redefine or delete virtual drives (MS-DOS device names) from the file system. (See demo program at http://www.appcontrols.com/software/virtualdrives.html)

6. **dcDiskInfo** - provides detailed information about system disk (Hard, Floppy, CD, RAM or Network drives). Returns the letters of available disks (ie. ACD), disk capacity and free space, file system information, serial number and shell icons (large and small), associated with specified disk.

7. **dcFileInfo** - provides detailed information about any file — size, type, shell name, creation and last modification dates, small and large shell icons. You can get the information even from opened file and specify/change the date/time when file was created. last modified or last accessed.

8. **dcFileCRC** - calculates a CRC (Cyclic Redundency Check) for any files. Supports three methods of checking: CRC32 (most frequently used method, used in such programs like ZIP, RAR, ARJ etc), CRC16 (applicable to XModem protocol), CRCArc (used by SEA's "ARC" utility).

9. **dcFileTail** - utility component with functionality similar to the *tail* command in Unix. It monitors files for their changes and triggers the event where program can receive appended part of data. Also, like the tail command of Unix it can return specified number of text lines from the end of file (very useful to receive some last entries from log-files).

© 1999-2001, UtilMind Solutions®

10. **dcVersionInfo** - used to extract version information from a file.

11. **dcShellIcon** - retrieves the icon images (32x32 and 16x16) for any specified shell object (file, folder or disk). It can determinate the real icons for shortcut files (without link overlays), or otherwise - put link mark to the shell icons . Able to show the shell object both in opened and closed state (for example, opened and closed folder icons: or), or even in selected state .

12. **dcShellLink** - creates the shell links (shortcut files) and provides detailed information about any existing shortcuts.

13. **dcShellProperties** - displays the "Properties" dialog box or context menu for the shell objects (file, folder or disk).

14. **dcFileAssociation** - lets an application to read, modify, install and uninstall the file types (extension associations) in the Windows shell. Component provides with detailed information about any file types installed to the system (ie. pas, .wav, .doc, .gif, .html), determines location to executable file, argument string, icon file, icon index and two icon images (32x32 and 16x16) associated with specified file type.
Also, with dcFileAssociation you can install new file types or modify existing, changing the location of executable file or an icon image, associated with specified file type. For example, you can install ".yourdatafile" extension, specify the executable program for this file type and point the file icon that will associated with this file extension in the shell. After opening the any file with ".yourdatafile" extension (clicking the file icon), the application associated with this file type will be executed to process some data from ".yourdatafile".

15. **dcFolderListView** - the ListView control for displaying contents of any directory. The directory contents can be filtered by file attributes or multiple file masks. User can browse folders, execute files, perform major file operations like in standard shell list view control used in Windows Explorer. dcFolderListView supports big number of additional features like automatical sorting of the list items with arrow-style sort mark on the header section, individual context menus for every shell object, possibility to hide or show some list columns, automatic drag'n'drop support via OLE (user can drop files from Windows Explorer and vice versa) + a lots of properties and methods for complete customization.

16. **dcFileEdit** - enchaned TEdit, with button at the right side of control which automatically executes the file-selection dialog and returns result in the edit control.

17. **dcFolderEdit** - edit control with button at the right side, used to select the folder from "Browse for Folder" dialog. See also dcBrowseDialog component.

18. **dcOpenDialog** - enhancement of standard TOpenDialog. dcOpenDialog can remember last used folder in the system registry.

19. **dcSaveDialog** - enhancement of standard TSaveDialog. dcSaveDialog can remember last

© 1999-2001, UtilMind Solutions®

used folder in the system registry.

20. **dcBrowseDialog** - makes an "Browse For Folders" dialog box available to your application. The purpose of the dialog box is to let user to select local / network folders *and files*. Also user can *create new folders* using special button. Use the Execute method to display this dialog box.

21. **dcSystemImageList** - gives access to the system image list. That's the image list which all icons (32x32 or 16x16) for various file types that Explorer uses. All these icons represents the files within the Windows shell.

22. **dcIcon2Bitmap** - utility component that will help you to convert icon images (TIcon) to bitmaps (TBitmap). In example, it may be very useful if you would like to store bitmaps instead of icons, returned by various components of DiskControls pack, in the TImageList collection.

Disk Controls pack (http://www.appcontrols.com/diskcontrols.html)
Copyright © 1999-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual** help authoring tool.

3 Version History

30.05.2005 **3.8.3**
Fixes and optimizations in **dcThread** class, and **dcFolderMonitor** component.

15.12.2004 **3.8.2**
Support of Win32 platform of Delphi 2005.

28.06.2004 **3.8.1**
The IncludeList and ExcludeList of dcMultiDiskScanner component does not allow to add duplicate string anymore + allows to export and import the pathes to the TStringList.

27.05.2004 **3.8**
Bugfixes and optimizations.

21.05.2003 **3.7**
dcDiskScanner and **dcMultiDiskScanner** now supports UNC names and can search folders in local network.

dcFolderView supports XP themes.

Some minor code optimizations.

05.11.2002 **3.6**
Added *MonitorType* property and *CheckChanges* method to **dcFileTail** component. These improvements allows to choose the kind of monitoring for changes in specified file and "manually" check it for changes without using separate thread.

21.09.2002 **3.5**
Release for Delphi 7 and a couple of minor fixes.

24.08.2002 **3.4**

Added **dcFileTail**, utility component with functionality similar to the *tail* command in Unix.

A lots of improvements in **dcFolderListView** component.

07.09.2001 **3.3**

Improved **dcFileInfo** component. Now you can change also file attributes (*FileAttributes* property).

Added *MIMEType* property to **dcFileAssociation** component. The new property allows to specify and determinate the media type of file extension.

31.08.2001 **3.2**

Improved **dcBrowseDialog** and **dcFolderEdit** control. Added "Create new folder" button (*NewFolder* structure) to the browse-for-folder dialog.

Improved **dcFileInfo** component. Now you can also change the date/time when the file was created, accessed or modified (*TimeCreated*, *TimeLastAccessed* and *TimeLastModified* properties).

23.08.2001 **3.1**

Bufixes / corrections in *WaitThread* and *SearchAttributes* properties of **dcDiskScanner** and **dcMultiDiskScanner** components.

Fixed bug in **dcFileInfo** component (*FileSize* property now returns correct file size in BCB 4/5).

20.07.2001 **3.0**

Added 6 new components: **dcFolderListView**, **dcFolderMonitor**, **dcVirtualDrives**, **dcFileEdit**, **dcFolderEdit** and **dcSystemImageList**.

Completely redesigned the **dcDiskScanner** and **dcMultiDiskScanner** engines. Both components was optimized for speed + added some additional properties/events (*FindAllFolders* and *Files* properties to dcDiskScanner; *OnFolderNotFound* event to dcMultiDiskScanner; *UseIcons* property both to dcDiskScanner and dcMultiDiskScanner). Note that *OnFileFound* event is incompatible with previous versions due to additional parameters. (*FileTime* and *SysImageIndex*. The *FileTime* returns date/time of last modification and *SysImageIndex* is the icon index in the system image list (see **dcSystemImageList** component.)

Completely redesigned the **dcShellProperties** component. Now it's supports additional options of the context menu + contains events which allows to hook, handle and discard context menu commands.

The dcShellExtension component was slightly improved and renamed to **dcFileAssociation** + added new property, *ExtCommandName*.

Added *Directory*, *Temporary*, *Compressed* and *Offline* flags to the *FileAttributes* property and *TimeLastAccessed* property to **dcFileInfo** component.

Added *SysImageIndex* properties to the **dcFileInfo**, **dcDiskInfo** and **dcShellIcon** components. The *SysImageIndex* returns the index of icon in the system image list.

Changes in the **dcBrowseDialog** component: *Title* property renamed to the *StatusText*; *Caption* renamed to *Title*.

Improved **dcVersionInfo** component. Added 4 new properties which represents 16-bit values for

© 1999-2001, UtilMind Solutions®

version number: *FileVerMajor*, *FileVerMinor*, FileVerRelease, *FileVerBuild*.

a lots of others minor improvements...

06.06.2001 **2.9**

Added new component - **dcFileCRC**, which calculates a CRC (Cyclic Redundency Check) for any files. Supports three methods of checking: CRC32, CRC16 and CRCArc.

Improved the **dcDiskScanner** and **dcMultiDiskScanner** components. Added new features which allows you to specify the priority of the searching thread (*TheadPriority* property), lets to suspend and resume the searching "on the fly" at run-time (*Suspended* property), wait for the search results before continuing next steps of your program (*WaitThread* property).

Added two methods to the *TDiskScanList* class (which operates with Include and Exclude lists of the **dcMultiDiskScanner** component): SaveToFile and LoadFromFile. These methods allows to save and restore the searching rules from the flat-text files.

24.05.2001 **2.8**

Updated the **dcFileInfo**, **dcDiskScanner** and **dcMultiDiskScanner** components. Improvements allows to get the information from opened files.

Fixed small bug in the **dcFileInfo**, now it correctly return size for huge files over 2Gb. The type of FileSize property has been changed to Int64.

26.04.2001 **2.7**

Minor bugfixes and optimizations in the **dcDiskScanner** and **dcMultiDiskScanner** components.

19.03.2001 **2.6**

Added new component — **dcVersionInfo**, used to extract version information from a file.
Improved the **dcBrowseDialog.** Added two new properties which affects on dialog appearance, and two options which let users to browse either for folders and files at the same time.

The *dcClasses* unit renamed to *dcInternal* to avoid conflict with DreamCompany's libraries (which also uses "dc" prefix in module names).

23.02.2001 **2.5**

Minor improvements in the DiskScanner engine (in DiskScannerThread).
Component names changed from **T***ComponentName* to **Tdc***ComponentName*.
Added 8 new components:

- dcShellIcon
- dcShellLink
- dcShellProperties
- dcShellFileOperation
- dcShellExtension
- dcBrowseDialog
- dcIcon2Bitmap

25.09.2000 **2.1**

Some bugfixes and improvements at **dcDiskScanner** and **dcMultiDiskScanner**. Now they supports *files* with size more than 2 gigabytes (FileSize parameter of the OnFileFound event changed to Extended).

08.08.2000 **2.0**

DiskControls pack. Completely rewritten scanning mechanism. Added our old DiskInfo, FileInfo and DiskBrowser components (was distributed by UtilMind Solutions as separate

© 1999-2001, UtilMind Solutions®

freeware/shareware components since 1999). Package now contains 5 components.

24.06.2000 **1.5**
Improvements of MultiDiskScanner. Added IncludeList and ExcludeList properties. MultiDiskScanner now considering the searching file attributes, size and time same way as it DiskScanner does.

17.06.2000 **1.2**
Added MultiDiskScanner component which able to search files in multiple locations. Minor imrovements in the DiskScanner engine.

11.06.2000 **1.0**
Initial public release of the TDiskScanner component.

Disk Controls pack (http://www.appcontrols.com/diskcontrols.html)
Copyright © 1999-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual** help authoring tool.

4 Installation Instructions

Package without source code

to Delphi 3
1. Download "DiskControlsD3.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start Delphi 3 IDE.
5. Open "DiskControlsD3.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 4
1. Download "DiskControlsD4.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start Delphi 4 IDE.
5. Open "DiskControlsD4.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 5
1. Download "DiskControlsD5.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start Delphi 5 IDE.
5. Open "DiskControlsD5.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 6
1. Download "DiskControlsD6.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start Delphi 6 IDE.
5. Open "DiskControlsD6.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 7
1. Download "DiskControlsD7.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start Delphi 7 IDE.
5. Open "DiskControlsD7.dpk" file.
6. Install package to the components palette ("Install" button).

to Delphi 2005
1. Download "DiskControlsD2005.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start Delphi 2005 IDE.
5. Open "DiskControlsD2005.dpk" file.
6. Install package to the components palette (right-click on "DiskControlsD2005.bpl" node in the Project Manager and select "Install" menu item).

to C++ Builder 3
1. Download "DiskControlsCB3.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start C++ Builder 3 IDE.
5. Open "DiskControlsCB3.bpk" file.
6. Select "Project \ Make DiskControlsCB3" menu item.
7. Select "Component \ InstallPackages" menu item.
8. Press "Add" button and select "DiskControlsCB3.bpl" file.

to C++ Builder 4
1. Download "DiskControlsCB4.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start C++ Builder 4 IDE.
5. Open "DiskControlsCB4.bpk" file.
6. Install package to the components palette ("Install" button).

to C++ Builder 5
1. Download "DiskControlsCB5.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start C++ Builder 5 IDE.
5. Open "DiskControlsCB5.bpk" file.
6. Install package to the components palette ("Install" button).

to C++ Builder 6
1. Download "DiskControlsCB6.zip" file.
2. Create "..\Lib\DiskControls" directory.
3. Unzip files and copy them to "..\Lib\DiskControls".
4. Start C++ Builder 6 IDE.
5. Open "DiskControlsCB6.bpk" file.
6. Install package to the components palette ("Install" button).

Source Code
1. Uninstall / delete all previous (trial) instances of DiskControls.
2. Create "..\Lib\DiskControls" directory.

© 1999-2001, UtilMind Solutions®

3. Unzip files from "Sources" directory and copy them to "..\Lib\DiskControls".
4. Run Delphi IDE.
5. Select "Component \ Install..." menu item.
6. Press "Add" button and select "_DCReg.pas" file.
7. Rebuild library.

Disk Controls pack (http://www.appcontrols.com/diskcontrols.html)
Copyright © 1999-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual** help authoring tool.

5 Registration Information

DiskControls pack 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 DiskControls on any 32bit version of Delphi or C++ Builder.

If you would like to use the DiskControls 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 developer license:
- **https://secure.element5.com/register.html?productid=140012 - $59,95**

2 developers license (save 20%):
- https://secure.element5.com/register.html?productid=140013 - $94,95

3 developers license (save 25%):
- https://secure.element5.com/register.html?productid=140014 - $134,95

Site license:
- https://secure.element5.com/register.html?productid=140015 - $239,95

Educational site license:
- https://secure.element5.com/register.html?productid=140016 - $139,95

Full version including 100% Source Code:

Single developer license:
- **https://secure.element5.com/register.html?productid=140018 - $139,95**

2 developers license (save 20%):
- https://secure.element5.com/register.html?productid=140019 - $224,95

3 developers license (save 25%):
- https://secure.element5.com/register.html?productid=140020 - $314,95

Site license:
- https://secure.element5.com/register.html?productid=140022 - $559,95

Educational site license:
- https://secure.element5.com/register.html?productid=140023 - $329,95

Upgrade to source code (for those who have license without sources):

Single developer:

- **https://secure.element5.com/register.html?productid=140024 - $90**

Company/organisation (site license):

- https://secure.element5.com/register.html?productid=140025 - $399

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).
2. ***Educational site license*** — same as site license described above but for educational institutions (schools, colleges, universities etc).
3. ***World-wide license*** covers all computers of single organisation world-wide. If your company has many branches even with thousands of computers, world license covers them all. Please contact us if you would like purchase the world-wide license.

See license agreement for more details.

Prices is subject to change after every major update. All licenses is royalty free, so you can use purchased DiskControls in ANY number of your projects.

Disk Controls pack (http://www.appcontrols.com/diskcontrols.html)
Copyright © 1999-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual** help authoring tool.

6 License Agreement

Copyright

The Disk Controls (software) is Copyright © 1999-2005, by Utilmind Solutions® (Utilmind). All rights reserved.

The authors - Utilmind Solutions® and Aleksey Kuznetsov (founder of Utilmind), exclusively own all copyrights to the DiskControls 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

© 1999-2001, UtilMind Solutions®

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.

Disk Controls pack (http://www.appcontrols.com/diskcontrols.html)
Copyright © 1999-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual** help authoring tool.

7 Contact

If you have questions about our products or other concerns you may have, please email us to **info@appcontrols.com**. We will respond you as soon as possible (this usually take no more than 24 hours).

Please visit our Web site: **http://www.appcontrols.com** (updates, FAQs, user's forums)

Email is always quickest, but alternatively you can snail-mail us at:
UtilMind Solutions,
Teremkivska-7, k.34,
Kiev, 03187, UKRAINE

Disk Controls pack (http://www.appcontrols.com/diskcontrols.html)
Copyright © 1999-2005, Utilmind Solutions. All Rights Reserved.
Documentation created with **Help&Manual** help authoring tool.

8 Components

8.1 dcDiskScanner

8.1.1 TdcDiskScanner

Overview
The dcDiskScanner component is the extremely easy to use and powerful threaded search engine that scans the drives and folders for files (specified by wildcarded mask) and / or directories that matches to specified searching criteria. DiskScanner gives you full power of standard "Find Files or Folders" utility.

How to use ?
To search files — just point the starting folder + wildcarded searching mask, and DiskScanner will return the matches by specified criteria with lightning-fast speed. You may also specify desired file attributes, size and time range, search or not the files in subfolders and hidden subfolders.
To start searching — call Execute method, and Stop to terminate the scanning. Return values can be handled at OnFileFound event. Besides file names it can return the file type, date and time of last modification, file icons (32x32 and 16x16).

new! dcMultiDiskScanner now supports UNC names and can search files in local network. To search network files you should specify '**\\?**' prefix in the search path. For example, or example, "\\?\C:\myworld\private" is seen as "C:\myworld\private" (will search files in local machine), and

"\\?\UNC\bill_g_1\hotstuff\coolapps" is seen as "\\bill_g_1\hotstuff\coolapps" (search files in bill_g_1 machine, at "hotsfull\coolaps" folder, if it is shared).

See also
dcMultiDiskScanner component.

8.1.2 Properties

8.1.2.1 Busy

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
property Busy: Boolean; // not published and read-only property!
```

Description
The Busy is the public property which indicates whether the scanning currently in progress (whether the dcDiskScanner component is "Busy").

When the component is "Busy", you can not call the Execute method. The Execute function will always return False when component is "busy".

See also
Execute method.

8.1.2.2 FindAllFolders

Applies to
dcDiskScanner component.

Declaration
```
property FindAllFolders: Boolean;
```

Description
The FindAllFolders property controls whether you would like to find all directories, which even does not match to the SearchMask, SearchSize and SearchTime.

Set FileAllFolders and *saDirectory* flag in SearchAttributes to True, to find all folders within the search path.

If you don't want to find directories at all — just set *saDirectory* flag in SearchAttributes to False.

This property very useful in situation when you would like to find and show only files with certain type(s) + would like to show all directories (with any extension) withiin the scanning folder. FindAllFolders are similiar to ShowAllFolders property of the dcFolderListView component.

The dcDiskScanner will not find ANY folders if *saDirectory* flag of SearchAttributes property is False.

See also
SearchMask and SearchAttributes properties.
ShowAllFolders property of dcFolderListView component.

© 1999-2001, UtilMind Solutions®

8.1.2.3 Folder

Applies to
dcDiskScanner, dcFolderMonitor and dcBrowseDialog components.

Declaration
```
property Folder: String;
```

Description
The Folder property specifies the starting folder for scanning for files and folders. Default value is "c:\". To specify the searching mask — use SearchMask property. To search files in the subfolders — make IncludeSubfolders property True, or False to exclude the subfolders from search.

This property can be specified if only the folder is really exist. Otherwise, exception will raised. Use **try**..**except** blocks to handle an exception.

See also
SearchMask and IncludeSubfolders properties and Execute method.

8.1.2.4 Files

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
property Files: TStringList; // Read-only!!
```

Description
The Files property is the list of strings which represents the names of found files and directories.

You can use this list, instead of OnFileFound event, to get names of all found files on completion of scanning (OnScanDone event). However, the OnFileFound event can provide you with more detailed information about found file.

See also
OnFileFound and OnScanDone events;
Execute method.

8.1.2.5 IncludeHiddenSubfolders

Applies to
dcDiskScanner component.

Declaration
```
property IncludeHiddenSubfolders: Boolean;
```

Description
The IncludeHiddenSubfolders controls whether you would like to search files within a hidden subfolders (directories with "hidden" attributes, for example — "RECYCLED" on every hard disk). Set IncludeHiddenSubfolders to True, if you would like to search files in hidden subfolders or False otherwise.

See also
IncludeSubfolders property.

© 1999-2001, UtilMind Solutions®

8.1.2.6 IncludeSubfolders

Applies to
dcDiskScanner component.

Declaration
```
property IncludeSubfolders: Boolean;
```

Description
The IncludeSubfolders property controls whether you would like to search the files within a subdirectories of disk folder, specified by Folder property. Make this property True to include subfolders under the scanning path, or False to exclude.

For example, if starting folder is "c:\" and IncludeSubfolders is True, then after calling Execute method, dcDiskScanner component will search whole disk.

Note
Even if you set the IncludeSubfolders to True, the DiskScanner will not scan hidden subfolders. To search hidden subfolders — set IncludeHiddenSubfolders to True also.

See also
Folder and IncludeHiddenSubfolders properties, OnScanFolder event.

8.1.2.7 Matches

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
type
  TDiskScannerMatches = class
  published
    property Limited: Boolean; // False by default (unlimited)
    property MaxMatches: LongInt;
  end;
```

Description
The Matches property controls the maximum number of matches that returned by dcDiskScanner component in the OnFileFound event. If you would like to limit the number of search results — make **Limited** property True, and specify the maximum number of search matches in the **MaxMatches** property. To cancel the limitation — set **Limited** property to False.

Remember, that you can always terminate the scanning process calling the Stop method.

See also
SearchAttributes, SearchSize, SearchTime properties.

8.1.2.8 SearchAttributes

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
type
  TdcScanAttribute = (saNormal, saArchive, saReadOnly, saHidden,
saSystem, saDirectory, saAny);
```

© 1999-2001, UtilMind Solutions®

```
  TdcScanAttributes = set of TdcScanAttribute;

property SearchAttributes: TdcScanAttributes;
```

Description

The SearchAttributes property specifies the possible file attributes for files or directories that should be found by dcDiskScanner component.

Value	Meaning
saNormal	The file or directory has normal or no attributes set. If True, DiskScanner will find files even without any attributes set.
saArchive	The file or directory is an archive file or directory. This is standard file attributes. Applications use this flag to mark files for backup or removal. Set saArchive flag to True to find files with Archive attributes set. This is normal DOS attribute that means that file is archived or copied to hard disk. Make saArchive True to find files with Archive attributes set. This file attribute usually obsolete in win32 systems, because all files kept on hard drives and even if they copied to floppy or CD, this attribute not cleaned. Anyway, this file attribute still used on *most* files.
saReadOnly	Attribute means that file or directory have read-only permission. Applications can read the file but cannot write to it or delete it. In the case of a directory, applications cannot delete it. Set saReadOnly to True to find files with Read-only attributes set.
saHidden	The file or directory is hidden. It is not included in an ordinary directory listing. Make saHidden True to find hidden files (note that hidden directories will not be found, unless the saDirectory is specified). OS hide most files which destruction lead to system failure (such files like IO.SYS, MSDOS.SYS and so forth).
saSystem	The file or directory is part of, or is used exclusively by the operating system. Make saSystem True to find system files.
saDirectory	The "file or directory" is a directory (folder). Make saDirectory True to find directories (if saHidden is True, DiskScanner will also find hidden directories, i.e: "Recycled". To find ALL directories, which even does not match SearchMask — set FindAllFolders property to True).
saAny	All (or any) attributes is set. Set saAny property to True to find files or directories with ANY file attributes.

Example

Following example demonstrates how to set or remove attributes in the SearchAttributes property. Let's suppose, we would like to add or remove from search results "hidden" files…

Delphi

```
// to add attribute
dcDiskScanner1.SearchAttributes := dcDiskScanner1.SearchAttributes + [saHidden];
// to remove attribute
dcDiskScanner1.SearchAttributes := dcDiskScanner1.SearchAttributes - [saHidden];
```

C++ Builder

```
// to add attribute
dcDiskScanner1->SearchAttributes = dcDiskScanner1->SearchAttributes << saHidden;
// to remove attribute
dcDiskScanner1->SearchAttributes = dcDiskScanner1->SearchAttributes >> saHidden;
```

See also

FindAllFolders, Matches, SearchMask, SearchSize and SearchTime properties.

© 1999-2001, UtilMind Solutions®

8.1.2.9 SearchMask

Applies to
dcDiskScanner component.

Declaration
```
property SearchMask: String;
```

Description
The SearchMask property specifies a valid searching mask of files that you would like to find, which can contain wildcard characters (* and ?). You may specify entire file name or just part of it.

For example, to find ALL files — the SearchMask should be "*.*". To find only files with .EXE extension — SearchMask should be "*.exe". To find all executable files that begins by "a" letter, set SearchMask to "a*.exe".

new! The SearchMask can contain multiple masks, separated by semicolon (;). For example, if you would like to show only graphic files — set FileMask to "***.bmp;*.gif;*.jpg**".

See also
Folder property and Execute method.

8.1.2.10 SearchSize

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
type
  TSearchSize = class
  published
    property AnySize: Boolean; // True by default
    property MinSizeKB: Integer;
    property MaxSizeKB: Integer;
  end;
```

Description
The SearchSize is the list of properties that controls the minimum and maximum file size range for files that dcDiskScanner must found.

To specify the size range — make **AnySize** property False (AnySize = True means that DiskScanner component will find files with ANY size) and specify minimum and maximum sizes in kilobytes at **MinSizeKB** and **MaxSizeKB** properties.

For example, if you would like to find files with size at least 50 kilobytes and at most 100 kilobytes, make MinSizeKB = 50, MaxSizeKB = 100 and AnySize = False.

See also
Matches, SearchAttributes, SearchSize, SearchTime properties.

8.1.2.11 SearchTime

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration

© 1999-2001, UtilMind Solutions®

```
type
  TFindFilesByDateKind = (ffModified, ffCreated, ffAccessed);
  TSearchTime = class
  published
    property AnyTime: Boolean; // True by default
    property FindFiles: TFindFilesByDateKind;
    property SinceTime: TDateTime;
    property TillTime: TDateTime;
  end;
```

Description

The SearchTime is the list of properties that controls the time range for files that were created, modified or accessed during specific time period, specified by **SinceTime** and **TillTime** properties.

AnyTime property controls whether the DiskScanner should consider the file time and find files with certain date/time. If AnyTime is True, DiskScanner will found files with any time (the modification / creation / access time will does not matter).

FindFiles property controls what DiskScanner should consider:

ffModified	date and time of last modifications in the found file;
ffCreated	date and time of file creation;
ffAccessed	date when file was last accessed (used).

SinceTime and **TillTime** properties controls the period of file creation or last modification.

See also

Matches, SearchAttributes, SearchSize properties;
OnFileFound event.

8.1.2.12 Suspended

Applies to

dcDiskScanner and dcMultiDiskScanner components.

Declaration

```
property Suspended: Boolean;
```

Description

The Suspended property indicates whether the searching thread is currently suspended. You can change the property value at run-time suspending and resuming the search process again.

See also

Execute and Stop methods.

8.1.2.13 ThreadPriority

Applies to

dcDiskScanner, dcMultiDiskScanner, dcFolderMonitor and dcFileTail components.

Declaration

```
property ThreadPriority: TThreadPriority;
```

Description

The ThreadPriority property determines the thread's scheduling priority relative to other threads in the process.

© 1999-2001, UtilMind Solutions®

ThreadPriority indicates and specifies the priority of searching process relatively to the main application thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the TThread component, as defined in the following table. Windows schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle-Windows won't interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning

Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

See also

Suspended and WaitThread properties.

8.1.2.14 UseIcons

Applies to

dcDiskScanner and dcMultiDiskScanner components.

Declaration

```
property UseIcons: Boolean;
```

Description

The UseIcons property controls whether you would like to retrieve icons (32x32 and 16x16) from files found by DiskScanner component.

If UseIcons = True, the icon images will be retrieved and their handles will be returned by LargeIcon and SmallIcon parameters of the OnFileFound event. If UseIcons = False, dcDiskScanner will not retrieve icons and LargeIcon and SmallIcon will have NIL (NULL) values.

If you would not use icons returned by OnFileFound event — leave UseIcons property False, because process of retrieving of icons may significantly slow down searching!

However, even if UseIcons is False, DiskScanner will determinate the SysImageIndex parameter.

See also

OnFileFound event.

8.1.2.15 WaitThread

Applies to

dcDiskScanner and dcMultiDiskScanner components.

© 1999-2001, UtilMind Solutions®

Declaration

```
property WaitThread: Boolean;
```

Description

The WaitThread property controls whether the procedure that calls the Execute method of the dcDiskScanner component should be suspended and wait until the scanning process will be done.

Set the WaitThread to True, if you would like to execute dcDiskScanner so that the application does not continue with next lines of code after calling the Execute method. Your application will find all files before continuing to next step.

See also

Execute method and OnScanDone event;
ThreadPriority and Suspended properties.

8.1.3 Methods

8.1.3.1 Execute

Applies to

dcDiskScanner and dcMultiDiskScanner components.

Declaration

```
function Execute: Boolean; // returns False if Busy
```

Description

The Execute method begins the scanning process, starting the search thread. Every time when the DiskScanner found files that matches to specified searching criteria, OnFileFound event occurs. When DiskScanner finished scanning of folder and going to subfolder, OnScanFolder even occurs. When DiskScanner done searching — OnScanDone occurs.

To terminate the scanning process — call Stop method.

Note

If scanning already in progress but you call Execute method again, the function will returns False. To determinate whether the searching is in progress — use Busy property.

See also

Stop method; OnFileFound, OnScanFolder and OnScanDone events;
Suspended and WaitThread properties.

8.1.3.2 Stop

Applies to

dcDiskScanner and dcMultiDiskScanner components.

Declaration

```
procedure Stop;
```

Description

The Stop method terminates the scanning process, initiated by Execute method. After the termination of searching thread, OnStopped and OnScanDone events will occurs.

See also

Execute method;
OnStopped and OnScanDone events;

© 1999-2001, UtilMind Solutions®

Suspended and WaitThread properties.

8.1.4 Events

8.1.4.1 OnFileFound

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration

```
type
  TdcFileFoundEvent = procedure(Sender: TObject;
    FileName, FileType: String; FileSize: Extended;
    FileTime: TDateTime; FileAttributes: TdcScanAttributes;
    LargeIcon, SmallIcon: TIcon; SysImageIndex: Integer;
    TotalFiles: Integer; TotalSize: Extended) of object;

property OnFileFound: TdcFileFoundEvent;
```

Description
The OnFileFound event occurs when the dcDiskScanner component found the file or directory (if `saDirectory` specified in the SearchAttributes) that matches to specified criteria (SearchMask, SearchAttributes, SearchSize and SearchTime properties).

Use the OnFileFound event handler to take specific action once the DiskScanner found files, and, for example, add this file to the list with search results. Please check out an example of handling this event.

Parameter	Meaning
FileName	The name of found file with full path to this file. Use *ExtractFileName* and *ExtractFilePath* routines from SysUtils unit to split the file name and path.
FileType	The type of file extension, if it was registered in the shell. For example, "System File" for .SYS or "MS-DOS application" for .COM. If extension was not registered, return value will be "EXT file", where EXT is the file extension.
FileSize	Size of found file in bytes. Use *FloatToStr* function to convert this value to string.
FileTime	Time of last modification.
FileAttributes	The set of file attributes (Normal, Archive, ReadOnly, Hidden, System or Directory). See TdcScanAttributes type.
LargeIcon, *SmallIcon*	32x32 and 16x16 icons, associated with this file extension in the shell. If file has .EXE extension, icon will be extracted from executable file. If file is Icon, the Icon image will returned. These parameters will be NIL (NULL), if UseIcon property is False.
SysImageIndex	determines an icon index in the system ImageList. Use this parameter only if you're using dcSystemImageList component as source of images for the ListView. SysImageIndex will be retrieved even if UseIcon is False.
TotalFiles	Total number of found files per current search session.
TotalSize	Total size of all files that found in current search session. Use *FloatToStr* function to convert this value to string.

Notes
The *LargeIcon* and *SmallIcon* will be retrieved only if UseIcons property is True.

© 1999-2001, UtilMind Solutions®

Example

Following example demonstrates how to handle the OnFileFound event of dcDiskScanner or dcMultiDiskScanner component. Let's say, we have the `ResultsListView: TListView` which we would like to fill with the search results and `ImageList: TImageList` which contains the 16x16 images associated with the file types.

```
procedure TForm1.dcDiskScannerFileFound(Sender: TObject; FileName,
  FileType: String; FileSize: Extended; FileTime: TDateTime;
  FileAttributes: TdcScanAttributes;
  LargeIcon, SmallIcon: TIcon; SysImageIndex: Integer;
  TotalFiles: Integer; TotalSize: Extended);
const
  KBStr = 'KB';
var
  St: String;
  ListItem: TListItem;
begin
  StatusBar.Panels[1].Text := 'Files ' + IntToStr(TotalFiles);
  StatusBar.Panels[2].Text := 'Total size: ' + FloatToStr(TotalSize) + '
bytes';

  ListItem := ListView1.Items.Add; // adding new item
{** file name}
  ListItem.Caption := FileName;
{** file type}
  ListItem.SubItems.Add(FileType);
{** file size}
  if not (saDirectory in FileAttributes) then
    if FileSize <> 0 then
      St := FloatToStrF(Int(FileSize / 1024 + 1), ffNumber, 18, 0) +
KBStr
    else
      St := '0' + KBStr
  else
    St := '';
  ListItem.SubItems.Add(St);
{** file time}
   ListItem.SubItems.Add(DateTimeToStr(FileTime));

{** icon image }
  ListItem.ImageIndex := SysImageIndex;
  if saHidden in FileAttributes then
    ListItem.Cut := True; { ghosted icon for hidden files }
  St := LowerCase(ExtractFileExt(FileName));
  { adding shortcut overlay for .lnk and .url }
  if (St = '.lnk') or (St = '.url') then
    ListItem.OverlayIndex := 1;

{ //* Use code below if you don't want to use
  //* system images and set DiskScanner1.UseIcons to True
  ListItem.ImageIndex := ImageList1.Count;
  ImageList1.AddIcon(SmallIcon); }
end;
```

See also

OnScanFolder, OnScanDone events;
Execute, Stop methods.

© 1999-2001, UtilMind Solutions®

8.1.4.2 OnScanDone

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
type
  TdcScanDoneEvent = procedure(Sender: TObject;
    TotalFiles: Ingeger; TotalSize: Extended;
    ElapsedTimeInSeconds: Integer) of object;

property OnScanDone: TdcScanDoneEvent;
```

Description
The OnScanDone event occurs once the dcDiskScanner component completes the scanning process in the location specified by Folder property and within all subfolders under specified folder (if IncludeSubfolders property is True). Use this event handler to get the total number of found files (*TotalFiles* parameter), their total size (*TotalSize* parameter, use *FloatToStr* function to convert this value to string), + time that passes after starting of the disk scanning (*ElapsedTimeInSeconds* — time that passes after calling the Execute method).

The OnScanDone also occurs after calling of the Stop method, to return the total number of found files and their total size.

Example
```
procedure TForm1.DiskScannerScanDone(Sender: TObject; TotalFiles:
   Integer; TotalSize: Extended; ElapsedTimeInSeconds: Integer);
begin
  ShowMessage('Scanning done.'#13#10 +
              'Total files found by specified criteria: ' +
              IntToStr(TotalFiles) + #13#10 +
              'Total size of files: ' + FloatToStr(TotalSize) + '
              bytes'#13#10 +
              'Elapsed Time: ' + IntToStr(ElapsedTimeInSeconds) + '
              seconds.');
end;
```

See also
OnFileFound, OnScanFolder events;
Execute, Stop methods.

8.1.4.3 OnScanFolder

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
type
  TdcScanFolderEvent = procedure(Sender: TObject; Folder: String) of
object;

property OnScanFolder: TdcScanFolderEvent;
```

Description
The OnScanFolder event occurs when the dcDiskScanner component is about to search the subfolder that located under starting Folder, if IncludeSubfolders property is True.

© 1999-2001, UtilMind Solutions®

See also
Folder and IncludeSubfolders properties.
OnFileFound and OnScanDone events;
Execute and Stop methods.

8.1.4.4 OnStopped

Applies to
dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
property OnStopped: TNotifyEvent;
```

Description
The OnStopped event occurs after calling the Stop method, when user terminate the searching process. However, the OnScanDone event will occurs after OnStopped anyway, to report about search results.

See also
Stop method and OnScanDone event.

8.1.5 OnFileFound example

Following example demonstrates how to handle the OnFileFound event of dcDiskScanner or dcMultiDiskScanner component. Let's say, we have the `ResultsListView: TListView` which we would like to fill with the search results and `ImageList: TImageList` which contains the 16x16 images associated with the file types.

```
procedure TForm1.dcDiskScannerFileFound(Sender: TObject; FileName,
  FileType: String; FileSize: Extended; FileTime: TDateTime;
  FileAttributes: TdcScanAttributes;
  LargeIcon, SmallIcon: TIcon; SysImageIndex: Integer;
  TotalFiles: Integer; TotalSize: Extended);
const
  KBStr = 'KB';
var
  St: String;
  ListItem: TListItem;
begin
  StatusBar.Panels[1].Text := 'Files ' + IntToStr(TotalFiles);
  StatusBar.Panels[2].Text := 'Total size: ' + FloatToStr(TotalSize) + '
bytes';

  ListItem := ListView1.Items.Add; // adding new item
{** file name}
  ListItem.Caption := FileName;
{** file type}
  ListItem.SubItems.Add(FileType);
{** file size}
  if not (saDirectory in FileAttributes) then
    if FileSize <> 0 then
      St := FloatToStrF(Int(FileSize / 1024 + 1), ffNumber, 18, 0) + KBStr
    else
      St := '0' + KBStr
  else
    St := '';
```

© 1999-2001, UtilMind Solutions®

```
    ListItem.SubItems.Add(St);
  {** file time}
     ListItem.SubItems.Add(DateTimeToStr(FileTime));

  {** icon image }
    ListItem.ImageIndex := SysImageIndex;
    if saHidden in FileAttributes then
      ListItem.Cut := True; { ghosted icon for hidden files }
    St := LowerCase(ExtractFileExt(FileName));
    { adding shortcut overlay for .lnk and .url }
    if (St = '.lnk') or (St = '.url') then
      ListItem.OverlayIndex := 1;

  { //* Use code below if you don't want to use
    //* system images and set DiskScanner1.UseIcons to True
    ListItem.ImageIndex := ImageList1.Count;
    ImageList1.AddIcon(SmallIcon); }
  end;
```

8.2 dcMultiDiskScanner

8.2.1 TdcMultiDiskScanner

Overview

The dcMultiDiskScanner component is enhancement of the dcDiskScanner component Able to search files in multiple locations with multiple searching masks. MultiDiskScanner operates with two lists — *IncludeList* and *ExcludeList* that describes desired and unwanted searching paths and file masks.

For example, you would like to find all files in the "c:\windows\" directory, including all subfolders except "c:\windows\system". Just put the "c:\windows*.*" path to the *IncludeList*, point that this path should include subfolders, and put "c:\windows\systems*.*" to the *ExcludeList*.

How to use ?

Just set the search rules, what folders and / or search masks you would like to include (IncludeList) and which to exclude (ExcludeList) and call the Execute method to start find files in many specified locations. Found files will returned by OnFileFound event.

You can set the including rules either at design and run-time of your application.

new! dcMultiDiskScanner now supports UNC names and can search files in local network. To search network files you should specify '**\\?**' prefix in the search path. For example, or example, "\\?\C:\myworld\private" is seen as "C:\myworld\private" (will search files in local machine), and "\\?\UNC\bill_g_1\hotstuff\coolapps" is seen as "\\bill_g_1\hotstuff\coolapps" (search files in bill_g_1 machine, at "hotsfull\coolaps" folder, if it is shared).

How does it works ?

Once you call the Execute method, the MultiDiskScanner starts new searching thread. Scanning process separated on two stages:

Stage 1: Excluding

First of all, MultiDiskScanner should determinate all files and directories that does not satisfy to the searching rules, for further excluding them from the search results. When component starts handling of ExcludeList — OnExcludingBegin event occurs. When component done all excludings and ready to find files by criterias specified in IncludeList — OnExcludingEnd occurs.

Stage 2: Searching

© 1999-2001, UtilMind Solutions®

After completion of first, excluding stage, MultiDIskScanner scans all locations specified in the IncludeList. When it found the file, and this file is not previously excluded from search results — OnFileFound event occurs. When it passes to another folder — OnScanFolder event occurs. When scanning is complete — OnScanDone occurs.

See also
dcDiskScanner component.

8.2.2 Properties

8.2.2.1 Busy

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration
```
property Busy: Boolean; // not published and read-only property!
```

Description
The Busy is the public property which indicates whether the scanning currently in progress (whether the dcMultiDiskScanner component is "Busy").

When the component is "Busy", you can not call the Execute method. The Execute function will always return False when component is "busy".

See also
Execute method.

8.2.2.2 Files

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration
```
property Files: TStringList; // Read-only!!
```

Description
The Files property is the list of strings which represents the names of found files and directories.

You can use this list, instead of OnFileFound event, to get names of all found files on completion of scanning (OnScanDone event). However, the OnFileFound event can provide you with more detailed information about found file.

See also
OnFileFound and OnScanDone events;
Execute method.

8.2.2.3 ExcludeList

Applies to
dcMultiDiskScanner component.

Declaration
```
property ExcludeList: TDiskScanList; // TList
```

Description
The ExcludeList property is the list of searching rules that defines the locations and searching

© 1999-2001, UtilMind Solutions®

masks which you would like to exclude from search results.

You can set the excluding rules either at design and run-time of your application.

The ExcludeList property has the TDiskScanList type. To add the new rule to ExcludeList at run-time — call `ExcludeList.AddPath(Location, IncludeSubfolders)` method. To clear the list — call `ExcludeList.Clear` method. Please see an **example** for more details.

Notes

Setting the full path name to the excluding rules is not necessarily. For example, if you would like to exclude just some file types (extenstions), you may specify these extensions without path information (i.e: "*.bmp" will exclude ALL bitmaps from the search results):

```
ExcludeList.AddPath('*.bmp', False);
```

Plues, if the path information is not specified, the dcMultiDiskScanner will NOT search ANY folders to prepare list of previously excluded files before actual searching.

For example, you would like to find all files in root folder on disk C:

```
IncludeList.Clear;  // clear list
IncludeList.AddPath('c:\*.*', True);
```

but would like to exclude all BMPs:

```
ExcludeList.Clear;
ExcludeList.AddPath('*.bmp', True);
//                    ^-- if you specify the path
// here, i.e: "C:\*.*", the dcMultiDiskScanner will
// scan entire disk to exclude files before actual
// search, and will take a lot of time to prepare
// the pre-excluded list
```

However, if you want to exclude files only from certain directories, you must specify the path. For example, to exclude all BMP-files from, let's say, "C:\Bitmaps" directory, you must speify following exclude:

```
ExcludeList.AddPath('C:\Bitmaps\*.bmp', True);
```

See also

IncludeList property.

8.2.2.4 IncludeList

Applies to

dcMultiDiskScanner component.

Declaration

```
property IncludeList: TDiskScanList; // TList
```

Description

The IncludeList property is the list of searching rules that defines the locations and searching masks which you would like to include to search results.

You can set the including rules either at design and run-time of your application.

The IncludeList property has the TDiskScanList type. To add the new rule to IncludeList at run-time — call `IncludeList.AddPath(Location, IncludeSubfolders)` method. To clear the list — call `IncludeList.Clear` method. Please see an **example** for more details.

© 1999-2001, UtilMind Solutions®

Notes

If you would like to find files in certain locations, you MUST specify the path information. In example:

```
IncludeList.AddPath('C:\My Bitmaps\*.bmp', True);
```

Specifying the searching rules without path information, or with incomplete path information, will be considered as try to search within *current* working directory, and/or on *current* disk. However, the path will be automatically fixed, accordingly to current working disk/directory, and the OnFileFound event will return files with complete path information.

- For example, you have specified the "*.bmp" to the IncludeList:
  ```
  IncludeList.AddPath('*.bmp', True); // no path specified
  ```
  This means that you would like to found ALL bitmaps within *current working* directory + within all its subdirectories.

- In case if you have specified the disk, but not pointed to the folder where to search files, in example, "D:*.*", the dcMultiDiskScanner will search files in *current* directory on specified disk:
  ```
  IncludeList.AddPath('D:*.bmp', False); // disk without path
  ```

- You can also specify folder where to search files but not point the disk. For example, "\My Bitmaps*.bmp". The dcMultiDiskScanner will try to search files in "\My Bitmaps\" directory on current disk.
  ```
  IncludeList.AddPath('\My Bitmaps\*.bmp', False); // path without disk
  ```

If you have specified the non-existing folder, the OnFolderNotExist event occurs to inform you about impossibility to search files in this folder.

See also

ExcludeList property;
OnFileFound and OnFolderNotExist event.

8.2.2.5 Matches

Applies to

dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
type
  TDiskScannerMatches = class
  published
    property Limited: Boolean; // False by default (unlimited)
    property MaxMatches: LongInt;
  end;
```

Description

The Matches property controls the maximum number of matches that can return the DiskScanner component. If you would like to set limit the number of search results — make **Limited** property True and specify the maximum number of search matches in the **MaxMatches** property. To return unlimited number of matches — set **Limited** property to False. Remember, that you can always terminate the scanning process calling the Stop method.

See also

SearchAttributes, SearchSize, SearchTime properties.

© 1999-2001, UtilMind Solutions®

8.2.2.6 SearchAttributes

Applies to

dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
type
  TdcScanAttribute = (saNormal, saArchive, saReadOnly, saHidden,
saSystem, saDirectory, saAny);
  TdcScanAttributes = set of TdcScanAttribute;

property SearchAttributes: TdcScanAttributes;
```

Description

The SearchAttributes property specifies the possible file attributes for files or directories that can be found by dcMultiDiskScanner component.

Value	Meaning
saNormal	The file or directory has normal or no attributes set. If True, MultiDiskScanner will find files even without any attributes set.
saArchive	The file or directory is an archive file or directory. This is standard file attributes. Applications use this flag to mark files for backup or removal. Set saArchive flag to True to find files with Archive attributes set. This is normal DOS attribute that means that file is archived or copied to hard disk. Make saArchive True to find files with Archive attributes set. This file attribute usually obsolete in win32 systems, because all files kept on hard drives and even if they copied to floppy or CD, this attribute not cleaned. Anyway, this file attribute still used on *most* files.
saReadOnly	Attribute means that file or directory have read-only permission. Applications can read the file but cannot write to it or delete it. In the case of a directory, applications cannot delete it. Set saReadOnly to True to find files with Read-only attributes set.
saHidden	The file or directory is hidden. It is not included in an ordinary directory listing. Make saHidden True to find hidden files (note that hidden directories will not be found, unless the saDirectory is specified). OS hide most files which destruction lead to system failure (such files like IO.SYS, MSDOS.SYS and so forth).
saSystem	The file or directory is part of, or is used exclusively by the operating system. Make saSystem True to find system files.
saDirectory	The "file or directory" is a directory (folder). Make saDirectory True to find directories (if saHidden is True, MultiDiskScanner will also find hidden directories, i.e: "Recycled").
saAny	All (or any) attributes is set. Set saAny property to True to find files or directories with ANY file attributes.

Example

Following example demonstrates how to set or remove attributes in the SearchAttributes property. Let's suppose, we would like to add or remove from search results "hidden" files…

Delphi

```
// to add attribute
dcDiskScanner1.SearchAttributes := dcDiskScanner1.SearchAttributes + [saHidden];
// to remove attribute
dcDiskScanner1.SearchAttributes := dcDiskScanner1.SearchAttributes - [saHidden];
```

C++ Builder

© 1999-2001, UtilMind Solutions®

```
// to add attribute
dcDiskScanner1->SearchAttributes = dcDiskScanner1->SearchAttributes << saHidden;
// to remove attribute
dcDiskScanner1->SearchAttributes = dcDiskScanner1->SearchAttributes >> saHidden;
```

See also
Matches, SearchSize, SearchTime properties.

8.2.2.7 SearchSize

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
type
  TSearchSize = class
  published
    property AnySize: Boolean; // True by default
    property MinSizeKB: Integer;
    property MaxSizeKB: Integer;
  end;
```

Description
The SearchSize is the list of properties that controls the minimum and maximum file size range for files that dcMultiDiskScanner must found.

To specify the size range — make **AnySize** property False (AnySize = True means that DiskScanner component will find files with ANY size) and specify minimum and maximum sizes in kilobytes at **MinSizeKB** and **MaxSizeKB** properties.

For example, if you would like to find files with size at least 50 kilobytes and at most 100 kilobytes, make MinSizeKB = 50, MaxSizeKB = 100 and AnySize = False.

See also
Matches, SearchAttributes, SearchSize, SearchTime properties.

8.2.2.8 SearchTime

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
type
  TFindFilesByDateKind = (ffModified, ffCreated, ffAccessed);
  TSearchTime = class
  published
    property AnyTime: Boolean; // True by default
    property FindFiles: TFindFilesByDateKind;
    property SinceTime: TDateTime;
    property TillTime: TDateTime;
  end;
```

Description
The SearchTime is the list of properties that controls the time range for files that were created, modified or accessed during specific time period, specified by **SinceTime** and **TillTime** properties.

AnyTime property controls whether the MultiDiskScanner should consider the file time and find files

© 1999-2001, UtilMind Solutions®

with certain date/time. If AnyTime is True, MultiDiskScanner will found files with any time (the modification / creation / access time will does not matter).

FindFiles property controls what MultiDiskScanner should consider:

ffModified	date and time of last modifications in the found file;
ffCreated	date and time of file creation;
ffAccessed	date when file was last accessed (used).

SinceTime and **TillTime** properties controls the period of file creation or last modification.

See also
Matches, SearchAttributes, SearchSize properties;
OnFileFound event.

8.2.2.9 Suspended

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration
```
property Suspended: Boolean;
```

Description
The Suspended property indicates whether the searching thread is currently suspended. You can change the property value at run-time suspending and resuming the search process again.

See also
Execute and Stop methods.

8.2.2.10 ThreadPriority

Applies to
dcMultiDiskScanner, dcDiskScanner, dcFolderMonitor and dcFileTail components.

Declaration
```
property ThreadPriority: TThreadPriority;
```

Description
The ThreadPriority property determines the thread's scheduling priority relative to other threads in the process.

ThreadPriority indicates and specifies the priority of searching process relatively to the main application thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the TThread component, as defined in the following table. Windows schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle-Windows won't interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.

© 1999-2001, UtilMind Solutions®

tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets *highest* priority.

Warning

Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

8.2.2.11 UseIcons

Applies to

dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
property UseIcons: Boolean;
```

Description

The UseIcons property controls whether you would like to retrieve icons (32x32 and 16x16) from files found by MultiDiskScanner component.

If UseIcons = True, the icon images will be retrieved and their handles will be returned by LargeIcon and SmallIcon parameters of the OnFileFound event. If UseIcons = False, dcMultiDiskScanner will not retrieve icons and LargeIcon and SmallIcon will have NIL (NULL) values.

If you would not use icons returned by OnFileFound event — leave UseIcons property False, because process of retrieving of icons may significantly slow down searching!

However, even if UseIcons is False, MultiDiskScanner will determinate the SysImageIndex parameter.

See also

OnFileFound event.

8.2.2.12 WaitThread

Applies to

dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
property WaitThread: Boolean;
```

Description

The WaitThread property controls whether the procedure that calls the Execute method of the dcMultiDiskScanner component should be suspended and wait until the scanning process will be done.

Set the WaitThread to True, if you would like to execute dcMultiDiskScanner so that the application does not continue with next lines of code after calling the Execute method. Your application will find all files before continuing to next step.

See also

Execute method.

© 1999-2001, UtilMind Solutions®

8.2.3 Methods

8.2.3.1 Execute

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration
```
function Execute: Boolean; // returns False if Busy
```

Description
The Execute method begins the scanning process, starting the search thread. Every time when the MultiDiskScanner found files that matches to specified searching criteria, OnFileFound event occurs. When DiskScanner finished scanning of folder and going to subfolder, OnScanFolder even occurs. When DiskScanner done searching — OnScanDone occurs.

To terminate the scanning process — call Stop method.

Note
If scanning already in progress but you call Execute method again, the function will returns False. To determinate whether the searching is in progress — use Busy property.

See also
Stop method;
Suspended and WaitThread properties.

8.2.3.2 Stop

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration
```
procedure Stop;
```

Description
The Stop method terminates the scanning process, initiated by Execute method. After the termination of searching thread, OnStopped and OnScanDone events will occurs.

See also
Execute method;
OnStopped and OnScanDone events;
Suspended and WaitThread properties.

8.2.4 Events

8.2.4.1 OnExcludingBegin

Applies to
dcMultiDiskScanner component.

Declaration
```
property OnExcludingBegin: TNotifyEvent;
```

Description
The OnExcludingBegin event occurs when the MultiDiskScanner component begin prepare the list of files that must be excluded from the search results. Excluding rules can be specified at

© 1999-2001, UtilMind Solutions®

ExcludeList property.

See also
OnExcludingEnd event.

8.2.4.2 OnExcludingEnd

Applies to
dcMultiDiskScanner component.

Declaration
```
property OnExcludingEnd: TNotifyEvent;
```

Description
The OnExcludingEnd event occurs when the MultiDiskScanner component done preparing of the list of files which must be excluded from the search results and ready to start scanning by IncludeList rules.

See also
OnExcludingBegin event.

8.2.4.3 OnFileFound

Applies to
dcMultiDiskScanner and dcDiskScanner components.

Declaration
```
type
  TdcFileFoundEvent = procedure(Sender: TObject;
    FileName, FileType: String; FileSize: Extended;
    FileTime: TDateTime; FileAttributes: TdcScanAttributes;
    LargeIcon, SmallIcon: TIcon; SysImageIndex: Integer;
    TotalFiles: Integer; TotalSize: Extended) of object;

property OnFileFound: TdcFileFoundEvent;
```

Description
The OnFileFound event occurs when the dcMultiDiskScanner component found the file or directory (if `saDirectory` specified in the SearchAttributes) that matches to specified criteria (IncludeList / ExcludeList lists, SearchAttributes, SearchSize and SearchTime properties).

Use the OnFileFound event handler to take specific action once the DiskScanner found files, and, for example, add this file to the list with search results. Please check out an example of handling this event.

Parameter	Meaning
FileName	The name of found file with full path to this file. Use *ExtractFileName* and *ExtractFilePath* routines from SysUtils unit to split the file name and path.
FileType	The type of file extension, if it was registered in the shell. If extension was not registered, return value will be "EXT file", where EXT is the file extension.
FileSize	Size of found file in bytes. Use *FloatToStr* function to convert this value to string
FileTime	Time of last modification;
FileAttributes	The set of file attributes (Normal, Archive, ReadOnly, Hidden, System or Directory). See TdcScanAttributes type.
LargeIcon,	

© 1999-2001, UtilMind Solutions®

SmallIcon 32x32 and 16x16 icons, associated with this file extension in the shell. If file has .EXE extension, icon will be extracted from executable file. If file is Icon, the Icon image will returned. These parameters will be NIL (NULL), if UseIcon property is False.

SysImageIndex determines an icon index in the system ImageList. Use this parameter only if you're using dcSystemImageList component as source of images for the ListView. SysImageIndex will be retrieved even if UseIcon is False.

TotalFiles Total number of found files per current search session.

TotalSize Total size of all files that found in current search session. Use *FloatToStr* function to convert this value to string.

Notes

The *LargeIcon* and *SmallIcon* will be retrieved only if UseIcons property is True.

Example

Following example demonstrates how to handle the OnFileFound event of dcDiskScanner or dcMultiDiskScanner component. Let's say, we have the `ResultsListView: TListView` which we would like to fill with the search results and `ImageList: TImageList` which contains the 16x16 images associated with the file types.

```
procedure TForm1.dcDiskScannerFileFound(Sender: TObject; FileName,
  FileType: String; FileSize: Extended; FileTime: TDateTime;
  FileAttributes: TdcScanAttributes;
  LargeIcon, SmallIcon: TIcon; SysImageIndex: Integer;
  TotalFiles: Integer; TotalSize: Extended);
const
  KBStr = 'KB';
var
  St: String;
  ListItem: TListItem;
begin
  StatusBar.Panels[1].Text := 'Files ' + IntToStr(TotalFiles);
  StatusBar.Panels[2].Text := 'Total size: ' + FloatToStr(TotalSize) + '
bytes';

  ListItem := ListView1.Items.Add; // adding new item
{** file name}
  ListItem.Caption := FileName;
{** file type}
  ListItem.SubItems.Add(FileType);
{** file size}
  if not (saDirectory in FileAttributes) then
    if FileSize <> 0 then
      St := FloatToStrF(Int(FileSize / 1024 + 1), ffNumber, 18, 0) +
KBStr
    else
      St := '0' + KBStr
  else
    St := '';
  ListItem.SubItems.Add(St);
{** file time}
  ListItem.SubItems.Add(DateTimeToStr(FileTime));

{** icon image }
  ListItem.ImageIndex := SysImageIndex;
```

© 1999-2001, UtilMind Solutions®

```
    if saHidden in FileAttributes then
      ListItem.Cut := True; { ghosted icon for hidden files }
    St := LowerCase(ExtractFileExt(FileName));
    { adding shortcut overlay for .lnk and .url }
    if (St = '.lnk') or (St = '.url') then
      ListItem.OverlayIndex := 1;

  { //* Use code below if you don't want to use
    //* system images and set DiskScanner1.UseIcons to True
    ListItem.ImageIndex := ImageList1.Count;
    ImageList1.AddIcon(SmallIcon); }
  end;
```

See also

OnScanFolder, OnScanDone events;
Execute, Stop methods.

8.2.4.4 OnFolderNotExist

Applies to

dcMultiDiskScanner component.

Declaration

```
type
  TdcScanFolderEvent = procedure(Sender: TObject; Folder: String) of
object;

property OnFolderNotExist: TdcScanFolderEvent;
```

Description

The OnFolderNotExist event occurs if the IncludeList or ExcludeList contains instructions to scan folder which is not exist. The *Folder* parameter determines the folder that can not be scanned.

This event occursbefore the component try to scan this folder. If the folder is exist — component will search files within this folder and OnScanFolder event occurs. If not exist — the OnFolderNotExist occurs.

Example

```
procedure TForm1.dcMultiDiskScannerFolderNotExist(Sender: TObject;
   Folder: String);
begin
  Application.MessageBox(PChar('Folder "' + Folder + '" not exist.'),
   'dcMultiDiskScanner', MB_ICONINFORMATION);
end;
```

See also

OnScanFolder event.

8.2.4.5 OnScanDone

Applies to

dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
type
  TdcScanDoneEvent = procedure(Sender: TObject;
    TotalFiles: Integer; TotalSize: Extended;
```

© 1999-2001, UtilMind Solutions®

```
    ElapsedTimeInSeconds: Integer) of object;

property OnScanDone: TdcScanDoneEvent;
```

Description

The OnScanDone event occurs once the dcMultiDiskScanner component completes the scanning process by ALL criterias specified in the IncludeList property. Use this event handler to get the total number of found files (*TotalFiles* parameter), their total size (*TotalSize* parameter, use *FloatToStr* function to convert this value to string), + and time that passes after starting of the disk scanning (*ElapsedTimeInSeconds* — time that passes after calling the Execute method).

The OnScanDone also occurs after calling of the Stop method, to return the total number of found files and their total size.

See also

OnFileFound, OnScanFolder events;
Execute, Stop methods.

8.2.4.6 OnScanFolder

Applies to

dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
type
  TdcScanFolderEvent = procedure(Sender: TObject; Folder: String) of
object;

property OnScanFolder: TdcScanFolderEvent;
```

Description

The OnScanFolder event occurs when the dcMultiDiskScanner component is about to search the subfolder that located under any folder specified in IncludeList criteria. MultiDiskScanner never search same directory twice per one session.

The *Folder* parameter determines the folder which about to be scanned.

If the folder which about to be scanned is not exists — OnFolderNotExist event occurs.

See also

OnFolderNotExist, OnFileFound, OnScanDone events;
Execute, Stop methods.

8.2.4.7 OnStopped

Applies to

dcMultiDiskScanner and dcDiskScanner components.

Declaration

```
property OnStopped: TNotifyEvent;
```

Description

The OnStopped event occurs after calling the Stop method, when user terminate the searching process. However, the OnScanDone event will occurs after OnStopped anyway, to report about search results.

© 1999-2001, UtilMind Solutions®

See also
Stop method and OnScanDone event.

8.2.5 Example (IncludeList/ExcludeList)

Following examples demonstrates how to change the IncludeList and ExcludeList searching rules at run-time of your application.

Example
Let's say, we would like to find all picture files (BMP, JPG, JPEG and GIF) at location specified by *PictureFolder* variable, but would like to exclude all files that begins by "a" letter from search results. We don't want to search the files in subfolders under *PictureFolder* directory.

```
with dcMultiDiskScanner do
 begin
  with IncludeList do
   begin
    Clear; // clear the IncludeList
    AddPath(PicturesFolder + '\*.bmp', False); // False - don't include
subfolders
    AddPath(PicturesFolder + '\*.jpg', False);
    AddPath(PicturesFolder + '\*.jpeg', False);
    AddPath(PicturesFolder + '\*.gif', False);
   end;

  with ExcludeList do
   begin
    Clear; // clear the ExcludeList
    AddPath('a*.*', False);
//or...
//  AddPath(PicturesFolder + '\a*.*', False);
   end;
  Execute;
 end;
```

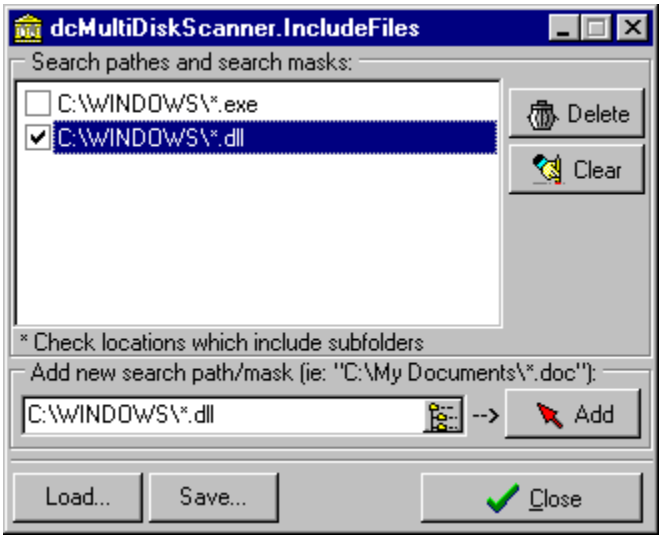
See also
dcMultiDiskScanner component, TDiskScanList and TDiskScannerPath classes.

8.2.6 Search Rules designers

The IncludeList and ExcludeList designers will help you to set the search rules at design-time.

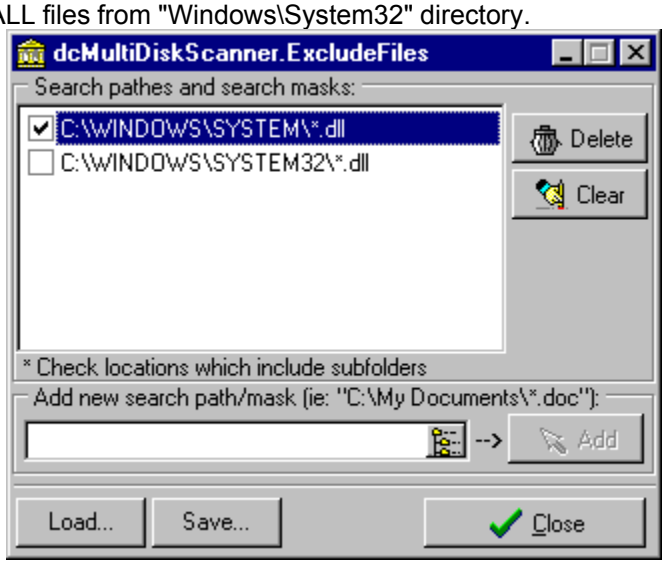
Snapshot *(IncludeList designer):*
The snapshot below demonstrates that MultiDiskScanner should include all .EXE and .DLL files in Windows directory including all subdirectories under "C:\Windows".

© 1999-2001, UtilMind Solutions®



Snapshot *(ExcludeList designer)*

The snapshot below shows that MultiDiskScanner component should exclude ALL files from "Windows\System" directory (including all files within "Windows\System" subfolders), and exclude ALL files from "Windows\System32" directory.



8.2.7 TDiskScanList class

Unit

dcDiskScanner. Used in the dcMultiDiskScanner component.

Declaration

```
type
  TDiskScanList = class(TList)
  public
    procedure AddPath(PathMask: String;
                      IncludeSubfolders: Boolean);
    procedure Clear;

    // functions returns True if successfull
    function LoadFromFile(FileName: String): Boolean;
```

© 1999-2001, UtilMind Solutions®

```
    function SaveToFile(FileName: String): Boolean;
  end;
```

Description

The TDiskScanList class is the list (successor of TList class) which contains the searching rules for IncludeList and ExcludeList properties. The TDiskScanList contains the list of TDiskScannerPath entries.

The TDiskScannerPath items contains the information about the *path and searching mask* to include or exclude from search results and whether would you like to scan *all subfolders* located under specified path.

Since TDiskScanList is successor of TList calss, it contains all properties and methods that comes with regular TList, plus three method which allows to easily operate with Include and Exclude lists:

Methods

AddPath

allows to add the new TDiskScannerPath items to the list. The parameters of the AddPath procedure — is the string that specifies a valid directory or path and filename, which can contain wildcard characters (***** and **?**) + boolean flag whether you would like to scan subdirectories under specified path. Example:

```
dcMultiDiskScanner1.IncludeList.AddPath('c:\windows\*.*', True);
```

Clear

clears all records (searching rules) from the IncludeList or ExcludeList.

SaveToFile

saves the include or exclude list from the text-file. Example:

```
dcMultiDiskScanner1.IncludeList.SaveToFile('c:\include.txt');
```

LoadFromFile

restores the include or include list from the text-file. Example:

```
dcMultiDiskScanner1.ExcludeList.LoadFromFile('c:\exclude.txt');
```

To add the new rule to IncludeList or ExcludeList at run-time — call `In(Ex)cludeList.AddPath(...)` method. To clear the list — call `In(Ex)cludeList.Clear` method. Please see an example for more details.

See also

TDiskScannerPath class and dcMultiDiskScanner component.

8.2.8 TDiskScannerPath class

Unit

dcDiskScanner. Used in dcMultiDiskScanner component.

Declaration

```
type
  TDiskScannerPath = class
  published
    property PathMask: String;
    property IncludeSubfolders: Boolean;
  end;
```

Description

The TDiskScannerPath class is an individual item of TDiskScanList class. The TDiskScanPath is the item that describes one searching rule for IncludeList and ExcludeList properties.

© 1999-2001, UtilMind Solutions®

The TDiskScannerPath items contains the information about the *path and searching mask* (**PathMask** property) to include or exclude from search results and whether would you like to *include all subfolders* (**IncludeSubfolders** property) located under specified path.

The **PathMask** is the string that specifies a valid directory or path and filename, which can contain wildcard characters (***** and **?**). **IncludeSubfolders** — boolean flag, whether to scan subdirectories under specified path.

You can change the searching rules (Include / Exclude list items) either at design and run-time (See example).

See also
TDiskScannerPath class and dcMultiDiskScanner component.

8.3 dcFolderMonitor

8.3.1 TdcFolderMonitor

Overview
The dcFolderMonitor component watch specified folder for changes and fires special notification event if any changed occurs in any file or the directory structure.

How to use?
Just specify the Folder which you wish to monitor, set up various Filters and write the OnChange event handler to perform some special operation when any changes occurs in specified Folder. To start monitoring — set Active property to True (or False to stop monitoring).

Also you can use the dcFolderMonitor together with dcFolderListView, to refresh content of the list view when any external changes occurs in the directory structure. To tie the dcFolderMonitor with the dcFolderListView just point the FolderMonitor property of the list view to any existing dcFolderMonitor component.

properties

Active	Activity flag. When True, the FolderMonitor constantly checks specified directory for changes;
Filter	The set of filters which specifies the notification level;
Folder	Specifies the directory which you would like to monitor;
ThreadPriority	The priority of the FolderMonitor thread relatively to the main application thread.

Events

OnChange	notifies you about changes in the files or directory structure.

See also
dcFolderListView component.

8.3.2 Properties

8.3.2.1 Active

Applies to
dcFolderMonitor component.

Declaration

```
property Active: Boolean;
```

© 1999-2001, UtilMind Solutions®

Description
The Active property controls whether the FolderMonitor is currently active and should constantly check content of directory specified in the Folder property for changes.

You don't need to change the activity flag when you are using the FolderMonitor together with dcFolderListView. In this case the dcFolderListView will specify and change the activity flag automatically.

See also
Folder and ThreadPriority properties.

8.3.2.2 Filter

Applies to
dcFolderMonitor component.

Declaration
```
type
  TdcFolderMonitorFilter = (fmFileName, fmDirName, fmAttributes, fmSize,
    fmWrite, fmAccess, fmSecurity);
  TdcFolderMonitorFilters = set of TdcFolderMonitorFilter;

property Filter: TdcFolderMonitorFilters;
```

Description
The Filter property specifies the level of notification about changes in the files in the directory structure of the Folder.

There are possible filters:

Value	Meaning
fmFileName	notifies you when the file name(s) changed (renamed, deleted, moved or has been created the new file), when True;
fmDirName	notifies you when the directory name(s) changed (renamed, deleted, moved or has been created the new directory), when True;
fmAttributes	notifies when the file attributes in any file is changed, when True;
fmSize	notifies when the file size changed, when True;
fmWrite	notifies when any file has been modified (WriteTime changed);
fmAccess	notifies when any file has been accessed (ReadTime changed);
fmSecurity	notifies when the security attributes has changed *(used in Win**NT** only!)*.

See also
Folder property.

8.3.2.3 Folder

Applies to
dcFolderMonitor, dcDiskScanner and dcBrowseDialog components.

Declaration
```
property Folder: String;
```

Description
The Folder property specifies the directory which should be constantly checked for changes.

© 1999-2001, UtilMind Solutions®

See also
Filter property.

8.3.2.4 ThreadPriority

Applies to
dcFolderMonitor, dcFileTail, dcDiskScanner and dcMultiDiskScanner components.

Declaration
```
property ThreadPriority: TThreadPriority;
```

Description
The ThreadPriority property determines the thread's scheduling priority relative to other threads in the process.

ThreadPriority indicates and specifies the priority of searching process relatively to the main application thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the TThread component, as defined in the following table. Windows schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle-Windows won't interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning
Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

8.3.3 Events

8.3.3.1 OnChange

Applies to
dcFolderMonitor component.

Declaration
```
property OnChange: TNotifyEvent;
```

Description
The OnChange event occurs when any changes occurs in the files within selected Folder or in the directory structure. To specify the notification level — use Filter property.

If you are using the FolderMonitor together with dcFolderListView component (FolderMonitor property of the dcFolderListView points to existing dcFolderMonitor), you don't need to handle OnChange event. The FolderListView will refresh the list items on any changes automatically.

© 1999-2001, UtilMind Solutions®

See also
Folder and Filter properties.

8.4 dcFileOperations

8.4.1 TdcFileOperations

Overview
The dcFileOperation component allows to perform the shell operations with files and folders, like copying, moving / renaming and deleting the shell object(s). Every process execute the shell dialog box which amuses users displaying smooth progress bar with according animated picture. These operations uses high level functions of the Windows Explorer so, for example, "Delete" command moves files to the "Recycled bin" instead of erasing them permanently.

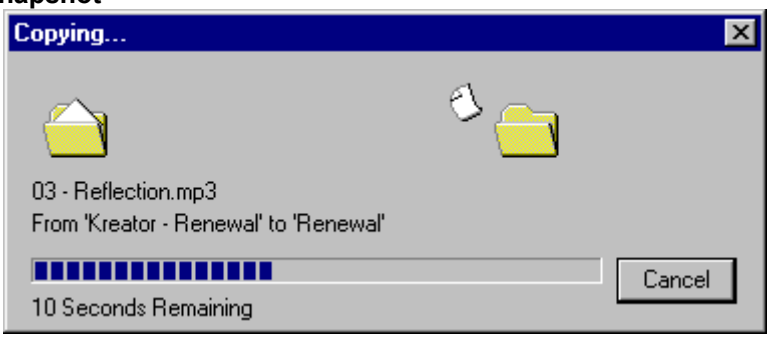
How to use ?
To perform the shell operation you have to:

1. Specify the list of files or/and folders which you wish to copy, delete, move or rename in the **FileList** property;
2. Select the destination folder in the DestFolder property (if operation is NOT Rename or Delete);
3. Then call the proper operation — Copy, Move, Rename or Delete.

- see also an example for more details.

Optionally, you may choose number of Options, and the title message for progress dialog.

Snapshot



8.4.2 Properties

8.4.2.1 DestFolder

Applies to
dcFileOperations component.

Declaration
```
property DestFolder: String;
```

Description
The DestFolder property is the name of the destination file or directory for the file operation. If DestFolder is not empty, its part will be added ahead to the destination file name.

Note
DestFolder value is ignored when you Delete files. They all will be moved to "Recycled bin".

See also

© 1999-2001, UtilMind Solutions®

FileList property.

8.4.2.2 FileList

Applies to
dcFileOperations component.

Declaration
```
property FileList: TStrings;
```

Description
The FileList property specifies one or more source file names (including full path). You should enumerate the source files or folders as items of this string list. The items can contain wildcard characters (**?** and *****). If the file contains spaces you have to take it in quotes (i.e: C:\The Bitmap.bmp must be added as "C:\The Bitmap.bmp"). For example:

```
FileList.Clear;

FileList.Add('C:\myfiles\document??.txt');
FileList.Add('C:\myfiles\picture.gif');
  // long names must be in quotes !!
FileList.Add('"C:\My Bitmaps\*.bmp"');
  // or...
for I := 0 to SomeItems.Count - 1 do
  FileList.Add('"' + SomeItems[I] + '"')
```

When you are using Copy or Move operations, these files will be copied or moved to the folder specified by the DestFolder property (If you call Delete, the files will be moved to "Recycled bin".)

If you wish to specify multiple locations where you would like to store moved or copied files, you can describe new locations near the originals, spearating them by "=" sign.

```
FileList[0]:='C:\myfiles\document??.txt=c:\mydocuments';
FileList[1]:='C:\myfiles\picture.gif=d:\mypictures';
FileList[2]:='"C:\My Bitmaps\*.bmp"="d:\bitmaps"';
```

If you would like to **Rename** some files, you must to specify new names, including the full path, separated from the originals by "=" sign. Unfortunately, due to strange behavior of the Windows Explorer, the files can not be ***renamed*** to different disk or directory, so new path must be equal to original. Example:

```
FileList.Clear;
FileList.Add('C:\myfiles\document.txt=c:\myfiles\mydoc.doc');
FileList.Add('C:\myfiles\picture.gif=c:\myfiles\greatpic.gif');
```

See also
DestFolder property;
Copy, Move, Rename and Delete methods.

8.4.2.3 Operation

Applies to
dcFileOperations component.

Declaration
```
type
  TdcFileOperationsType = (foCopy, foDelete, foMove, foRename);
```

© 1999-2001, UtilMind Solutions®

```
property Operation: TdcFileOperationsType;
```

Description

The Operation property specifies which operation to perform when you call Execute method.

The Operation can be one of the following values:

Value	Meaning
foCopy	Copies the files specified by FileList to the location specified by DestFolder.
foDelete	Deletes the files specified by FileList (DestFolder is ignored).
foMove	Moves the files specified by FileList to the location specified by DestFolder.
foRename	Renames the files specified by FileList (new names should be separated from old by "=" sign).

Use Copy, Move, Rename and Delete methods instead of Execute if you don't want to specify operation type in this property.

See also

Execute, Copy, Move, Rename and Delete methods;
FileList and DestFolder properties.

8.4.2.4 Options

Applies to

dcFileOperations, dcShellIcon, dcOpenDialog, dcSaveDialog, dcBrowseDialog components.

Declaration

```
type
  // File operation flags
  TdcFileOperationsOptions = set of (fofAllowUndo, fofFilesOnly,
fofNoConfirmation, fofNoConfirmMkDir, fofNoErrorUI, fofRenameCollision,
fofSilent, fofSimpleProgress);

property Options: TdcFileOperationsOptions;
```

Description

The Options property contains the flags which control the file operation. The Options can be a combination of following values:

Value	Meaning
fofAllowUndo	Preserves undo information, if possible. For example, if fofAllowUndo is False, the Delete operation will delete file(s) permamently instead of moving them to "Recycled bin";
fofFilesOnly	Performs the operation *only on files* if a wildcard filename (*.*) is specified;
fofNoConfirmation	Responds with "yes to all" for any dialog box that is displayed;
fofNoConfirmMkDir	Does not confirm the creation of a new directory if the operation requires one to be created;
fofNoErrorUI	don't show error message box;
fofRenameCollision	Gives the file being operated on a new name (such as "Copy #1 of...") in a move, copy, or rename operation if a file of the target name already exists;
fofSilent	Does not display a progress dialog box;
fofSimpleProgress	Displays a progress dialog box, but does not show the filenames. The value of ProgressTitle will be used instead of filenames

© 1999-2001, UtilMind Solutions®

See also
ProgressTitle and Operation properties.

8.4.2.5 ProgressTitle

Applies to
dcFileOperations component.

Declaration
```
property ProgressTitle: String;
```

Description
The ProgressTitle property specifies the title message for dialog box which shows the operation progress.

The ProgressTitle string used only if *fofSimpleProgress* flag in the Options is True.

See also
Options property.

8.4.3 Methods

8.4.3.1 Execute

Applies to
dcFileOperations component.

Declaration
```
function Execute: Boolean; // returns True if succeed
```

Description
The Execute method executes the file operation which you have specified in the Operation property. Function returns True if succeed.

See also
Copy, Move, Rename and Delete methods.

8.4.3.2 Copy

Applies to
dcFileOperations component.

Declaration
```
function Copy: Boolean; // returns True if succeed
```

Description
The Copy method copies file(s) and/or folder(s) from the FileList to the directory specified in DestFolder property. Function returns True if succeed.

See also
FileList and DestFolder properties;
Execute, Move, Rename and Delete methods.

8.4.3.3 Delete

Applies to
dcFileOperations component.

© 1999-2001, UtilMind Solutions®

Declaration

```
function Delete: Boolean; // returns True if succeed
```

Description

The Delete method moves file(s) and/or folder(s) from the FileList to "Recycled bin". Function returns True if succeed (file(s) has been successfully deleted), or False if any error occurs or operation has been canceled by user.

Notes

Value of DestFolder property will be ignored.

If *fofAllowUndo* flag in Options is False, the file(s) will be deleted permanently, without further possibility to restore them.

See also

FileList and Options properties.
Execute, Copy, Move and Rename methods.

8.4.3.4 Move

Applies to

dcFileOperations component.

Declaration

```
function Move: Boolean; // returns True if succeed
```

Description

The Move method moves file(s) and/or folder(s) described in the FileList to the directory specified in DestFolder property. Function returns True if succeed.

See also

FileList and DestFolder properties;
Execute, Copy, Rename and Delete methods.

8.4.3.5 Rename

Applies to

dcFileOperations component.

Declaration

```
function Rename: Boolean; // returns True if succeed
```

Description

The Rename method renames the file(s) and/or folder(s) described in the FileList property. The new names should be described in the FileList as well, and separated from original names by "=" sign. The items of the FileList should have following format:

```
path\OldName=samepath\NewName
```

files can not be renamed to another disk or directory, so "new" path must be equal to original. For example:

```
FileList.Clear;
// long filenames must be in quotes (")
FileList.Add('"C:\My Files\doc.txt"="C:\My Files\mydoc.doc"');
FileList.Add('"C:\folder"="C:\my folder"');
```

See also

© 1999-2001, UtilMind Solutions®

Execute, Copy, Move, and Delete methods.

8.4.4 Events

8.4.4.1 OnError

Applies to
dcFileOperations component.

Declaration

```
type
  TdcFileOperationsEvent = procedure(Sender: TObject; const S: String)
of object;

function OnError: TdcFileOperationsEvent;
```

Description
The OnError event occurs if any problem has prevented the file operation to be successfully performed. The *S* parameter returns the error message.

See also
Copy, Move, Rename, Delete and Execute methods.

8.4.5 Example

The following example demonstrates how to copy some pictures selected from TOpenPictureDialog to some location (previously specified in the DestFolder).

```
procedure TMainForm.AddBtnClick(Sender: TObject);
var
  I: Integer;
begin
  if OpenPictureDialog1.Execute then
   with dcFileOperations1 do
    try
      FileList.Clear;
      I := OpenPictureDialog1.Files.Count;
      if I <> 0 then
       for I := 0 to I - 1 do
        // long filenames must be in quotes (")
        FileList.Add('"' + OpenPictureDialog1.Files[I] + '"');
      Copy; // or any other operation
    except
    end;
end;
```

8.5 dcFileAssociation

8.5.1 TdcFileAssociation

Overview
The **dcFileAssociation** component lets you to read, install or uninstall the file types (extension associations) to the system registry. Component provides with detalied information about any file types installed to the system (ie. pas, .wav, .doc, .gif, .html). It determinates the executable file, argument string, icon file, icon index and two icons (32x32 and 16x16) associated with specified file type. To get the information about any file extension — just specify the extension at Extension

© 1999-2001, UtilMind Solutions®

property.

Also, with **dcFileAssociation** you may install new extensions or modify existing, changing the location of executable file or extension icon. For example, you can install ".yourdatafile" extension, specify the executable program for this file type and point the file icon associated with this file extension in the shell. After opening the any file with ".yourdatafile" extension (clicking the file icon), the application associated with this file type will be executed to process some data from ".yourdatafile".

How to read the information about the file type ?

Just specify any file extension (ie. pas, .wav, .doc, .gif, .html) to the EXTENSION property. All others properties will be filled with information about extension.

How to install the new file type (file association) ?

1. Switch the component to Read-Write mode (amReadWrite value of AccessMode property)
2. Assign the file extension to the Extension property.
3. Specify the type of file in the FileDescription property.
4. Point the executable file associated with this file type in the ExecutableFile property.
5. Point the file where stored the extension icon in the IconLocation property. You can point to any icon file — program (.exe), dynamic link library (.dll) or icon (.ico). Value can be same as assigned to the ExecutableFile property.
6. Specify the icon index optionally in the IconIndex property.

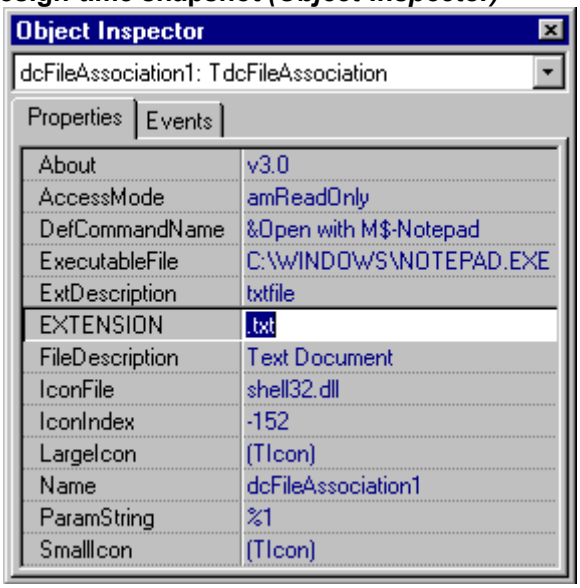
Since you don't need to call any method, just done some manipulations with a few properties, you can install the new file extension even at design-time. Please see the code sample for more details.

Alternatively you can use InstallFileType method.

How to uninstall the file type ?

Just call the UninstallFileType method and specify the extension in *Ext* parameter of UninstallFileType function.

Design-time snapshot *(Object Inspector)*



© 1999-2001, UtilMind Solutions®

8.5.2 Properties

8.5.2.1 AccessMode

Applies to
dcFileAssociation component.

Declaration
```
type
  TdcFileExtAccessMode = (amReadOnly, amReadWrite);

property AccessMode: TdcFileExtAccessMode;
```

Description
The AccessMode property controls whether other properties of this component can be modified (whether the information about file type specified in the EXTENSION property can be modified).

By default, AccessMode is `amReadOnly`. This means that you can only read the information but can not change the information for file type specified in EXTENSION property.

Set AccessMode to `amReadWrite` to install new, or modify existing file type.

Caution
⚠ Please be careful on modifying the information of existing file types!! Your changes will override previous settings in the Windows shell!

See also
EXTENSION property.

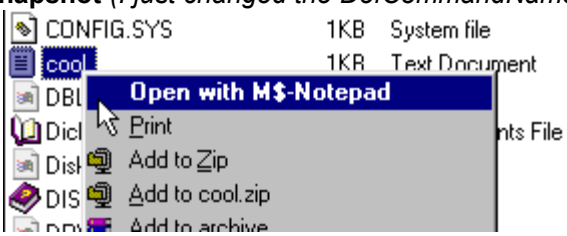
8.5.2.2 DefCommandName

Applies to
dcFileAssociation component.

Declaration
```
property DefCommandName: String;
```

Description
The DefCommandName property determines and specifies the text for default command which are used to open the file. The text for default command appears as first (default) item in the context menu for any shell objects.

Snapshot *(I just changed the DefCommandName for .TXT files)*



See also
EXTENSION property.

8.5.2.3 ExecutableFile

Applies to
dcFileAssociation component.

Declaration
```
property ExecutableFile: String;
```

Description
The ExecutableFile property determines the location of executable file, associated with the file extension specified in the EXTENSION property.

For example, when user starts any non-executable file with pre-installed file extension, the executable file, associated with that file extension will be started to process the file. For example, when user clicks the BMP icon, the shell will automatically start the "Paint" program, to allow user to modify the picture.

As a rule, the executable file associated with the file extension must be started with some command line parameters. These parameters informs the executable about the file that need to be processed. See ParamString property for more details.

If you would like to change this property and point another executable, or point an EXE for new file extension, make sure that AccessMode property is `amReadWrite` (write permission).

See also
ParamString property and example of usage.

8.5.2.4 ExtDescription

Applies to
dcFileAssociation component.

Declaration
```
property ExtDescription: String;
```

Description
The ExtDescription property determines the unique extension identifier, used internally by Windows shell.

See also
FileDescription property.

8.5.2.5 EXTENSION

Applies to
dcFileAssociation component.

Declaration
```
property EXTENSION: String;
```

Description
The EXTENSION property specifies the file extension to retrieve the information about file type.

You can play with EXTENSION property at design-time, specifying various file types (i.e: BMP, DOC, WAV, PAS, CPP, DLL, BPL, DPK and so forth).

© 1999-2001, UtilMind Solutions®

8.5.2.6 FileDescription

Applies to
dcFileAssociation component.

Declaration
```
property FileDescription: String;
```

Description
The FileDescription property determines or specifies the description of file type, specified in the EXTENSION property.

You can see description for the file types in the Windows Explorer. Open the Explorer, and select "Detail" view…

To modify the FileDescription value, make sure that AccessMode is `amReadWrite`.

See also
ExtDescription property.

8.5.2.7 IconIndex

Applies to
dcFileAssociation and dcFileAssociation components.

Declaration
```
property IconIndex: Integer;
```

Description
The IconIndex property determines or specifies the index of icon within the icon file, specified by IconLocation property.

See also
IconLocation property.

8.5.2.8 IconLocation

Applies to
dcFileAssociation and dcFileAssociation components.

Declaration
```
property IconLocation: String;
```

Description
The IconLocation property determines or specifies the location of the file which contains the icon associated with the file type specified in the EXTENSION property. The icon file can be an EXE, DLL or ICO. To determinate or specify the icon index within the icon file, use IconIndex property.

To modify the ParamString value, make sure that AccessMode is `amReadWrite`.

See also
IconIndex property.

© 1999-2001, UtilMind Solutions®

8.5.2.9 LargeIcon

Applies to
dcFileAssociation and dcShellLink components.

Declaration
```
property LargeIcon: TIcon; // read only !!
```

Description
The LargeIcon property retrieves the 32x32 icon which represents the file type, specified in the EXTENSION, within the Windows shell.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
SmallIcon property.

8.5.2.10 MIMEType

Applies to
dcFileAssociation component.

Declaration
```
property MIMEType: String;
```

Description
The MIMEType property determines or specifies the media type of the file extension. The media type (also known as Multipurpose Internet Mail Extension (MIME)) parameter used by Internet applications to determinate which *content type* it should use on sending the e-mail messages and POSTing the files via HTTP protocol.

For more information about Internet media types, please read RFC 2045, 2046, 2047, 2048, and 2077 (http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html). Check out also the Internet media type registry at ftp://ftp.iana.org/in-notes/iana/assignments/media-types.

See also
EXTENSION property.

8.5.2.11 ParamString

Applies to
dcFileAssociation component.

Declaration
```
property ParamString: String;
```

Description
The ParamString property determines and specifies the command-line parameters which should be transmitted to an executable file, when user starts any file, associated with this executable.

For example, you have installed the new file extension and would like to transmit to executable the name of data-file which starts your program. You must specify the "%1" parameter to the ParamString property, then, in the executable, catch the first command line parameter using *ParamStr(1)* routine. See FormCreate method in the sample code.

© 1999-2001, UtilMind Solutions®

To modify the ParamString value, make sure that AccessMode is `amReadWrite`.

See also
ExecutableFile property and example of usage.

8.5.2.12 SmallIcon

Applies to
dcFileAssociation and dcShellLink components.

Declaration
```
property SmallIcon: TIcon; // read only !!
```

Description
The SmallIcon property retrieves the 16x16 icon which represents the file type, specified in the EXTENSION, within the Windows shell.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
LargeIcon property.

8.5.3 Methods

8.5.3.1 InstallFileType

Applies to
dcFileAssociation component.

Declaration
```
function TdcFileAssociation.InstallFileType(Extension, ExecutableFile,
    ParamString, MIMEType, ExtDescription, FileDescription, IconFile:
    String; IconIndex: Integer): Boolean;
```

Description
The InstallFileType method is the alternative way to install the new file type to the Windows shell.

Parameters	Meaning
Extension	specifies the file extension. For example, '.test';
ExecutableFile	specifies the executable file associated with this file type (see ExecutableFile property);
ParamString	specifies the command-line parameters to transmit to the executable file on startup (see ParamString property);
MIMEType	specifies the media-type of the extension (see MIMEType property). This parameter can be empty;
ExtDescription	unique identifier for file extension, used internally within Windows shell (see ExtDescription property);
FileDescription	description of file type (will be visible in Windows explorer and "Properties" dialog) (see FileDescription property);
IconFile	location of the icon file (can be an EXE, DLL or ICO) (see IconLocation property);
IconIndex	index of the icon in the *IconFile* (see IconIndex property).

© 1999-2001, UtilMind Solutions®

To uninstall the extension from the shell — use UninstallFileType method.

Example

```
procedure TForm1.InstallButtonClick(Sender: TObject);
begin
   dcFileAssociation1.InstallFileType('.avi',
   'C:\Program Files\Windows Media Player\MPLAYER2.EXE', '%1', '',
   'avifile', 'AVI Clip', '', 0);
end;
```

See also

UninstallFileType method;
ExecutableFile, ParamString, MIMEType, ExtDescription, FileDescription, IconLocation and IconIndex properties.

8.5.3.2 UninstallFileType

Applies to

dcFileAssociation component.

Declaration

```
function UninstallFileType(Ext: String): Boolean;
```

Description

The UninstallFileType method uninstalls the file type, specified in the Ext parameter, from the system registry and clears all registry keys which describes this file extension.

See also

Example of usage;
InstallFileType method.

8.5.4 dcFileAssociation example

Following example demonstrates how to install and uninstall the file extension into the shell.
Let's say we'd like to install and uninstall the ".TEST" extension and point our executable as primary executable for this type …

```
procedure TForm1.InstallTestExtensionBtnClick(Sender: TObject);
begin
   dcFileAssociation.AccessMode := amReadWrite;
   dcFileAssociation.EXTENSION := 'test';
   dcFileAssociation.FileDescription := 'Test File';
   dcFileAssociation.ExecutableFile := Application.ExeName;
   dcFileAssociation.IconLocation := Application.ExeName;
   dcFileAssociation.IconIndex := 1;

{ ALTERNATIVELY, you can use InstallFileType method }
   dcFileAssociation.InstallFileType('.test', Application.ExeName, '%1',
'', 'testfile', 'My Test File', Application.ExeName, 1);
end;

procedure TForm1.UninstallTestExtensionBtnClick(Sender: TObject);
begin
   dcFileAssociation.UninstallFileType('test');
end;
```

© 1999-2001, UtilMind Solutions®

```
{ how to catch the command line parameter and get the name of data file
(in ParamStr(1)) }
procedure TForm1.FormCreate(Sender: TObject);
begin
  if ParamCount <> 0 then
    ShowMessage('Started by "' + ParamStr(1) + '" file !');
end;
```

8.6 dcVirtualDrives

8.6.1 TdcVirtualDrives

Overview

The dcVirtualDrives component lets an application to define, redefine and delete the virtual drives in the file system.

The Virtual Drives (also known as MS-DOS device names) is the symbolic links, in the object name space in 32-bit versions of Windows, which points to the directories of your existing drives (hard, floppy, CD and so on). The code that converts an MS-DOS path into a corresponding path in 32-bit versions of Windows uses these symbolic links to map the virtual devices and drive letters.

The dcVirtualDrives can manage the virtual drives either at run- and design-time, on either Win9x or NT-family machines.

How to use?

The dcVirtualDrives component automatically detects available virtual drives and returns the drive names in the VirtualDrives property. To get the target directory of the virtual drive — specify the first letter of this drive in the Drive property, and get the path of target directory from Target property.

To add the virtual drive to the file system, or redefine existing virtual drave to another target — use AddDrive method. To delete the virtual drive — use DeleteDrive method.

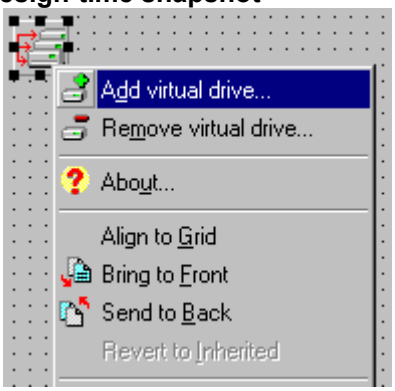
The list of UnusedDrives is a good reference to give you an idea which drives are currently free and can be defined as virtual drive.

Notes

You can NOT specify the Drive when the VirtualDrives list is empty, and can NOT specify the Target directory when the Drive is not specified.

To add the new virtual drive at *design-time*, click the right mouse button on the component image and select "Add virtual drive..." menu item.

Design-time snapshot



© 1999-2001, UtilMind Solutions®

See also
dcDiskInfo component;

The demo application (**"Virtual Drives Manager"**) at
http://www.appcontrols.com/software/virtualdrives.html

8.6.2 Properties

8.6.2.1 Drive

Applies to
dcVirtualDrives component.

Declaration
```
property Drive: Char;
```

Description
The Drive property specifies the disk letter for the virtual drive, to determinate its target directory in the Target property (see also GetTargetByDrive method).

You can specify the only drive which available in the VirtualDrives list. You can NOT specify / select the Drive, when the VirtualDrives list is empty. To add new virtual drive — use AddDrive method at *run-time*, or click the right mouse button on the component image and select the "Add virtual drive..." menu item at *desing-time*.

See also
Target and VirtualDrives properties;
GetTargetByDrive, AddDrive and DeleteDrive methods.

8.6.2.2 Target

Applies to
dcVirtualDrives component.

Declaration
```
property Target: String;
```

Description
The Target proprety determines or changes (redefines) the target directory for virtual drive specified in the Drive property.

You can NOT specify Target, if the Drive property is not specified. To add new virtual drive — at *run-time* use AddDrive method. To add new virtual drive — use AddDrive method at *run-time*, or click the right mouse button on the component image and select the "Add virtual drive..." menu item at *desing-time*.

See also
Drive and VirtualDrives properties;
AddDrive and DeleteDrive methods.

8.6.2.3 UnusedDrives

Applies to
dcVirtualDrives component.

Declaration
```
property UnusedDrives: String; // Read-only!!
```

© 1999-2001, UtilMind Solutions®

Description
The UnusedDrives property determines the number of drives which are not used in user's system and returns the first letters of these free drives. The list of unused drives returned as string.

The list of unused drives is a good reference to give you an idea which drives are currently free and can be defined as virtual drive.

See also
Drive and Target properties;
AddDrive and DeleteDrive methods.

8.6.2.4 VirtualDrives

Applies to
dcVirtualDrives component.

Declaration
```
property VirtualDrives: String; // Read-only!!
```

Description
The VirtualDrives property determines the number of available virtual drives in user's system and returns the drive letters that identifies them within the system. The list of available virtual drives returned as string.

If the VirtualDrives list is empty, you can add new virtual drive, using AddDrive method at *run-time*, or clicking the right mouse button on the component image and select the "Add virtual drive..." menu item at *desing-time*.

See also
FreeDrives, Drive and Target properties;
AddDrive and DeleteDrive methods.

8.6.3 Methods

8.6.3.1 AddDrive

Applies to
dcVirtualDrives component.

Declaration
```
function AddDrive(Drive: Char; Target: String): Boolean;
     // Returns True if successful
```

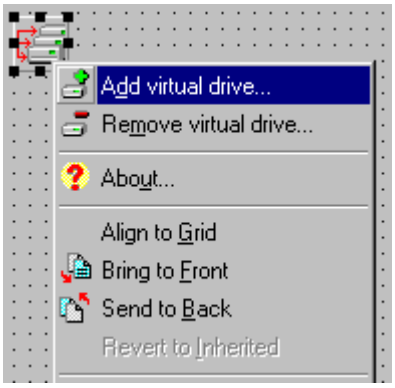
Description
The AddDrive method used to add the new virtual drive to the file system, or redefine existing virtual drive to another target. *Drive* parameter specifies the new drive letter and *Target* — target directory which should be used as separate disk. Function returns True if succeeded or False otherwise.

Example
Following line will define new virtual drive for directory with your projects:
```
dcVirtualDrives1.AddDrive('Z', 'C:\Program Files\Borland\Delphi
5\Projects');
```

To add the virtual drive at design-time — click the right mouse button on the component image

© 1999-2001, UtilMind Solutions®

and select "Add virtual drive" menu item:



See also
DeleteDrive method;
Drive, Target and VirtualDrives properties.

8.6.3.2 DeleteDrive

Applies to
dcVirtualDrives component.

Declaration
```
function DeleteDrive(Drive: Char): Boolean;
     // Returns True if successful
```

Description
The DeleteDrive method used to delete the virtual drive from file system. The *Drive* parameter specifies the first letter of existing virtual drive. Function returns True if succeeded or False otherwise.

Example
```
dcVirtualDrives1.DeleteDrive('Z');
```

See also
AddDrive method;
Drive, Target and VirtualDrives properties.

8.6.3.3 GetTargetByDrive

Applies to
dcVirtualDrives component.

Declaration
```
function GetTargetByDrive(Drive: Char): String;
```

Description
The GetTargetByDrive method returns the target directory of the virtual drive, specified in the *Drive* parameter.

Example
```
DiskZTarget := GetTargetByDrive('Z');
```

See also
AddDrive and DeleteDrive methods;

© 1999-2001, UtilMind Solutions®

Drive, Target and VirtualDrives properties.

8.7 dcDiskInfo

8.7.1 TdcDiskInfo

Overview

The dcDiskInfo component provides detailed information about system disks (Hard, Floppy, CD, RAM or Network drives). Returns the letters of available disks (ie: ACD), disk capacity and free space, file system information, serial number and shell icons (large and small), associated with specified disk.

How to use ?

Just specify the disk letter at the **Disk** property and read neccessary disk information from other properties:

AvailableDrives	shows the letters of available drives (ie. if you have floppy disk, one hard disk C: and one CD-ROM, the return value will "acd");
DiskSize, DiskFree	determines the amount of total and free disk space;
DriveType	determines the type of specified drive;
FileSystem	determines the type of file system of specified drive;
VolumeLabel	determines the disk volume label;
SerialNumber, SerialNumberStr	determines the serial number of specified disk;
TotalClusters, FreeFlusters, SectorsPerCluster, BytesPerSector	determines the low-level disk information;
LargeIcon, SmallIcon	returns the icon associated with specified disk.

See also

dcFileInfo component

8.7.2 Properties

8.7.2.1 AvailableDrives

Applies to

dcDiskInfo component.

Declaration

```
property AvailableDrives: String; // Read-only
```

Description

The AvailableDrives property determines the number of available drives in user's system and returns the drive letters that identifies them within the system.

The list of available drives returned as string. For example, if you have one floppy disk, one hard disk (with one logical disk C:) and one CD-ROM drive, the return value will be "acd" (A: - floppy, C: - hard disk and D: - CD-ROM).

See also

Disk property.

© 1999-2001, UtilMind Solutions®

8.7.2.2 BytesPerSector

Applies to
dcDiskInfo component.

Declaration
```
property BytesPerSector: DWord; // Read only!
```

Description
The BytesPerSector property determines the number of bytes per one sector of drive specified in the Disk property.

See also
SectorsPerCluster, TotalClusters, FreeClusters properties.

8.7.2.3 Disk

Applies to
dcDiskInfo component.

Declaration
```
property Disk: Char;
```

Description
The Disk property specifies the disk letter and read the information about specified drive to provide other properties with according values.

Disk letter is the character that identifies the drive in the user's system. For example, disk A is the floppy drive, C is the system disk and so forth. To read information about floppy drive "A:" — assign 'a' character to Disk property. See an example which demonstrates how to deteminate letter of CD-ROM drive and read the information about CD.

See also
AvailableDrives property.

8.7.2.4 DiskFree

Applies to
dcDiskInfo component.

Declaration
```
property DiskFree: Int64; // Read only!
```

Description
The DiskFree property determines the amount of available space on disk specified in the Disk property.

Remarks
Since the Delphi 3 and C++ Builder 3 does not supports the Int64 type, the DiskSize and DiskFree properties uses the "Extended" (float) type. Int64 are used in the Delphi4 and higher.

See also
DiskSize property.

© 1999-2001, UtilMind Solutions®

8.7.2.5 DiskSize

Applies to
dcDiskInfo component.

Declaration
```
property DiskSize: Int64; // Read only!
```

Description
The DiskSize property determines the total amount space on drive specified in the Disk property. To read the amount of available space — use DiskFree property.

Return value will **-1**, If no disk in the drive.

Remarks
Since the Delphi 3 and C++ Builder 3 does not supports the Int64 type, the DiskSize and DiskFree properties uses the "Extended" (float) type. Int64 are used in the Delphi4 and higher.

See also
DiskFree property.

8.7.2.6 DriveType

Applies to
dcDiskInfo component.

Declaration
```
type
  TdcDriveType = (dtUnknown, dtNoDrive, dtFloppy, dtFixed, dtNetwork,
dtCDROM, dtRAM);

property DriveType: TdcDriveType; // Read only!
```

Description
The DriveType property determines the type of drive specified in the Disk property.

See also
FileSystem property.

8.7.2.7 FileSystem

Applies to
dcDiskInfo component.

Declaration
```
property FileSystem: String; // Read only!
```

Description
The FileSystem property determines the type of file system on disk specified in the Disk property and returns the string identifier of detected file system.

See also
DriveType property.

© 1999-2001, UtilMind Solutions®

8.7.2.8 FreeClusters

Applies to
dcDiskInfo component.

Declaration
```
property FreeClusters: DWord; // Read only!
```

Description
The FreeClusters property determines the number of free clusters on drive specified in the Disk property.

See also
BytesPerSector, SectorsPerCluster, TotalClusters properties.

8.7.2.9 LargeIcon

Applies to
dcDiskInfo, dcFileInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration
```
property LargeIcon: TIcon; // Read only!
```

Description
The LargeIcon property returns 32x32 shell icon, associated with drive specified in Disk property.

For example, usual large icon image for floppy disk will be , for hard disk — , and for CD-Rom. You may read the icon image but can not write. To convert the icon image to bitmap — use acIcon2Bitmap component.

If you would like to get the shell icon associated with the drive without reading the information from drive — use acShellIcon component.

See also
SmallIcon property.

8.7.2.10 SectorsPerCluster

Applies to
dcDiskInfo component.

Declaration
```
property SectorsPerCluster: DWord; // Read only!
```

Description
The SectorsPerClusters property determines the number of sectors per one cluster on disk specified in the Disk property.

See also
BytesPerSector, TotalClusters, FreeClusters properties.

8.7.2.11 SerialNumber

Applies to
dcDiskInfo component.

© 1999-2001, UtilMind Solutions®

Declaration

```
property SerialNumber: LongInt; // Read only!
```

Description

The SerialNumber property detemines the serial number for disk specified in the Disk property. Serial number value is unique for every disk.

Use SerialNumberStr property to get the serial number value as string in hexadecimal format.

See also

SerialNumberStr property.

8.7.2.12 SerialNumberStr

Applies to

dcDiskInfo component.

Declaration

```
property SerialNumberStr: String; // Read only!
```

Description

The SerialNumberStr property determines the serial number of disk specified in the Disk property and converts the serial number value to string in hexadecimal format.

Use SerialNumber property to get the disk serial number as long integer value.

See also

SerialNumber property.

8.7.2.13 SmallIcon

Applies to

dcDiskInfo, dcFileInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration

```
property SmallIcon: TIcon; // Read only!
```

Description

The SmallIcon property returns 16x16 shell icon, associated with drive specified in Disk property.

For example, usual small icon image for floppy disk will be , for hard disk — , and for CD-Rom. You may read the icon image but can not write. To convert the icon image to bitmap — use acIcon2Bitmap component.

If you would like to get the shell icon associated with the drive without reading the information from drive — use acShellIcon component.

See also

LargeIcon property.

8.7.2.14 SysImageIndex

Applies to

dcDiskInfo, dcFileInfo and dcShellIcon components.

Declaration

© 1999-2001, UtilMind Solutions®

```
property SysImageIndex; // Read only!
```

Description
The SystemImageIndex property determines the index of the icon for current drive in the system image list (see dcSystemImageList component). This icon represents the disk within the Windows shell.

See also
LargeIcon and SmallIcon properties;
dcSystemImageList component.

8.7.2.15 TotalClusters

Applies to
dcDiskInfo component.

Declaration
```
property TotalClusters: DWord; // Read only!
```

Description
The TotalClusters property determines the total number of clusters on disk specified in the Disk property.

See also
BytesPerSector, SectorsPerCluster, FreeClusters properties.

8.7.2.16 VolumeLabel

Applies to
dcDiskInfo component.

Declaration
```
property VolumeLabel: String; // Read only!
```

Description
The VolumeLable property determines the volume label string for disk specified in the Disk property.

8.7.3 Example

Following example demonstrates how to determinate the letter of CD-ROM drive and read the information about current CD. Code below detects the CD-drive and shows the serial number of CD.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  I: Integer;
begin
  for I := Length(dcDiskInfo1.AvailableDrives) downto 1 do
   begin
    dcDiskInfo1.Disk := dcDiskInfo1.AvailableDrives[I];
    if dcDiskInfo1.DriveType = dtCDROM then
     begin
      if dcDiskInfo1.DiskSize <> -1 then // Disk inserted
        ShowMessage('Serial number of CD in drive ' + dcDiskInfo1.Disk +
': is ' + dcDiskInfo1.SerialNumberStr)
      else // no disk
        ShowMessage('CD-ROM drive is ' + dcDiskInfo1.Disk + ':, but you
need to insert the disk before read its serial number.');
```

© 1999-2001, UtilMind Solutions®

```
        Exit;
      end;
    end;
    ShowMessage('CD-ROM not found.');
end;
```

8.8 dcFileInfo

8.8.1 TdcFileInfo

Overview

The dcFileInfo component provides detailed information about any file — size, type, shell name, creation and last modification dates, small and large shell icons. You can get the information even from opened file and specify/change the date/time when file was created. last modified or last accessed.

How to use ?

Just specify the file name in the FileName property (you can do it even at design-time), and dcFileInfo component will read the file information to fill other properties with according values.

What information it provides ?

DisplayName	is the file name displayed by shell;
FileSize	size of file specified in FileName property;
FileType	type of file;
FileAttributes	determines/changes file attributes (Archive, System, Hidden etc);
TimeCreated	determines/changes file creation time;
TimeLastAccessed	determines/changes time when it was last accessed (used);
TimeLastModified	determines/changes time when it was last modified;
LargeIcon, SmallIcon	large and small icons (32x32 and 16x16), associated with type of specified file and identifies the file withing the Windows shell;
SysImageIndex	image index in the system image list for the icon, which represents the file withing the Windows shell.

See also

dcShellIcon and dcDiskInfo components.

8.8.2 Properties

8.8.2.1 DisplayName

Applies to

dcFileInfo, dcShellIcon components.

Declaration

```
property DisplayName: String; // Read only!
```

Description

The DisplayName property determines the string which used by the Windows shell to show the name for the shell object specieid in the FileName property.

See also

FILENAME property.

© 1999-2001, UtilMind Solutions®

8.8.2.2 FileAttributes

Applies to
dcFileInfo component.

Declaration
```
type
  TdcFileInfoAttributes = set of (fiaArchive, fiaHidden, fiaReadOnly,
fiaSystem, fiaDirectory, fiaTemporary, fiaCompressed, fiaOffline);

property FileAttributes: TdcFileInfoAttributes;
```

Description
The FileAttributes property determines and changes the attributes for the file specified in the FileName property.

Value	Meaning
fiaArchive	The file is an archive file. Applications use this value to mark files for backup or removal;
fiaHidden	The file is hidden. It is not included in an ordinary directory listing;
fiaReadOnly	The file is read-only. Applications can read the file but cannot write to it or delete it;
fiaSystem	The file is part of the operating system or is used exclusively by it;
fiaDirectory	The file is a directory;
fiaTemporary	The file is being used for temporary storage. Applications should write to the file only if absolutely necessary. Most of the file's data remains in memory without being flushed to the media because the file will soon be deleted;
fiaCompressed	The file or directory is compressed. For a file, this means that all of the data in the file is compressed. For a directory, this means that compression is the default for newly created files and subdirectories;
fiaOffline	The data of the file is not immediately available. Indicates that the file data has been physically moved to offline storage.

Remarks
1. *fiaTemporary*, *fiaCompressed* and *fiaOffline* attributes works on NT-family machines only.
2. If you'd like to change file attributes make sure that file not locked by another application. Otherwise they will remains untouched.
3. You can NOT set *fiaDirectory* flag to usual file, or clear this flag from the directory. You can not change *fiaDirectory* attribute at all.

See also
DisplayName, FileSize, FileType, TimeCreated, TimeLastModified properties.

8.8.2.3 FILENAME

Applies to
dcFileInfo, dcFileCRC, dcFileTail, dcVersionInfo, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration
```
property FILENAME: String;
```

Description

© 1999-2001, UtilMind Solutions®

The FILENAME property specifies the file name with full path, to get the information from this file. At once after pointing the filename, all other properties will be filled with according values to provide you wilth detailed information about the file.

See also
DisplayName property.

8.8.2.4 FileSize

Applies to
dcFileInfo and dcFileTail components.

Declaration
```
property FileSize: Int64; // Read only!
```

Description
The FileSize property determines the size of the file specified in the FILENAME property.

Remarks
Since the Delphi 3 and C++ Builder 3 does not supports the Int64 type, the FileSize for these Delphi/BCB versions uses the "Extended" (float) type. Int64 are used in the Delphi4 and higher.

See also
DisplayName, FileAttributes, FileType, TimeCreated, TimeLastModified properties.

8.8.2.5 FileType

Applies to
dcFileInfo and dcShellIcon components.

Declaration
```
property FileType: String; // Read only!
```

Description
The FileType property determines the type of file (specified in the FileName property), if its extension is registered within the Windows shell. If the name was not registered it returns ".EXT type" string, where the "EXT" is the file extension.

To determinate the icons associated with current file type — use LargeIcon and SmallIcon properties.

See also
DisplayName, FileAttributes, FileSize, TimeCreated, TimeLastModified properties; dcFileAssociation component.

8.8.2.6 LargeIcon

Applies to
dcFileInfo, dcDiskInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration
```
property LargeIcon: TIcon; // Read only!
```

Description
The LargeIcon property determines the 32x32 icon associated with the type of file, specified in the FileName property, and identifies the file withing the Windows shell. If the specified file is Win32

© 1999-2001, UtilMind Solutions®

executable, it returns the first icon found in this .EXE.

See also
SmallIcon and FileType properties.

8.8.2.7 SmallIcon

Applies to
dcFileInfo, dcDiskInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration
```
property SmallIcon: TIcon; // Read only!
```

Description
The SmallIcon property determines the 16x16 icon associated with the type of file, specified in the FileName property, and identifies the file withing the Windows shell. If the specified file is Win32 executable, it returns the first icon found in this .EXE.

See also
SmallIcon and FileType properties.

8.8.2.8 SysImageIndex

Applies to
dcFileInfo, dcDiskInfo and dcShellIcon components.

Declaration
```
property SysImageIndex; // Read only!
```

Description
The SystemImageIndex property determines the index of the icon for current file in the system image list (see dcSystemImageList component). This icon represents the file within the Windows shell.

See also
LargeIcon and SmallIcon properties;
dcSystemImageList component.

8.8.2.9 TimeCreated

Applies to
dcFileInfo component.

Declaration
```
property TimeCreated: TDateTime;
```

Description
The TimeCreated property determines and changes the creation date of the file specified in the FileName property.

If you change the property value, file creation time will be changed as well.

Note
File time will be changed only if write access to file is allowed (file not locked by another applications).

© 1999-2001, UtilMind Solutions®

See also
TimeLastModified and TimeLastAccessed properties.

8.8.2.10 TimeLastAccessed

Applies to
dcFileInfo component.

Declaration
```
property TimeLastAccessed: TDateTime;
```

Description
The TimeLastAccessed property determines and changes the date when the file was last used (accessed).

If you change the property value, the time when file was last accessed will be changed accordingly.

Note
File time will be changed only if write access to file is allowed (file not locked by another applications).

See also
TimeCreated and TimeLastModified properties.

8.8.2.11 TimeLastModified

Applies to
dcFileInfo component.

Declaration
```
property TimeLastModified: TDateTime;
```

Description
The TimeLastModified property determines and changes the time of last modification for file specified in the FileName property.

If you change the property value, the file modification time will be changed accordingly.

Note
File time will be changed only if write access to file is allowed (file not locked by another applications).

See also
TimeCreated and TimeLastAccessed properties.

8.9 dcFileCRC

8.9.1 TdcFileCRC

Overview
The dcFileCRC component calculates a CRC (Cyclic Redundency Check) for any files. Supports three methods of checking: CRC32 (most frequently used method, used in such programs like ZIP, RAR, ARJ etc), CRC16 (applicable to XModem protocol), CRCArc (used by SEA's "ARC" utility).

How to use ?
Just specify the FileName and checking Method and read the result from the CRC property. This

© 1999-2001, UtilMind Solutions®

works even at design-time and can calculate the CRC from the opened file.

8.9.2 Properties

8.9.2.1 CRC

Applies to
dcFileCRC component.

Declaration
```
property CRC: DWord; // Read-only !
```

Description
The CRC is the read-only property which show the result of cyclic redundancy check (CRC) for the file specified by FileName property.

To get the CRC — specify the FileName and the CRC Method.

Use the IntToHex function from the SysUtils unit to convert the result to hexadecimal value.

8.9.2.2 FILENAME

Applies to
dcFileCRC, dcFileInfo, dcFileTail, dcVersionInfo, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration
```
property FILENAME: String;
```

Description
The FILENAME property of the dcFileCRC component used to specify the file name to perform the cyclic redundancy checking (CRC) of specified file. Result of calculation returned by CRC property.

See also
Method property.

8.9.2.3 Method

Applies to
dcFileCRC component.

Declaration
```
type
  TdcFileCRCMethod = (crc32, crc16, crcArc);

property Method: TdcFileCRCMethod;
```

Description
The Method property of the dcFileCRC component used to specify the method of the cyclic redundancy checking (CRC) to calculate the checksum for file specified in the FILENAME property. Result of calculation returned by CRC property.

The dcFileCRC component supports 3 methods of the CRC calculation:

Method	Meaning
crc32	most frequently used way to perform the CRC checking. Used in such programs like ZIP, RAR, ARJ and so forth... Result of crc32 is 32-bit value;

© 1999-2001, UtilMind Solutions®

crc16	CRC calculation applicable to the XModem protocol. Result is 16-bit value;
crcArc	method used by SEA's "ARC" utility. Result is 16-bit value.

See also
FILENAME property.

8.10 dcFileTail

8.10.1 TdcFileTail

Overview
The dcFileTail is the utility component with functionality similar to the *tail* command in Unix. It monitors files for their changes and triggers the event where program can receive appended part of data. Also, like the *tail* command of Unix it can return specified number of text lines from the end of file (very useful to receive some last entries from log-files), see LastLines and LastLinesCount properties.

Besides the "tail" features, it can be used to perform simple monitoring of file changes (either constant monitoring in separate thread, or "manual" monitoring by executing special method). You probably know that when you modify the code of your unit outside of the Delphi IDE, when that unit are currently opened, the IDE notifies you about changes and prompts whether you want to use modified sources… The dcFileTail can do this monitoring for you.

How to use?
Just specify the file which you would like to monitor in the FileName property, and this file will be monitored. At once the file changed, the component will triggers OnFileChanged and/or OnFileAppended events (with OnFileAppended you can receive the data which was just appended to file.

If you would like to retrieve some lines from the end of file — read them from LastLines property. The number of returned lines can be adjusted in LastLinesCount property. To get current size of file — use FileSize property.

If you don't want to constantly check whether file is modified, just switch the MonitorType property to `mtCheckByRequest` and execute CheckChanges method in the OnActivate event of your form.

8.10.2 Properties

8.10.2.1 FileName

Applies to
dcFileCRC, dcFileInfo, dcFileTail, dcVersionInfo, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration
```
property FileName: String;
```

Description
The FileName property of the dcFileTail component used to specify the file which you would like to monitor.

When the file size is changed, the component will trigger OnFileChanged and/or OnFileAppended events.

See also

© 1999-2001, UtilMind Solutions®

FileSize property;
OnFileChanged and OnFileAppended events.

8.10.2.2 FileSize

Applies to
dcFileTail and dcFileInfo components.

Declaration
```
property FileSize: Int64; // Read only!
```

Description
The FileSize property determines the size of the file specified in the FileName property.

Remarks
Because the Delphi 3 and C++ Builder 3 does not supports the Int64 type, the FileSize for these Delphi/BCB versions uses the "Extended" (float) type. Int64 are used in the Delphi4 and higher.

See also
FileName property.

8.10.2.3 LastLines

Applies to
dcFileTail component.

Declaration
```
property LastLines: TStringList; // Read-only!!!
```

Description
The LastLines property used to retrieve some text lines from the end of file. This works both at run- and design-time.

The number of lines which should be retrieved can be specified in the LastLinesCount property.

See also
LastLinesCount property.

8.10.2.4 LastLinesCount

Applies to
dcFileTail component.

Declaration
```
property LastLinesCount: Word;
```

Description
The LastLinesCount specifies the number of lines which should be retrieved at LastLines property.

See also
LastLines property.

8.10.2.5 MonitorType

Applies to
dcFileTail component.

© 1999-2001, UtilMind Solutions®

Declaration

```
type
  TdcMonitorType = (mtCheckConstantly, mtCheckByRequest)

property MonitorType: TdcMonitorType;
```

Description

The MonitorType property specifies the kind of monitoring for the changes in file (specified in FileName property).

See also

FileName property;
CheckChanges method;
OnFileChanged event.

8.10.2.6 ThreadPriority

Applies to

dcFileTail, dcFolderMonitor, dcDiskScanner and dcMultiDiskScanner components.

Declaration

```
property ThreadPriority: TThreadPriority;
```

Description

The ThreadPriority property determines the thread's scheduling priority relative to other threads in the process.

ThreadPriority indicates and specifies the priority of searching process relatively to the main application thread. Adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the TThread component, as defined in the following table. Windows schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

Values	Meaning
tpIdle	The thread executes only when the system is idle-Windows won't interrupt other threads to execute a thread with tpIdle priority.
tpLowest	The thread's priority is two points below normal.
tpLower	The thread's priority is one point below normal.
tpNormal	The thread has normal priority.
tpHigher	The thread's priority is one point above normal.
tpHighest	The thread's priority is two points above normal.
tpTimeCritical	The thread gets highest priority.

Warning

Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events.

© 1999-2001, UtilMind Solutions®

8.10.3 Methods

8.10.3.1 CheckChanges

Applies to
dcFileTail component.

Declaration
```
procedure CheckChanges;
```

Description
The CheckChanges method allows to check the file for changes "manually" without using the separate thread, if you have set the MonitorType property to `mtCheckByRequest`.

See also
MonitorType property;
OnFileChanged event.

8.10.4 Events

8.10.4.1 OnFileAppended

Applies to
dcFileTail component.

Declaration
```
type
  TdcFileTailFileAppendedEvent = procedure(Sender: TObject;
AppendedData: TStream) of object;

property OnFileAppended: TdcFileTailFileAppendedEvent;
```

Description
The OnFileAppended event occurs when the file size has been changed AND the new file size is more than previous size (let's suppose that some data has been appended to the end of file).

If you want to retrieve appended data — use *AppendedData* parameter. This is usual TMemoryStream.

Example *(adding appended data to the TMemo)*
```
procedure TForm1.dcTail1FileAppended(Sender: TObject;
  AppendedData: TStream);

  function StreamToString(Stream: TStream): String;
  var
    Len: DWord;
  begin
    with Stream do
     begin
      Len := Size - Position;
      SetLength(Result, Len);
      Read(Result[1], Len);
     end;
  end;

begin
   Memo1.Lines.Add(StreamToString(AppendedData));
```

© 1999-2001, UtilMind Solutions®

```
end;
```

See also
OnFileChanged event.

8.10.4.2 OnFileChanged

Applies to
dcFileTail component.

Declaration

```
type
  TdcFileTailFileChangedEvent = procedure(Sender: TObject; OldFileSize,
NewFileSize: Int64) of object;

property OnFileChanged: TdcFileTailFileChangedEvent;
```

Description
The OnFileChanged event occurs when the file size is changed.

Write the OnFileChanged event handler to be informed when file updated and receive previous and new files size from *OldFileSize* and *NewFileSize* parameters.

Remarks
Because the Delphi 3 and C++ Builder 3 does not supports the Int64 type, the *OldFileSize* and *NewFileSize* parameters uses the "Extended" (float) type. Int64 are used in the Delphi4 and higher.

See also
OnFileAppended event.

8.11 dcVersionInfo

8.11.1 TdcVersionInfo

Overview
The dcVersionInfo component is used to extract version information from a file.

How to use ?
Simply set the FileName property to a file which contains version information and read any of dcVersionInfo's properties.

Example 1:
The following example demonstrates how to get the FileVersion for the SHELL32.DLL:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with dcVersionInfo1 do
   try
// if file does not contain the version info - exception raised
     FileName := 'shell32.dll';

// if succeed displaying the FileVersion (string value)
     LabelVersion1.Caption := FileVersion;

// and 16-bit values:
     LabMajor.Caption := IntToStr(FileVerMajor); // Major version
     LabMinor.Caption := IntToStr(FileVerMinor); // Minor version
```

© 1999-2001, UtilMind Solutions®

```
      LabRelease.Caption := IntToStr(FileVerRelease); // Release #
      LabBuild.Caption := IntToStr(FileVerBuild); // Build #
    except
    end;
end;
```

Example 2:
For example, we'd like to check the version number of the Internet Explorer (COMCTL32.DLL):

```
const
  IE_VER581 = (5 shl 16) or 81; // IE5.1   (Win2000 Pro)
  IE_VER580 = (5 shl 16) or 80; // IE5     (Win98 SE)
  IE_VER472 = (4 shl 16) or 72; // IE4.01
  IE_VER471 = (4 shl 16) or 71; // IE4
  IE_VER470 = (4 shl 16) or 70; // IE3
  IE_VER400 = (4 shl 16) or 00; // Win95 first release version

begin
  // ------- skipped --------
  try
    dcVersionInfo1.FileName := 'comctl32.dll';
    if dcVersionInfo1.FileMajorVersion >= IE_VER472 then
     begin
      // ----
     end;
  except
  end;
```

Notes

dcVersionInfo will raise an exception if the file cannot be found, or if the file does not contain version information. The exception will be raised when you attempt to point the FILENAME to the file without version information. Use **try**..**except** blocks to handle an exception.

8.11.2 Properties

8.11.2.1 Comments

Applies to
dcVersionInfo component.

Declaration

```
property Comments: String; // Read only!
```

Description
The Comments property contains the text from the standard "Comments" version info field.

8.11.2.2 CompanyName

Applies to
dcVersionInfo component.

Declaration

```
property CompanyName: String; // Read only!
```

Description
The CompanyName property contains the text from the standard "CompanyName" version info field.

© 1999-2001, UtilMind Solutions®

8.11.2.3 FileDescription

Applies to
dcVersionInfo component.

Declaration
```
property FileDescription: String; // Read only!
```

Description
The FileDescription property contains the text from the standard "FileDescription" version info field.

8.11.2.4 FileFlags

Applies to
dcVersionInfo component.

Declaration
```
property FileFlags: LongInt; // Read only!
```

Description
The FileFlags property is a bitmask that contains the attributes of the file.

FileFlags contains information about the file's attributes. This includes whether the file has debug info, is a pre-release version, a private build, a special build, or other file attributes. See the VS_FIXEDFILEINFO topic in the Win32 help for a list of the possible values.

See also
FileFlagsMask property.

8.11.2.5 FileFlagsMask

Applies to
dcVersionInfo component.

Declaration
```
property FileFlagsMask: LongInt; // Read only!
```

Description
The FileFlagsMask is a bitmap which describes the valid bits for the FileFlags property.

See also
FileFlags property.

8.11.2.6 FileMajorVersion

Applies to
dcVersionInfo component.

Declaration
```
property FileMajorVersion: LongInt; // Read only!
```

Description
The FileMajorVersion property represents the left-most 32 bits of the file version number.

FileMajorVersion and FileMinorVersion are used together to build a 64-bit file version number. The 64-bit value is represented in Microsoft's file version format.

© 1999-2001, UtilMind Solutions®

Use HiWord(*FileMajorVersion*) and LoWord(*FileMajorVersion*) functions to convert this 32-bit number to 16-bit version values.

Example

Following code demonstrates how to convert FileMajorVersion value to the real version number (major and minor):

```
with dcVersionInfo1 do
  Label1.Caption := IntToStr(HiWord(FileMajorVersion)) + '.' +
IntToStr(LoWord(FileMajorVersion));
```

See also

FileMinorVersion, ProductMajorVersion, ProductMinorVersion, FileVersion, FileVersionFloat properties.

8.11.2.7 FileMinorVersion

Applies to

dcVersionInfo component.

Declaration

```
property FileMinorVersion: LongInt; // Read only!
```

Description

FileMinorVersion represents the right-most 32 bits of the file version number.

FileMinorVersion and FileMajorVersion are used together to build a 64-bit file version number. The 64-bit value is represented in Microsoft's file version format.

Use HiWord(*FileMinorVersion*) and LoWord(*FileMinorVersion*) functions to convert this 32-bit number to 16-bit values which represents the release and build numbers for specified file.

Example

Following code demonstrates how to convert FileMajorVersion value to the build and release numbers:

```
with dcVersionInfo1 do
  Label1.Caption := IntToStr(HiWord(FileMinorVersion)) + '.' +
IntToStr(LoWord(FileMinorVersion));
```

See also

FileMajorVersion, ProductMajorVersion, ProductMinorVersion, FileVersion, FileVersionFloat properties.

8.11.2.8 FILENAME

Applies to

dcVersionInfo, dcFileInfo, dcFileCRC, dcFileTail, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration

```
property FILENAME: String;
```

Description

The FILENAME property is the file for which version information is being requested.

Set FileName to the name of a file for which you want version information. After setting FileName

© 1999-2001, UtilMind Solutions®

read any of the version info properties to extract the version information.

8.11.2.9 FileVerBuild

Applies to
dcVersionInfo component.

Declaration
```
property FileVerBuild: Word; // Read only!
```

Description
The FileVerBuild property determines the 16-bit build number of the file, specified in the FileName.

The same value can be retrieved from FileMinorVersion property, using LoWord(FileMinorVersion) function.

See also
FileMinorVersion, FileVerMajor, FileVerMinor, FileVerRelease and FileVerBuild properties.

8.11.2.10 FileVerMajor

Applies to
dcVersionInfo component.

Declaration
```
property FileVerMajor: Word; // Read only!
```

Description
The FileVerMajor property determines the 16-bit major version number of the file, specified in the FileName.

The same value can be retrieved from FileMajorVersion property, using HiWord(FileMajorVersion) function.

See also
FileMajorVersion, FileVerMajor, FileVerMinor, FileVerRelease and FileVerBuild properties.

8.11.2.11 FileVerMinor

Applies to
dcVersionInfo component.

Declaration
```
property FileVerMinor: Word; // Read only!
```

Description
The FileVerMinor property determines the 16-bit minor version number of the file, specified in the FileName.

The same value can be retrieved from FileMajorVersion property, using LoWord(FileMajorVersion) function.

See also
FileMajorVersion, FileVerMajor, FileVerMinor, FileVerRelease and FileVerBuild properties.

© 1999-2001, UtilMind Solutions®

8.11.2.12FileVerRelease

Applies to
dcVersionInfo component.

Declaration
```
property FileVerRelease: Word; // Read only!
```

Description
The FileVerBuild property determines the 16-bit build number of the file, specified in the FileName.

The same value can be retrieved from FileMinorVersion property, using HiWord(FileMinorVersion) function.

See also
FileMinorVersion, FileVerMajor, FileVerMinor, FileVerRelease and FileVerBuild properties.

8.11.2.13FileVersion

Applies to
dcVersionInfo component.

Declaration
```
property FileVersion: String; // Read only!
```

Description
The FileVersion property is a text string containing the file version number.

FileVersion is a text string which contains the version in "major version.minor version.release.build" format. Each value is separated with a dot. For example, version 2.05 of a product would be represented by the string "2.5.0.0". To get a floating point representation of the file version, read the FileVersionFloat property.

See also
FileVersionFloat property.

8.11.2.14FileVersionFloat

Applies to
dcVersionInfo component.

Declaration
```
property FileVersionFloat: Double; // Read only!
```

Description
The FileVersionFloat property is a floating point representation of the file version.

Use FileVersionFloat in floating point comparison operations. The following example compares the file version to a floating point number.

```
if dcVersionInfo1.FileVersionFloat < 2.0 then // ...
```

See also
FileVersion property.

© 1999-2001, UtilMind Solutions®

8.11.2.15InternalName

Applies to
dcVersionInfo component.

Declaration
```
property InternalName: String; // Read only!
```

Description
InternalName contains the text from the standard "InternalName" version info field.

8.11.2.16LanguageName

Applies to
dcVersionInfo component.

Declaration
```
property LanguageName: String; // Read only!
```

Description
The LanguageName property is the version info language in plain text.

US English resources. for example, LanguageName will contain the text "English (United States)." The high word of the TranslationValue property contains the language identifier numeric format.

See also
TranslationValue

8.11.2.17LegalCopyright

Applies to
dcVersionInfo component.

Declaration
```
property LegalCopyright: String; // Read only!
```

Description
The LegalCopyright property contains the text from the standard "LegalCopyright" version info field.

8.11.2.18LegalTrademark

Applies to
dcVersionInfo component.

Declaration
```
property LegalTrademark: String; // Read only!
```

Description
The LegalTrademark property contains the text from the standard "LegalTrademark" version info field.

8.11.2.19OriginalFilename

Applies to
dcVersionInfo component.

Declaration

© 1999-2001, UtilMind Solutions®

```
property OriginalFilename: String; // Read only!
```

Description
The OriginalFilename property contains the text from the standard "OriginalFilename" version info field.

8.11.2.20 ProductMajorVersion

Applies to
dcVersionInfo component.

Declaration
```
property ProductMajorVersion: LongInt; // Read only!
```

Description
The ProductMajorVersion property represents the left-most 32 bits of the product version number.

ProductMajorVersion and ProductMinorVersion are used together to build a 64-bit file version number. The 64-bit value is represented in Microsoft's file version format.

See also
ProductMinorVersion, FileMajorVersion, FileMinorVersion properties.

8.11.2.21 ProductMinorVersion

Applies to
dcVersionInfo component.

Declaration
```
property ProductMinorVersion: LongInt; // Read only!
```

Description
The ProductMinorVersion property represents the right-most 32 bits of the product version number.

ProductMinorVersion and ProductMajorVersion are used together to build a 64-bit file version number. The 64-bit value is represented in Microsoft's file version format.

See also
ProductMajorVersion, FileMajorVersion, FileMinorVersion properties.

8.11.2.22 ProductName

Applies to
dcVersionInfo component.

Declaration
```
property ProductName: String; // Read only!
```

Description
The ProductName property contains the text from the standard "ProductName" version info field.

8.11.2.23 ProductVersion

Applies to
dcVersionInfo component.

Declaration

© 1999-2001, UtilMind Solutions®

```
property ProductVersion: String; // Read only!
```

Description
The ProductVersion property is a string representation of the standard "ProductVersion" version info field.

See also
ProductVersionFloat property.

8.11.2.24 ProductVersionFloat

Applies to
dcVersionInfo component.

Declaration
```
property ProductVersionFloat: Double; // Read only!
```

Description
The ProductVersionFloat property is a floating point representation of the standard "ProductVersion" version info field.

See also
ProductVersion property.

8.11.2.25 TranslationValue

Applies to
dcVersionInfo component.

Declaration
```
property TranslationValue: LongInt; // Read only!
```

Description
The TranslationValue property is the value used to retrieve version info resources.

TranslationValue is a 32-bit value that contains the language identifier in the high word and the character set identifier in the low word.

See also
LanguageName property.

8.12 dcShellIcon

8.12.1 TdcShellIcon

Overview
The dcShellIcon component retrieves the icon images (32x32 and 16x16) for any specified shell object (file, folder or disk). It can determinate the real icon for shortcut files (without link overlays), or otherwise - put link mark to the shell icons . Able to show the shell object either in opened or closed state (for example, opened and closed folder icons: or), or even in selected state (see **Options** for more details).

How to use ?
Simply specify the FileName, choose Options that you're prefer, and get the 32x32 and 16x16 icons from the LargeIcon and SmallIcon properties.

© 1999-2001, UtilMind Solutions®

See also
dcFileAssociation, dcFileInfo, dcDiskInfo and dcShellLink components.

8.12.2 Properties

8.12.2.1 DisplayName

Applies to
dcFileInfo, dcShellIcon components.

Declaration
```
property DisplayName: String; // Read only!
```

Description
The DisplayName property determines the string which used by the Windows shell to show the name for the shell object specieid in the FileName property.

See also
FileName property.

8.12.2.2 FILENAME

Applies to
dcFileInfo, dcFileCRC, dcFileTail, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration
```
property FileName: String;
```

Description
The FileName property specifies the name of the shell object (file, folder or disk) to retrieve the icon images from LargeIcon and SmallIcon properties. The type of icons to retrieve can be specified in the Options.

You can specify not only the file names, but also directories (i.e: "c:\windows") or even disks (i.e: "c:\" or "a:\").

See also
Options, LargeIcon and SmallIcon properties.

8.12.2.3 FileType

Applies to
dcShellIcon and dcFileInfo components.

Declaration
```
property FileType: String; // Read only!
```

Description
The FileType property determines the type of file (specified in the FileName property), if its extension is registered within the Windows shell. If the name was not registered it returns ".EXT type" string, where the "EXT" is the file extension.

See also
FileName property.

© 1999-2001, UtilMind Solutions®

8.12.2.4 LargeIcon

Applies to
dcDiskInfo, dcFileInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration
```
property LargeIcon: TIcon; // Read only!
```

Description
The LargeIcon property determines the 32x32 icon associated with the type of shell object, specified in the FileName property, and identifies this object withing the Windows shell. The type of the icon to retrieve can be specified in the Options.

See also
SmallIcon, Options and FileName properties.

8.12.2.5 Options

Applies to
dcShellIcon component.

Declaration
```
type
  TdcShellIconOptions = set of (ioDereferenceShellLinks, ioLinkOverlay,
ioOpen, ioSelected);

property Options: TdcShellIconOptions;
```

Description
The Options property specifies the type of the icon, associated with shell object, specified in the FileName property, to retrieve this icon from the Windows shell.

The Options for the icons are following:

Values	Meaning
ioDereferenceShellLinks	If True — removes the shortcut mark from the retrieved icon, if the FileName points to the shortcut file;
ioLinkOverlay	puts the shortcut mark to the retrieved icon (i.e:), if True;
ioOpen	retrieves the icon "in open state". For example, opened folder icon is ;
ioSelected	If True, the retrieved icon will be displayed as "selected" (i.e:).

See also
FileName property.

8.12.2.6 SmallIcon

Applies to
dcDiskInfo, dcFileInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration
```
property SmallIcon: TIcon; // Read only!
```

Description
The SmallIcon property determines the 16x16 icon associated with the type of shell object, specified in the FileName property, and identifies this object withing the Windows shell. The type of the icon to retrieve can be specified in the Options.

© 1999-2001, UtilMind Solutions®

See also
LargeIcon, Options and FileName properties.

8.12.2.7 SysImageIndex

Applies to
dcShellIcon, dcFileInfo and dcDiskInfo components.

Declaration
```
property SysImageIndex; // Read only!
```

Description
The SystemImageIndex property determines the index of the icon for current file in the system image list (see dcSystemImageList component). This icon represents the file within the Windows shell.

See also
LargeIcon and SmallIcon properties;
dcSystemImageList component.

8.13 dcShellLink

8.13.1 TdcShellLink

Overview
The **dcShellLink** component lets you quickly and easily create the shell links (also known as shortcuts) to files and folders (in any location) and read the information from shorcut files (.lnk).

How to read the information from shortcut ?
Just point the **FileName** property to the .lnk file and read the information from other properties:

LinkTarget	Points to the file which should be opened on starting s shortcut (on clicking the shortcut icon);
Parameters	The command-line parameters for executable file specified by LinkTarget property. (if LinkTarget is an EXE);
WorkingDirectory	Working directory for executable file specified in LinkTarget property. (if LinkTarget is an EXE);
RunAs	Specifies how the window should be opened: minimized, maximized or in its normal state;
IconLocation, IconIndex	Points the location to icon file or another file which contains the icon resources. IconIndex property specifies the resource index, if IconLocation is an EXE or DLL;
HotKey	Specifies the key combination which executes a shortcut;
LargeIcon, SmallIcon	Read-only properties. Contains the 32x32 and 16x16 icons of shortcut link.

The FILENAME property works either at run- and design-time, so you can play with dcShellLink component, pointing the .lnk file at design-time. After specifyng the FILENAME, all other properties will be filled with information about the shortcut.

How to create the shell link (shortcut file) ?
To create a shortcut you must specify the shortcut information to according properties and call **SaveToFile(FileName)** method, where the FileName is the name of shortcut. Function returs False when it fails.

© 1999-2001, UtilMind Solutions®

For example, you would like to create the shell link, which points to your current executable file + want to specify the icon *#4* from *shell32.dll* file.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  dcShellLink1.LinkTarget := Application.ExeName;

  // next two lines will set icon #4 from shell32.dll
  dcShellLink1.IconLocation := 'shell32.dll';
  dcShellLink1.IconIndex := 4;

  // saving to Test.lnk file
  dcShellLink1.SaveToFile('c:\Test.lnk');
end;
```

How to delete the shell link (shortcut file) ?

Shell links (also known as shortcuts) is usual files, with .LNK extension. To remove a shortcuts — just delete them using standard routines, such like *DeleteFile*…

8.13.2 Properties

8.13.2.1 FILENAME

Applies to

dcFileInfo, dcFileCRC, dcFileTail, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration

```
property FILENAME: String;
```

Description

The FILENAME property specifies the name of shortcut file, to read the information from the shell link (.LNK file). At once after pointing the FILENAME to the shortcut, the dcShellLink component will read all link information to fill other properties with determinated values.

The FILENAME property is an alternative way to read the information from the shell link and does the same as LoadFromFile method. However, unlike the LoadFromFile, the FILENAME property works either at run- and design-time, so you can play with the component specifying the .LNK files and reading shortcut information even at design-time.

See also

LinkTarget and WorkingDirectory properties.

8.13.2.2 HotKey

Applies to

dcShellLink component.

Declaration

```
property HotKey: TShortcut;
```

Description

The HotKey property determines and specifies the key combination which executes a shortcut.

To convert a text string to the HotKey and vice versa, you can use TextToShortCut and ShortCutToText functions from standard "Menus" unit.

© 1999-2001, UtilMind Solutions®

8.13.2.3 IconIndex

Applies to
dcShellLink and dcFileAssociation components.

Declaration
```
property IconIndex: Integer;
```

Description
The IconIndex property determines or specifies the index of icon within the icon file, specified by IconLocation property.

See also
IconLocation property.

8.13.2.4 IconLocation

Applies to
dcShellLink and dcFileAssociation components.

Declaration
```
property IconLocation: String;
```

Description
The IconLocation property determines or specifies the location of the file which contains the icon for the shortcut. The icon file can be an EXE, DLL or ICO. To determinate or specify the icon index within the icon file, use IconIndex property.

See also
IconIndex property.

8.13.2.5 LargeIcon

Applies to
dcDiskInfo, dcFileInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration
```
property LargeIcon: TIcon; // read only !!
```

Description
The SmallIcon property retrieves the 32x32 shortcut icon associated with an .LNK file.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
SmallIcon property.

8.13.2.6 LinkTarget

Applies to
dcShellLink component.

Declaration
```
property LinkTarget: String;
```

Description

© 1999-2001, UtilMind Solutions®

The LinkTarget property determines or specifies the location of the shortcut target. The LinkTarget is the application which should be executed when user starts the the shortcut (clicks the shortcut icon).

See also
Parameters, WorkingDirectory and RunAs properties.

8.13.2.7 Parameters

Applies to
dcShellLink component.

Declaration
```
property Parameters: String;
```

Description
The Parameters property determines or specifies the command-line parameters transmitted to the executable file, specified by the LinkTarget property, on starting the shortcut.

See also
LinkTarget property.

8.13.2.8 RunAs

Applies to
dcShellLink component.

Declaration
```
type
  TdcFileAssociationRunAs = (raNormal, raMinimized, raMaximized);

property RunAs: TdcFileAssociationRunAs;
```

Description
The RunAs parameter determines or specifies how the the application, executed by shell link (shortcut) should be started.

Value	Meaning
raNormal	starts the application in normal-size window;
raMinimized	minimize the application minimized;
raMaximized	maximize the application on startup.

See also
LinkTarget property.

8.13.2.9 ShortDescription

Applies to
dcShellLink component.

Declaration
```
property ShortDescription: String;
```

Description
The ShortDescription property determines or specifies the description of the shortcut.

© 1999-2001, UtilMind Solutions®

This is not obligatory, optional shortcut parameter and, in most cases, can be empty.

8.13.2.10 SmallIcon

Applies to
dcDiskInfo, dcFileInfo, dcShellIcon, dcFileAssociation, dcShellLink components.

Declaration
```
property SmallIcon: TIcon; // read only !!
```

Description
The SmallIcon property retrieves the 16x16 shortcut icon associated with an .LNK file.

The icon will be retrieved from the file specified in IconLocation property, by index specified in IconIndex property.

See also
LargeIcon property.

8.13.2.11 WorkingDirectory

Applies to
dcShellLink component.

Declaration
```
property WorkingDirectory: String;
```

Description
The WorkingDirectory property determines and specifies the working directory for application which will be executed when user clicks a shortcut.

This is the working directory for the program specified in the LinkTarget property.

See also
LinkTarget property.

8.13.3 Methods

8.13.3.1 LoadFromFile

Applies to
dcShellLink component.

Declaration
```
function LoadFromFile(const FileName: String): Boolean;
```

Description
The LoadFromFile method reads the shortcut information from an .LNK file and fills the properties with determined values.

Alternatively you can read the information from shortcut file just pointing the FileName property to .LNK file. Unlike the LoadFromFile method, the FileName property can be specified either at run- and design-time, so you can play with the component specifying the .LNK files and reading shortcut information even at design-time.

See also
FileName property and SaveToFile method.

© 1999-2001, UtilMind Solutions®

8.13.3.2 SaveToFile

Applies to
dcShellLink component.

Declaration
```
function SaveToFile(const FileName: String): Boolean;
```

Description
The SaveToFile method creates the shell link (shortcut file), in location specified by *FileName* parameter, and writes the shortcut information specified in various properties of dcShellLink component.

See also
LoadFromFile method and example.

8.13.4 ShellLink example

Following code demonstrates how to create a shell link to your program.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  dcShellLink1.LinkTarget := Application.ExeName;

  // next two lines will set icon #4 from shell32.dll
  dcShellLink1.IconLocation := 'shell32.dll';
  dcShellLink1.IconIndex := 4;

  // saving to Test.lnk file on desktop
  dcShellLink1.SaveToFile('c:\Test.lnk');
end;
```

8.14 dcShellProperties

8.14.1 TdcShellProperties

Overview
The TdcShellProperties component used to show the "Properties" dialog box or context menu for the shell objects (file, folder or disk).

How to use ?
Just specify the name of shell object in the FileName property (this can be file, folder or even disk), and call ShowContextMenu method to display the context-menu from current mouse position, or ShowProperties to display the "Properties" dialog box.

You can also perform the same operations without specifying the FileName property — just call the ShowContextMenuByFile and ShowPropertiesByFile methods and specify the name of shell objects as method parameter.

new! The dcShellProperties component, of the DiskControls v3.0 and later, supports number advanced options which controls the behavior of the context menu. You can show or hide some groups of menu items in the context menu; allow or disallow operations assocated with menu items. See MenuOptions property and list of events for more details.

Example 1 *(following line shows the properties of "c:\autoexec.bat" file):*
```
dcShellProperties1.ShowPropertiesByFile('c:\autoexec.bat');
```

© 1999-2001, UtilMind Solutions®

Example 2 *(demostrates how to work with shell context menu)*

```
procedure TForm1.ListView1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  ListItem: TListItem;
begin
  if Button = mbRight then
   begin
    ListItem := ListView1.GetItemAt(X, Y);
    if ListItem <> nil then
     begin
      SetFocus;
      ListItem.Selected := True;

      with dcShellProperties1 do
       begin
        if ListView1.ReadOnly then
          MenuOptions := MenuOptions - [moCanRename]
        else
          MenuOptions := MenuOptions + [moCanRename];

        ShowContextMenuByFile(ListItem.Caption);
       end;
     end;
   end;
end;

procedure TForm1.dcShellProperties1Rename(Sender: TObject;
  FileName: String);
begin
  if ListView1.Selected <> nil then
    ListView1.Selected.EditCaption;
end;
```

8.14.2 Properties

8.14.2.1 FileName

Applies to

dcFileInfo, dcFileCRC, dcFileTail, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration

```
property FileName: String;
```

Description

The FileName property specifies the path to the shell object, to show the ContextMenu or the Properties dialog for this shell object. The FileName can point either to files or folders.

To show the context menu for this file — use ShowContextMenu method. To show the "Properties" dialog — use ShowProperties method.

See also

ShowContextMenu and ShowProperties methods.

© 1999-2001, UtilMind Solutions®

8.14.2.2 MenuAlignment

Applies to
dcShellProperties component.

Declaration

```
type
  TPopupAlignment = (paLeft, paRight, paCenter);

property MenuAlignment: TPopupAlignment;
```

Description
The MenuAlignment determines where the context pop-up menu appears when the user clicks the right mouse button.

Set MenuAlignment to control the placement of the popup menu. These are the possible values:

Value	Meaning
paLeft	The pop-up menu appears with its top left corner under the mouse pointer.
paCenter	The pop-up menu appears with the top center of the menu under the mouse pointer.
paRight	The pop-up menu appears with its top right corner under the mouse pointer.

The default value is paLeft.

See also
MenuOptions property.

8.14.2.3 MenuOptions

Applies to
dcShellProperties component.

Declaration

```
type
  TdcContextMenuOption = (moCanRename, moAllowDelete, moAllowCut,
moAllowCopy, moAllowPaste, moAllowCreateShortcut, moActionItems,
moExtendedItems, moSystemItems);
  TdcContextMenuOptions = set of TdcContextMenuOption;

property MenuOptions: TdcContextMenuOptions;
```

Description
The MenuOptions property controls the behavior of the context popup menu for the shell objects.

These are the possible values:

Value	Meaning
moCanRename	shows "Rename" menu item in the context menu. Set moCanRename flag to True if you would like to allow ranaming of the shell object. The shell object could not be renamed automatically. You must write OnRename event handler to perform this operation;
moAllowDelete	allows to delete shell objects (move to trash can), when True. To disallow deleting — set *moAllowDelete* flag to False;
moAllowCut	allows to cut shell objects to clipboard, when True. To disallow cutting — set *moAllowCut* flag to False;
moAllowCopy	allows to copy shell objects to clipboard, when True. To disallow copying —

© 1999-2001, UtilMind Solutions®

	set *moAllowCopy* flag to False;
moAllowPaste	allows to paste shell objects from clipboard, when True. To disallow pasting — set *moAllowPaste* flag to False;
moAllowCreateShortcut	allows to create shortucts for shell objects, when True. To disallow creation of the shortcuts — set *moAllowCreateShortcut* flag to False;

Also, you can show or hide some groups of the context menu:

Value	Meaning
moActionItems	shows or hides the "Action" menu items. These menu items identifies some default actions for the shell object, like "Open", "Print" or "Explore". If you would like to show only "action" items — set *moExtendedItems* and *moSystemItems* to False;
moExtendedItems	shows or hides the "extended" verbs of the system menu. These menu items is the shell extensions for some shell objects. If you'd like to show only "extended" and "action" items — set *moSystemItems* to False;
moSystemItems	shows or hides the "system" menu items. These menu items represents system operations for the shell object. Operations like "Cut", "Copy", "Paste", "Delete", "Rename" (if *moCanRename* is True), "Properties" and so forth... If you would like to show only "system" items — set *moActionItems* and *moExtendedItems* to False;

Example

```
with dcShellProperties1 do
  MenuOptions := MenuOptions - [moExtendedItems];
```

See also

ShowContextMenu, ShowContextMenuByFile methods;
OnDelete, OnRename events.

8.14.3 Methods

8.14.3.1 ShowContextMenu

Applies to

dcShellProperties component.

Declaration

```
function ShowContextMenu: Boolean;
```

Description

The ShowContextMenu method executes the shell context menu, for the shell object specified in the FileName property. The popup menu will appears at current mouse position. ShowContextMenu method returns True if succeeed, or False otherwise.

Use MenuOptions property to specify some advanced options for the context menu and MenuAlignment to specify the menu position.

You can test this feature at design-time, specifying the FileName property in the Object Inspector and clicking the right mouse button on the component image. In the popup menu select the "Test ShowContextMenu" item.

Example

```
procedure TForm1.ListView1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
```

© 1999-2001, UtilMind Solutions®

```
  ListItem: TListItem;
begin
  if Button = mbRight then
   begin
    ListItem := ListView1.GetItemAt(X, Y);
    if ListItem <> nil then
     begin
      SetFocus;
      ListItem.Selected := True;

      with dcShellProperties1 do
       begin
        FileName := ListItem.Caption // ?

        if ListView1.ReadOnly then
          MenuOptions := MenuOptions - [moCanRename]
        else
          MenuOptions := MenuOptions + [moCanRename];

        ShowContextMenu;
       end;
     end;
   end;
end;

procedure TForm1.dcShellProperties1Rename(Sender: TObject;
  FileName: String);
begin
  if ListView1.Selected <> nil then
    ListView1.Selected.EditCaption;
end;
```

See also

MenuOptions and MenuAlignment properties;
ShowContextMenuByFile method.

8.14.3.2 ShowContextMenuByFile

Applies to

dcShellProperties component.

Declaration

```
function ShowContextMenuByFile(FileName: String): Boolean;
```

Description

The ShowContextMenuByFile method executes the shell context menu for the file specified in the *FileName* parameter (current value of the FileName property does not matter). The menu will appears at current mouse position.

Use MenuOptions property to specify some advanced options for the context menu and MenuAlignment to specify the menu position.

You can test this feature at design-time, specifying the FileName property in the Object Inspector and clicking the right mouse button on the component image. In the popup menu select the "Test ShowContextMenu" item.

© 1999-2001, UtilMind Solutions®

Example

```
procedure TForm1.ListView1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  ListItem: TListItem;
begin
  if Button = mbRight then
   begin
    ListItem := ListView1.GetItemAt(X, Y);
    if ListItem <> nil then
     begin
      SetFocus;
      ListItem.Selected := True;

      with dcShellProperties1 do
       begin
        if ListView1.ReadOnly then
          MenuOptions := MenuOptions - [moCanRename]
        else
          MenuOptions := MenuOptions + [moCanRename];

        ShowContextMenuByFile(ListItem.Caption);
       end;
     end;
   end;
end;

procedure TForm1.dcShellProperties1Rename(Sender: TObject;
  FileName: String);
begin
  if ListView1.Selected <> nil then
    ListView1.Selected.EditCaption;
end;
```

See also

MenuOptions and MenuAlignment properties;
ShowContextMenu method.

8.14.3.3 ShowProperties

Applies to

dcShellProperties component.

Declaration

```
function ShowProperties: Boolean;
```

Description

The ShowProperties method executes the "Properties" dialog box, for the shell object specified in the FileName property.

ShowProperties method returns True if succeeed, or False otherwise.

You can test this feature at design-time, specifying the FileName property in the Object Inspector and clicking the right mouse button on the component image. In the popup menu select the "Test ShowProperties" item.

© 1999-2001, UtilMind Solutions®

See also
ShowPropertiesByFile method.

8.14.3.4 ShowPropertiesByFile

Applies to
dcShellProperties component.

Declaration
```
function ShowPropertiesByFile(FileName: String): Boolean;
```

Description
The ShowPropertiesByFile method shows the "Properties" dialog box for the shell object specified in the FileName parameter (value of the FileName property does not matter).

ShowPropertiesByFile method returns True if succeeed, or False otherwise.

Example *(this line shows the properties of "c:\autoexec.bat" file):*
```
dcShellProperties1.ShowPropertiesByFile('c:\autoexec.bat');
```

See also
ShowProperties method.

8.14.4 Events

8.14.4.1 OnAfterMenuItemClick

Applies to
dcShellProperties component.

Declaration
```
type
  TdcAfterMenuItemClickEvent = procedure(Sender: TObject; FileName:
    String; MenuItemID: Integer) of object;

property OnAfterMenuItemClick: TdcAfterMenuItemClickEvent;
```

Description
The OnAfterMenuItemClick event occurs after selecting the menu item from the context menu for shell object.

The *FileName* parameter determines the name of the shell object, and *MenuItemID* determines the unique ID of selected menu item.

See also
OnBeforeMenuItemClick event.

8.14.4.2 OnBeforeMenuItemClick

Applies to
dcShellProperties component.

Declaration
```
type
  TdcBeforeMenuItemClickEvent = procedure(Sender: TObject; FileName:
    String; MenuItemID: Integer; var Discard: Boolean) of object;
```

© 1999-2001, UtilMind Solutions®

```
property OnBeforeMenuItemClick: TdcBeforeMenuItemClickEvent;
```

Description

The OnBeforeMenuItemClick event occurs after selecting the menu item from the context menu for shell object.

The *FileName* parameter determines the name of the shell object, and *MenuItemID* determines the unique ID of selected menu item. If you would like to discard this operation — set *Discard* parameter to True in the event handler.

See also

OnAfterMenuItemClick event.

8.14.4.3 OnDelete

Applies to

dcShellProperties component.

Declaration

```
type
  TdcContextMenuEvent = procedure(Sender: TObject; FileName: String) of
object;

property OnDelete: TdcContextMenuEvent;
```

Description

The OnDelete property occurs when user clicks "Delete" item in the context menu for the shell object. The *FileName* parameter determines the file name which should be renamed.

To disallow deleting of the shell objects from context menu — set *moAllowDelete* flag of the MenuOptions property to False.

See also

OnRename event.

8.14.4.4 OnRename

Applies to

dcShellProperties component.

Declaration

```
type
  TdcContextMenuEvent = procedure(Sender: TObject; FileName: String) of
object;

property OnRename: TdcContextMenuEvent;
```

Description

The OnRename event occurs when user selects the "Rename" item in the context menu. The *FileName* parameter determines the file name which should be renamed.

"Rename" menu item will be available in context menu only if *moCanRename* flag of MenuOptions property is True.

Shell objects could not be renamed automatically. You *must* write the event handler to perform this operation.

© 1999-2001, UtilMind Solutions®

Example

```
procedure TForm1.dcShellPropertiesRename(Sender: TObject; FileName:
  String);
begin
  if ListView1.Selected <> nil then
    ListView1.Selected.EditCaption;
end;
```

See also

MenuOptions property; OnDelete event;
ShowContextMenu and ShowContextMenuByFile methods.

8.15 dcFolderListView

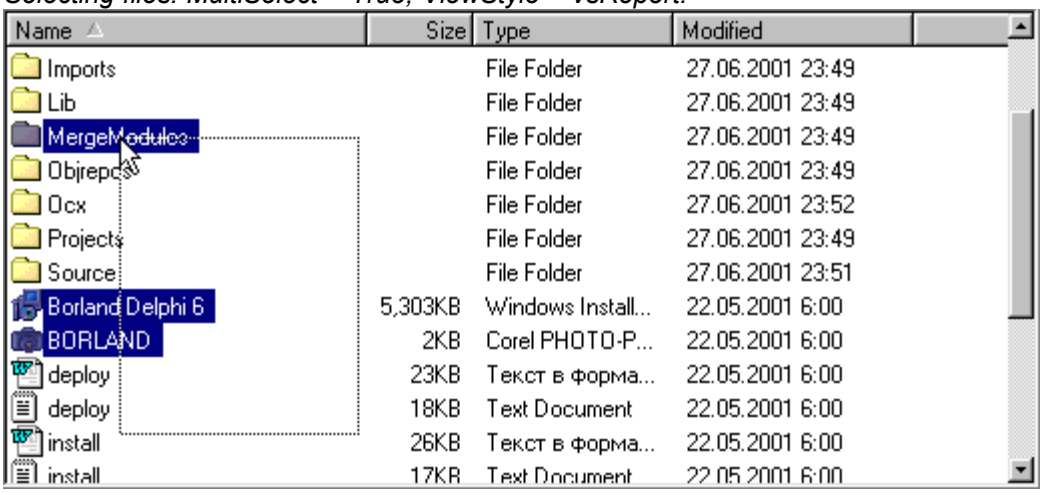
8.15.1 TdcFolderListView

Overview

The dcFolderListView component is the ListView control for displaying contents of any directory. The directory contents can be filtered by file attributes or multiple file masks. User can browse folders, execute files, perform major file operations like in standard shell list view control used in Windows Explorer. dcFolderListView supports big number of additional features like automatical sorting of the list items with arrow-style sort mark on the header section, individual context menus for every shell object, possibility to hide or show some list columns, automatic drag'n'drop support via OLE (user can drop files from Windows Explorer and vice versa) + a lots of properties and methods for complete customization.

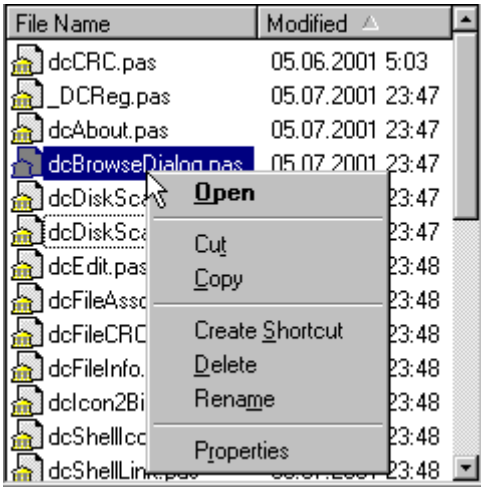
Snapshots

Selecting files. MultiSelect = True; ViewStyle = vsReport.



Invoking the context menu for shell object. Some unwanted columns are hidden. The directory contents are filtered by .pas extension.

© 1999-2001, UtilMind Solutions®



See also
dcFolderMonitor, dcDiskScanner, dcSystemImageList, dcShellProperties, dcFileOperations and dcFileAssociation components.

8.15.2 Properties

8.15.2.1 Columns

Applies to
dcFolderListView component.

Declaration

```
type
  TdcListColumns = class
  published
    property Name: TdcListColumn;
    property Size: TdcListColumn;
    property FileType: TdcListColumn;
    property Modified: TdcListColumn;
    property Attributes: TdcListColumn;
  end;

property Columns: TdcListColumns;
```

Description
The Columns structure lets you easily specify standard columns for file list view control, when ViewStyle = vsReport.

8.15.2.1.1 Attributes

Applies to
dcFolderListView component as subproperty of the Columns structure.

Declaration

```
type
  TdcListColumn = class
  published
    property Alignment: TAlignment;
    property Caption: String;   // 'Attributes' by default
    property Width: TWidth;     // 50 by default
```

© 1999-2001, UtilMind Solutions®

```
    property Visible: Boolean; // False by default
  end;

property Attributes: TdcListColumn;
```

Description

The TdcListColumn structure intended to specify parameters for certain column of the dcFolderListView. You can set the Alignment, Caption (use it to localize the header) and Width of the header column, and whether you would like to show or hide this column (Visible property).

The Attributes column is invisible by default. To make it visible — set Visible property to True. To translate the caption to another language — specify your own text in Caption.

See also

Name, Size, FileType, Modified and Attributes structures (columns).

8.15.2.1.1.1 Alignment

Applies to

dcFolderListView component as subproperty of Columns.Attributes structure.

Declaration

```
property Alignment: TAlignment;
```

Description

See also

Caption, Visible and Width properties.

8.15.2.1.1.2 Caption

Applies to

dcFolderListView component as subproperty of Columns.Attributes structure.

Declaration

```
property Caption: String;
```

Description

See also

Alignment, Visible and Width properties.

8.15.2.1.1.3 Visible

Applies to

dcFolderListView component as subproperty of Columns.Attributes structure.

Declaration

```
property Visible: Boolean;
```

Description

See also

Alignment, Caption and Width properties.

© 1999-2001, UtilMind Solutions®

8.15.2.1.1.4 Width

Applies to
dcFolderListView component as subproperty of Columns.Attributes structure.

Declaration
```
property Width: TWidth; // -1..MaxInt
```

Description

See also
Alignment, Caption and Visible properties.

8.15.2.1.2 FileType

Applies to
dcFolderListView component as subproperty of the Columns structure.

Declaration
```
type
  TdcListColumn = class
  published
    property Alignment: TAlignment;
    property Caption: String;  // 'Type' by default
    property Width: TWidth;    // 100 by default
    property Visible: Boolean;
  end;

property FileType: TdcListColumn;
```

Description
The TdcListColumn structure intended to specify parameters for certain column of the dcFolderListView. You can set the Alignment, Caption (use it to localize the header) and Width of the header column, and whether you would like to show or hide this column (Visible property).

The FileType column is visible by default. To hide it — set Visible property to False. To translate the caption to another language — specify your own text in Caption.

See also
Name, Size, FileType, Modified and Attributes structures (columns).

8.15.2.1.3 Modified

Applies to
dcFolderListView component as subproperty of the Columns structure.

Declaration
```
type
  TdcListColumn = class
  published
    property Alignment: TAlignment;
    property Caption: String;  // 'Modified' by default
    property Width: TWidth;    // 120 by default
    property Visible: Boolean;
  end;

property Modified: TdcListColumn;
```

© 1999-2001, UtilMind Solutions®

Description

The TdcListColumn structure intended to specify parameters for certain column of the dcFolderListView. You can set the Alignment, Caption (use it to localize the header) and Width of the header column, and whether you would like to show or hide this column (Visible property).

The Modified column is visible by default. To hide it — set Visible property to False. To translate the caption to another language — specify your own text in Caption.

See also

Name, Size, FileType, Modified and Attributes structures (columns).

8.15.2.1.4 Name

Applies to

dcFolderListView component as subproperty of the Columns structure.

Declaration

```
type
  TdcListColumn = class
  published
    property Alignment: TAlignment;
    property Caption: String;  // 'Name' by default
    property Width: TWidth;    // 180 by default
    property Visible: Boolean; // Always True and read-only!!
  end;

property Name: TdcListColumn;
```

Description

The TdcListColumn structure intended to specify parameters for certain column of the dcFolderListView. You can set the Alignment, Caption (use it to localize the header) and Width of the header column, and whether you would like to show or hide this column (Visible property).

The Name column is visible by default. To hide it — set Visible property to False. To translate the caption to another language — specify your own text in Caption.

See also

Name, Size, FileType, Modified and Attributes structures (columns).

8.15.2.1.5 Size

Applies to

dcFolderListView component as subproperty of the Columns structure.

Declaration

```
type
  TdcListColumn = class
  published
    property Alignment: TAlignment;
    property Caption: String;  // 'Size' by default
    property Width: TWidth;    // 60 by default
    property Visible: Boolean;
  end;

property Size: TdcListColumn;
```

© 1999-2001, UtilMind Solutions®

Description
The TdcListColumn structure intended to specify parameters for certain column of the dcFolderListView. You can set the Alignment, Caption (use it to localize the header) and Width of the header column, and whether you would like to show or hide this column (Visible property).

The Size column is visible by default. To hide it — set Visible property to False. To translate the caption to another language — specify your own text in Caption.

See also
Name, Size, FileType, Modified and Attributes structures (columns).

8.15.2.2 Cursors

Applies to
dcFolderListView component (dcComCtrls unit)

Declaration
```
type
  TdcListViewCursors = class
  published
    property Border: TCursor;
    property Divider: TCursor;
    property Header: TCursor;
    property ScrollHorz: TCursor;
    property ScrollVert: TCursor;
  end;

property Cursors: TdcListViewCursors;
```

Description
Use Cursors structure to specify custom cursor images for some non-client parts of dcFolderListView control. You can specify cursor for border of control, header, columns divider, horizontal and vertical scrollbars.

8.15.2.2.1 Border

Applies to
dcFolderListView component as subproperty of Cursors structure.

Declaration
```
property Border: TCursor;
```

Description
The Border property specifies the image for mouse pointer, when it passes over border of dcFolderListView control.

See also
Header, Divider, ScollHorz and ScrollVert properties.

8.15.2.2.2 Divider

Applies to
dcFolderListView component as subproperty of Cursors structure.

Declaration
```
property Divider: TCursor;
```

© 1999-2001, UtilMind Solutions®

Description
The Divider property specifies the image for mouse pointer, when it passes over the area which separates the header columns.

See also
Border, Header, ScollHorz and ScrollVert properties.

8.15.2.2.3 Header

Applies to
dcFolderListView component as subproperty of Cursors structure.

Declaration
```
property Header: TCursor;
```

Description
The Header property specifies the image for mouse pointer, when it passes over the header of dcFolderListView control (when ViewStyle = vsReport).

See also
Border, Divider, ScollHorz and ScrollVert properties.

8.15.2.2.4 ScrollHorz

Applies to
dcFolderListView component as subproperty of Cursors structure.

Declaration
```
property ScrollHorz: TCursor;
```

Description
The ScrollHorz property specifies the image for mouse pointer, when it passes over horizontal scrollbar of dcFolderListView control.

See also
ScrollVert, Border, Header and Divider properties.

8.15.2.2.5 ScrollVert

Applies to
dcFolderListView component as subproperty of Cursors structure.

Declaration
```
property ScrollVert: TCursor;
```

Description
The ScrollVert property specifies the image for mouse pointer, when it passes over vertical scrollbar of dcFolderListView control.

See also
ScollHorz, Border, Header and Divider properties.

8.15.2.3 FileAttributes

Applies to
dcFolderListView component.

© 1999-2001, UtilMind Solutions®

Declaration

```
type
  TdcScanAttribute = (saNormal, saArchive, saReadOnly, saHidden,
saSystem, saDirectory, saAny);
  TdcScanAttributes = set of TdcScanAttribute;

property FileAttributes: TdcScanAttributes;
```

Description

The SearchAttributes property specifies the possible file attributes for files or directories that should be displayed by dcFolderListView.

Possible values for file attributes is:

Value	Meaning
saNormal	The file or directory has normal or no attributes set. If True, dcFolderListView will display files even without any attributes set.
saArchive	The file or directory is an archive file or directory. This is standard file attributes. Applications use this flag to mark files for backup or removal. Set saArchive flag to True to find files with Archive attributes set. This is normal DOS attribute that means that file is archived or copied to hard disk. Make saArchive True to find files with Archive attributes set. This file attribute usually obsolete in win32 systems, because all files kept on hard drives and even if they copied to floppy or CD, this attribute not cleaned. Anyway, this file attribute still used on *most* files.
saReadOnly	Attribute means that file or directory have read-only permission. Applications can read the file but cannot write to it or delete it. In the case of a directory, applications cannot delete it. Set saReadOnly to True to display files with Read-only attributes set.
saHidden	The file or directory is hidden. It is not included in an ordinary directory listing. Make saHidden True to show hidden files (note that hidden directories will not be found, unless the saDirectory is specified). OS hide most files which destruction lead to system failure (such files like IO.SYS, MSDOS.SYS and so forth).
saSystem	The file or directory is part of, or is used exclusively by the operating system. Make saSystem True to find system files.
saDirectory	The "file or directory" is a directory (folder). Make saDirectory True to show folders (if saHidden is True, dcFolderListView will also show hidden folders, i.e: "Recycled").
saAny	All (or any) attributes is set. Set saAny property to True to find files or directories with ANY file attributes.

8.15.2.4 FileMask

Applies to

dcFolderListView component.

Declaration

```
property FileMask: String; // *.* by default
```

Description

The FileMask property is the wildcarded mask, filter which specifies which files you would like to show in the dcFolderListView control. The FileMask can contain wildcard characters (***** and **?**). For example, if you would to show only .PAS files — set FileMask to "***.pas**". To show only .PAS files which begins from "**a**" character — set FileMask to "**a*.pas**"

© 1999-2001, UtilMind Solutions®

As well as the SearchMask property of the dcDiskScanner component, the FileMask can contain multiple mask, separated by semicolon (;). For example, if you would like to show only graphic files — set FileMask to "***.bmp;*.gif;*.jpg**".

Note

Even after specifying the filter mask, the dcFolderListView will show ALL folders as far as ShowAllFolders property is True, disregarding to FileMask.

For example, if you would like to show only files and folders that begins from "**a**" character, set FileMask to "**a*.***" and ShowAllFolders to False.

See also

ShowAllFolders property.

8.15.2.5 Folder

Applies to

dcFolderListView component.

Declaration

```
property Folder: String; // 'C:\' by default
```

Description

The Folder property identifies the current folder to view in the dcFolderListView control. Set this property to change the current folder, or read it to determine the current folder.

At once after changing of the Folder property, the content of the list view will be automatically refreshed.

To filter displayed context of the folder — use FileMask and FileAttributes property.

To be notified when the Folder value is changed (when user passes to another subfolder, in example, clicking the folder icons) — use OnFolderChanged event. The OnFolderChanged event will NOT occurs when you changing the Folder value programatically.

See also

FileMask and FileAttributes properties;
OnFolderChanged event.

8.15.2.6 FolderMonitor

Applies to

dcFolderListView component.

Declaration

```
property FolderMonitor: TdcFolderMonitor;
```

Description

The FolderMonitor property is the pointer to any dcFolderMonitor component on the form. Select the FolderMonitor to be notified when any changes occurs in the files of specified Folder, or any changes occurs in its directory structure

After specifying the FolderMonitor, it will be tied with the FolderListView and can not be used stand-alone. The dcFolderMonitor will always watch the same monitor as which displayed in the list view.

© 1999-2001, UtilMind Solutions®

See also
Folder property;
dcFolderMonitor component.

8.15.2.7 HeaderMenu

Applies to
dcFolderListView component.

Declaration
```
type
  TdcListViewHeaderMenu = class
  published
    property AscendingItem: TMenuItem;
    property DescendingItem: TMenuItem;
    property PopupMenu: TPopupMenu;
  end;

property HeaderMenu: TdcListViewHeaderMenu;
```

Description
The HeaderMenu structure specifies custom popup menu for the header sections of the dcFolderListView control. You can point custom PopupMenu + "Ascending" and "Descending" items in this menu, which will automatically change sorting direction when user selects these items.

Usually you don't need to specify custom menu, because the dcFolderListView already has built-in menu with two default items "Sort Ascending" and "Sort Descending". Use the custom popup menu only if you would like to add additional features to the header menu or wish to translate the text for menu items to another language.

If you don't want to show the popup menu for header columns at all — set ShowHeaderMenu property to False.

See also
ShowHeaderMenu and SortDirection properties.

8.15.2.7.1 AscendingItem

Applies to
dcFolderListView component as subproperty of HeaderMenu structure.

Declaration
```
property AscendingItem: TMenuItem;
```

Description
The AscendingItem property points to the menu item, in the custom PopupMenu for header sections, which will automatically change the sorting direction to sdAscending.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
PopupMenu, DescendingItem and SortDirection properties.

© 1999-2001, UtilMind Solutions®

8.15.2.7.2 DescendingItem

Applies to
dcFolderListView component as subproperty of HeaderMenu structure.

Declaration
```
property DescendingItem: TMenuItem;
```

Description
The DescendingItem property points to the menu item, in the custom PopupMenu for header sections, which will automatically change the sorting direction to sdDescending.

You don't need to handle the OnClick event for this menu item, all operations will be performed automatically.

See also
AscendingItem and PopupMenu properties.

8.15.2.7.3 PopupMenu

Applies to
dcFolderListView component as subproperty of HeaderMenu structure.

Declaration
```
property PopupMenu: TPopupMenu;
```

Description
The PopupMenu property points to the custom popup menu, which will be used when user clicks the right mouse button. To point the menu items which will sort the columns when user clicks them — use AscendingItem and DescendingItem properties.

By default, the dcFolderListView already has the built-in popup menu with two menu items, "Sort Ascending" and "Sort Descending". Use the custom popup menu only if you would like to add additional features to the header menu or wish to translate the text for menu items to another language.

See also
AscendingItem and DescendingItem properties.

8.15.2.8 HeaderStyle

Applies to
dcFolderListView component.

Declaration
```
type
  THeaderStyle = (hsButtons, hsFlat);

property HeaderStyle: THeaderStyle;
```

Description
The HeaderStyle property specifies the appearance of the header sections above the ListView control. (Header visible only if the ViewStyle is vsReport).

There are the possible values:

Value	Meaning

© 1999-2001, UtilMind Solutions®

hsButtons	The column sections are buttons;
hsFlat	The column sections are flat buttons.

Note
⚠ User can not click the header section (change SortDirection) when HeaderStyle = hsFlat.

8.15.2.9 KBStr

Applies to
dcFolderListView component.

Declaration
```
property KBStr: String; // 'KB' by default
```

Description
The KBStr property specifies the string which will be added after the number, which represents the size of file in kilobytes, in the Size column (when the ViewStyle = vsReport).

Default KBStr value is 'KB'. However, you can change this string to any other or translate to another language. If you don't want to show this suffix at all, and show only numbers — clear the KBStr value.

Snapshoot *(Russian translation)*

Размер	Тип
1,067 Кб	Помощь
532 Кб	ISU файл
218 Кб	Системный
197 Кб	WinZip файл

See also
Columns.Size structure (Size column).

8.15.2.10 Options

Applies to
dcFolderListView component.

Declaration
```
type
  TdcFolderListViewOptions = set of (floOpenFiles, floExploreFolders,
floShowContextMenu, floAllowDelete, floOleDrag, floOleDrop);

property Options: TdcFolderListViewOptions;
```

Description
The Options property specifies behaivor of the dcFolderListView, giving to the list view or restricting some abilities of standard Windows Explorer.

There are possible flags:

Value	Meaning
floOpenFiles	Determines whether the user can open files or execute programs doubleclicking the file icon or pressing the "Enter" on selected list item. (If *floExploreFolders* flag is False and user doubleclicks the folder, it will opens the folder in separate window.);

© 1999-2001, UtilMind Solutions®

floExploreFolders	Determines whether user can pass to the subfolders doubleclicking the folder icon image (or selecting "Explore" item in the context menu for the list item, when *floShowContextMenu* is True). When *floExploreFolder* is True, user also can press "Enter" to pass to the subfolder, or press "Backspace" to return back to the up-level folder;
floShowContextMenu	Whether the context menu for the shell object should pops up when user clicks right mouse button on list item;
floAllowDelete	Determines whether the FolderListView should allow deleting files when user clicks "Del" button or selects "Delete" item in the context menu. When *floAllowDelete* is False you can handle deleting programatically calling the DeleteSelectedFiles method. If you want to disallow user to rename files — just set ReadOnly property of the dcFolderListView to False.
floOleDrag	enables automatic Drag support via OLE. User can drag (move) files to any Windows Explorer window.
floOleDrop	automatic Drop support via OLE. When floOleDrop = True, user can drop to the FolderListView any files from the Windows Explorer.

See also
Folder property and DeleteSelectedFiles method.

8.15.2.11 RegistrySaver

Applies to
dcFolderListView component.

Declaration
```
type
  TdcListViewRegistrySaver = class
  published
    property Enabled: Boolean;
    property RegKey: String;
    property RegLocation: TdcRegLocation;
  end;

property RegistrySaver: TdcListViewRegistrySaver;
```

Description
The RegistrySaver structure intended to save and restore the sorting rules, ViewStyle and widths of columns on every exit and startup of your program.

Just specify the RegLocation and RegKey where to store these settings, and set Enabled to True. The SortColumn, SortDirection properties and widths of the list columns will be saved on every exit, to be restored on every program restart.

8.15.2.11.1 Enabled

Applies to
dcFolderListView component as subproperty of RegistrySaver structure.

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

Description
The Enabled property of the RegistrySaver structure controls whether you would like to save the

© 1999-2001, UtilMind Solutions®

sorting rules (SortColumn and SortDirection properties), ViewStyle, and widths of list columns on every program exit, to restore them on every program restart. Set Enabled to True to automatically save and restore appearance of the dcFolderListView component.

See also
RegKey and RegLocation properties.

8.15.2.11.2 RegKey

Applies to
dcFolderListView component as subproperty of RegistrySaver structure.

Declaration
```
property RegKey: String;
```

Description
The RegKey property defines the path to the key in the system registry where you would like to store the ListView's settings (sorting rules and widths of columns). All stored settings will be updated every time when user close the form

Default property value is `'\Software\CompanyName\ProgramName\ListView'`. You should point the RegKey to location where stored the configuration settings of your own application. For example: `'\Software\UtilMind Solutions\The Cool Program\Main ListView'`.

See also
RegLocation and Enabled properties.

8.15.2.11.3 RegLocation

Applies to
dcFolderListView component as subproperty of RegistrySaver structure.

Declaration
```
type
  TdcRegLocation = (rlCurrentUser, rlLocalMachine);

property RegLocation: TdcRegLocation;
```

Description
The RegLocation specifies the root key of the hierarchy of subkeys, where you would like to store the ListView's settings (sorting rules and widths of columns). RegLocation property can point to HKEY_CURRENT_USER (rlCurrentUser) or HKEY_LOCAL_MACHINE (rlLocalMachine) root keys.

When RegLocation is `rlCurrentUser`, dcFolderListView stores all settings in the registry section for current logged user. When RegLocation is `rlLocalMachine` settings will be saved for ALL users of current machine.

See also
RegKey and Enabled properties.

8.15.2.12ShowAllFolders

Applies to
dcFolderListView component.

Declaration
```
property ShowAllFolders: Boolean; // True by default
```

© 1999-2001, UtilMind Solutions®

Description
The ShowAllFolders property controls whether you would like to show ALL directories, which even does not match to the searching mask (FileMask property).

Set ShowAllFolders and *saDirectory* flag in FileAttributes to True, to show all folders within the search path.

If you don't want to show folders at all — just set *saDirectory* flag in FileAttributes to False.

This property very useful in situation when you would like to find and show only files with certain type(s) + would like to show all directories (with any extension) withiin the scanning folder. ShowAllFolders are similiar to FindAllFolders property of the dcDiskScanner component.

The dcFolderListView will not show ANY folders if *saDirectory* flag of FileAttributes property is False.

See also
FileMask and FileAttributes properties;
FindAllFolders property of dcDiskScanner component.

8.15.2.13 ShowFileExtensions

Applies to
dcFolderListView component.

Declaration
```
property ShowFileExtensions: Boolean;
```

Description
The ShowFileExtensions property controls whether the dcFolderListView controls must show the file extensions for known file types (file types described in the shell). By default ShellFileExtensions is False, however, if you would like to show the extensions for all file types — set it to True.

8.15.2.14 ShowHeaderMenu

Applies to
dcFolderListView component.

Declaration
```
property ShowHeaderMenu: Boolean;
```

Description
The ShowHeaderMenu property specifies whether you would like to use the popup menu for the header sections.

See also
HeaderMenu structure.

8.15.2.15 ShowSortMark

Applies to
dcFolderListView component.

Declaration
```
property ShowSortMark: Boolean;
```

© 1999-2001, UtilMind Solutions®

Description
The ShowSortMark property specifies whether you would like to show the arrow-style sort mark (or) on the header sections, which indicates the sorting direction of the column. Set ShowSortMark to False if you don't want to show these marks.

The sort mark can be "raised" or "sunken". To change the style of sort mark use SortMarkKind property.

Note
The sort mark will be invisible anyway, if SortType property is stNone.

See also
SortMarkKind, SortColumn and SortDirection properties.

8.15.2.16 ShowToolTips

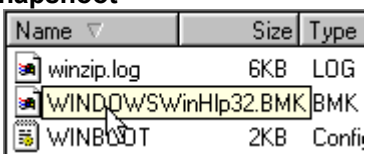
Applies to
dcFolderListView component.

Declaration
```
property ShowToolTips: Boolean;
```

Description
The ShowToolTips property controls whether you would like to show tooltips (hints) when mouse pointer moves over partially hidden list item.

To specify behavior of the tooltips — use ToopTipOptions property.

Snapshoot



See also
ToolTipOptions property.

8.15.2.17 SortColumn

Applies to
dcFolderListView component.

Declaration
```
property SortColumn: Word;
```

Description
The SortColumn property determines and specifies the number of column which should be sorted. To specify the sorting direction — use SortDirection property.

See also
SortDirection and ShowSortMark properties.

8.15.2.18 SortDirection

Applies to
dcFolderListView component.

© 1999-2001, UtilMind Solutions®

Declaration

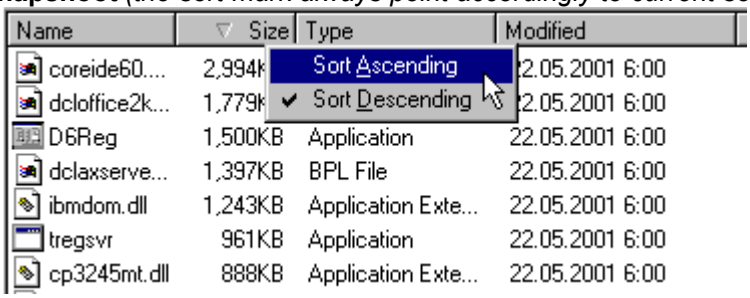
```
type
  TdcListViewSortDirection = (sdAscending, sdDescending);

property SortDirection: TdcListViewSortDirection;
```

Description

The SortDirection property controls the sorting direction for currently sorted column. The direction can be ascending (sdAscending) or descending (sdDescending).

User can change the sorting direction at run-time, clicking the column header or selecting "Sort Ascending" or "Sort Descending" menu items in the header popup menu.

Snapshoot *(the sort mark always point accordingly to current sorting direction)*



The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

See also

SortColumn, ShowSortMark and HeaderMenu properties.

8.15.2.19SortMarkAlign

Applies to

dcFolderListView component.

Declaration

```
type
  TdcListViewSortMarkAlign = (maText, maEdge);

property SortMarkAlign: TdcListViewSortMarkAlign;
```

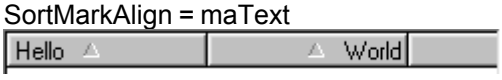
Description

The SortMarkAlign property controls placement of the arrow-style sort mark, which indicates current sorted column and sorting direction.

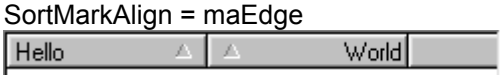
The sort mark can be displayed near the text (maText) or near the edge (maEdge) of header section.

Snapshots

SortMarkAlign = maText



SortMarkAlign = maEdge



Notes

© 1999-2001, UtilMind Solutions®

If you don't want to show the sort mark at all — set ShowSortMark property to False.
The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

See also
SortMarkKind, ShowSortMark, SortColumn and SortDirection properties.

8.15.2.20SortMarkKind

Applies to
dcFolderListView component.

Declaration
```
type
  TdcListViewSortMarkKind = (mkSunken, mkRaised);

property SortMarkKind: TdcListViewSortMarkKind;
```

Description
The SortMarkKind property controls the appearance of the arrow-style sort mark, which indicates current sorted column and sorting direction.

The sort mark can be "sunken" (mkSunken, by default) or "raised" (mkRaised).

Snapshots

or

Notes
If you don't want to show the sort mark — set ShowSortMark property to False.
The arrow-style sort mark can be invisible if the list is unsorted (SortType = stNone).

See also
SortMarkAlign, ShowSortMark, SortColumn and SortDirection properties.

8.15.2.21ToolTipOptions

Applies to
dcFolderListView component.

Declaration
```
type
  TdcListViewToolTipOption = (ttoInstantShow, ttoLongStay, ttoBelowItem,
ttoOffset);
  TdcListViewToolTipOptions = set of TdcListViewToolTipOption;

property ToolTipOptions: TdcListViewToolTipOptions;
```

Description
The ToolTipOptions property controls behavior of the tooltips (hints) for partially hidden list items.

There are possible values:

Value	Meaning
ttoInstantShow	show the tooltips instantly when the mouse overs the list item (without a second delay);
ttoLongStay	tooltips does not disappears until user move mouse pointer away from the list item, when ttoLongStay is True. Usually tooltips stays visible only 5 seconds;

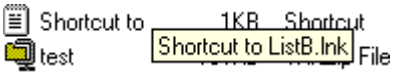
© 1999-2001, UtilMind Solutions®

ttoBelowItem	the tooltip will be displayed below item (will not cover it by itself);
ttoOffset	offsets the tooltip to display it to the right and below the item.

Snapshoots

ttoBelowItem = True

ttoOffset = True



See also

ShowToolTips property.

8.15.3 Methods

8.15.3.1 Back

Applies to

dcFolderListView component.

Declaration

```
procedure Back;
```

Description

Use Back method to pass to up-level folder (under that folder which currently specified in Folder property).

Example

```
procedure TForm1.UpBtnClick(Sender: TObject);
begin
  dcFolderListView1.Back;
end;

procedure TForm1.dcFolderListView1FolderChanged(Sender: TObject;
  Folder: String);
begin
  UpBtn.Enabled := dcFolderListView1.IsBackPossible;
  dcFolderEdit1.Text := Folder;
end;
```

See also

IsBackPossible method and Folder property.

8.15.3.2 CreateFolder

Applies to

dcFolderListView component.

Declaration

```
procedure CreateFolder(FolderName: String; Edit: Boolean);
```

Description

The CreateFolder method creates new folder and adds new list item (which represents new folder) to

© 1999-2001, UtilMind Solutions®

the list view and specifies for this folder default name (*FolderName* parameter).

If you want to allow user to edit the name of folder at once after its creation — set *Edit* parameter to True.

Example

```
procedure TForm1.PopupMenuFolder1Click(Sender: TObject);
begin
  dcFolderListView1.CreateFolder('New Folder', True);
end;
```

See aslo

CreateShortcut method.

8.15.3.3 CreateShortcut

Applies to

dcFolderListView component.

Declaration

```
procedure CreateShortcut;
```

Description

The CreateShortcut method executes "Create Shortcut" dialog which lets user to create the shell link in current directory (specified in the Folder property).

Example

```
procedure TForm1.PopupMenuShortcut1Click(Sender: TObject);
begin
  dcFolderListView1.CreateShortcut;
end;
```

See aslo

CreateShortcut method and Folder property.

8.15.3.4 DeleteSelectedFiles

Applies to

dcFolderListView component.

Declaration

```
function DeleteSelectedFiles: Boolean; // returns True if succeed
```

Description

The DeleteSelectedFiles property deletes the list items of the dcFolderListView, which represents the real files and folders. The deleted files will be moved to "Recycled bin".

When floAllowDelete flag is True in Options, user are able to delete files selecting the list items and clicking the "Del" button, or selecting the "Delete" item in the context menu for the shell object. In this case, the FolderListView will delete file(s) and/or folder(s) automatically, without necessity to call this method.

See also

Options property and Folder property.

© 1999-2001, UtilMind Solutions®

8.15.3.5 FileAttributesByListItem

Applies to
dcFolderListView component.

Declaration
```
function FileAttributesByListItem(ListItem: TListItem):
  TdcScanAttributes;
```

Description
The FileAttributesByListItem method returns the file attributes by list item of the dcFolderListView.

Example
```
procedure TForm1.dcFolderListView1Click(Sender: TObject);
var
   Pos: TPoint;
   ListItem: TListItem;
begin
   with dcFolderListView1 do
    begin
     { determination of the mouse position }
     GetCursorPos(Pos);
     Pos := ScreenToClient(Pos);

     { determination of the list item by mouse position }
     ListItem := GetItemAt(Pos.X, Pos.Y);
     if ListItem <> nil then
      if saDirectory in FileAttributesByListItem(ListItem) then
       begin
        ShowMessage('Click on directory');
       end;
    end;
end;
```

See also
FileNameByListItem, FileSizeByListItem, FileTypeByListItem, FileTimeByListItem, FileAttributesByListItem and ListItemByFileName methods.

8.15.3.6 FileNameByListItem

Applies to
dcFolderListView component.

Declaration
```
function FileNameByListItem(ListItem: TListItem): String;
```

Description
The FileNameByListItem method returns the full path and the file name with extension by list item of the dcFolderListView.

Example
```
procedure TForm1.dcFolderListView1MouseDown(Sender: TObject;
   Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
   ListItem: TListItem;
begin
   with dcFolderListView1 do
```

© 1999-2001, UtilMind Solutions®

```
    begin
      ListItem := GetItemAt(X, Y);
      if ListItem <> nil then
        Edit1.Text := FileNameByListItem(ListItem);
    end;
  end;
```

See also
FileNameByListItem, FileSizeByListItem, FileTypeByListItem, FileTimeByListItem, FileAttributesByListItem and ListItemByFileName methods.

8.15.3.7 FileSizeByListItem

Applies to
dcFolderListView component.

Declaration
```
function FileSizeByListItem(ListItem: TListItem): Extended;
```

Description
The FileSizeByListItem method returns the file size by list item of the dcFolderListView.

Use FloatToStr function to convert the returned value to string.

See also
FileNameByListItem, FileSizeByListItem, FileTypeByListItem, FileTimeByListItem, FileAttributesByListItem and ListItemByFileName methods.

8.15.3.8 FileTimeByListItem

Applies to
dcFolderListView component.

Declaration
```
function FileTimeByListItem(ListItem: TListItem): TDateTime;
```

Description
The FileSizeByListItem method returns the date and time of last modification (last usage) of the file represented by list item of the dcFolderListView.

See also
FileNameByListItem, FileSizeByListItem, FileTypeByListItem, FileTimeByListItem, FileAttributesByListItem and ListItemByFileName methods.

8.15.3.9 FileTypeByListItem

Applies to
dcFolderListView component.

Declaration
```
function FileTypeByListItem(ListItem: TListItem): String;
```

Description
The FileTypeByListItem method returns the file type of the file which represented by list item of the dcFolderListView component.

See also

© 1999-2001, UtilMind Solutions®

FileNameByListItem, FileSizeByListItem, FileTypeByListItem, FileTimeByListItem, FileAttributesByListItem and ListItemByFileName methods.

8.15.3.10 GetColumnAt

Applies to
dcFolderListView component.

Declaration
```
function GetColumnAt(X: Integer): TListColumn;
```

Description
The GetColumnAt method returns the list column at the specified X-coordinate.

Call GetColumnAt to obtain the TListColumn object for the list item found at the position specified by the X parameter. X specify the coordinate of the position, in pixels, relative to the top left corner of the list view. If there is no item at the location, GetColumnAt returns nil.

Example
```
procedure TForm1.dcFolderListView1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  Column: TListColumn;
begin
  Column := dcFolderListView1.GetColumnAt(X);
  if Column <> nil then
    Caption := Column.Caption;
end;
```

See also
GetItemAt method.

8.15.3.11 GetItemAt

Applies to
dcFolderListView component.

Declaration
```
function GetItemAt(X, Y: Integer): TListItem;
```

Description
The GetItemAt method returns the list item at the specified position.

Call GetItemAt to obtain the TListItem object for the list item found at the position specified by the X and Y parameters. X and Y specify the coordinates of the position, in pixels, relative to the top left corner of the list view. If there is no item at the location, GetItemAt returns nil.

Example
```
procedure TForm1.dcFolderListView1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
  Item: TListItem;
begin
  Item := dcFolderListView1.GetItemAt(X, Y);
  if Item <> nil then
    Caption := Item.Caption;
end;
```

© 1999-2001, UtilMind Solutions®

See also
GetColumnAt method.

8.15.3.12IsBackPossible

Applies to
dcFolderListView component.

Declaration
```
function IsBackPossible: Boolean;
```

Description
The IsBackPossible method returns True if current Folder is not root-directory and it is possible to go to up-level folder. When it returns False, this means that the dcFolderListView currently shows content of the root folder (i.e: C:\, D:\ etc).

Example
```
procedure TForm1.UpBtnClick(Sender: TObject);
begin
  dcFolderListView1.Back;
end;

procedure TForm1.dcFolderListView1FolderChanged(Sender: TObject;
  Folder: String);
begin
  UpBtn.Enabled := dcFolderListView1.IsBackPossible;
  dcFolderEdit1.Text := Folder;
end;
```

See also
Back method and Folder property.

8.15.3.13IsScrollBarsVisible

Applies to
dcFolderListView component.

Declaration
```
function IsScrollBarsVisible: Boolean;
```

Description
The IsScrollBarsVisible method determines whether the content of the ListView does not fits to the control's bounds and it shows the scrollbars.

IsScrollBarsVisible returns True if horizontal or vertical scrollbar actually are visible, or False if ListView doesn't need the scrollbars.

Example
```
while FolderListView1.IsScrollBarsVisible do
  FolderListView1.Height := FolderListView1.Height + 24;
```

8.15.3.14ListItemByFileName

Applies to
dcFolderListView component.

Declaration

© 1999-2001, UtilMind Solutions®

```
function ListItemByFileName(FileName: String): TListItem;
```

Description
The ListItemByFileName returns the list item of the dcFolderListView which represents the same file name as specified by *FileName* parameter.

ListItemByFileName always returns correct list item, since every file (represented by list item) has unique file name.

See also
FileNameByListItem, FileSizeByListItem, FileTypeByListItem, FileTimeByListItem and FileAttributesByListItem methods.

8.15.3.15Refresh

Applies to
dcFolderListView component.

Declaration
```
procedure Refresh;
```

Description
The Refresh method refreshes content of the dcFolderListView, re-scanning the content of the current Folder and retrieving information about files and directories.

When the list view content is about to be updated — OnBeginUpdate event occurs. Once the list view has finished updating — OnEndUpdate occurs.

See also
OnBeginUpdate and OnEndUpdate events.

8.15.3.16ShowFolderProperties

Applies to
dcFolderListView component.

Declaration
```
procedure ShowFolderProperties;
```

Description
The ShowItemProperties method displays the "Properties" dialog for current folder that displayed in the list view (to specify or determinate current folder use Folder property). If list view currently displaying the root folder (like C:\ or D:\), the ShowFolderProperties method will show the "Properties" for disk.

Example
```
procedure TForm1.PopupMenuProperties1Click(Sender: TObject);
begin
  dcFolderListView1.ShowFolderProperties;
end;
```

See aslo
ShowItemProperties method and Folder property.

© 1999-2001, UtilMind Solutions®

8.15.3.17 ShowItemProperties

Applies to
dcFolderListView component.

Declaration
```
procedure ShowItemProperties;
```

Description
The ShowItemProperties method executes the "Properties" dialog for selected list item (file or folder).

Example
```
procedure TForm1.PopupMenuProperties1Click(Sender: TObject);
begin
   dcFolderListView1.ShowItemProperties;
end;
```

See aslo
ShowFolderProperties method.

8.15.3.18 SelectAll

Applies to
dcFolderListView component.

Declaration
```
procedure SelectAll;
```

Description
SelectAll method selects all items of the list view.

See also
UnselectAll method.

8.15.3.19 UnselectAll

Applies to
dcFolderListView component.

Declaration
```
procedure UnselectAll;
```

Description
UnselectAll method removes selection for selected item of the list view.

See also
SelectAll method.

8.15.4 Events

8.15.4.1 OnBeginUpdate

Applies to
dcFolderListView component.

Declaration

© 1999-2001, UtilMind Solutions®

```
property OnBeginUpdate: TNotifyEvent;
```

Description

The OnBeginUpdate event occurs when the dcFolderListView needs to update (refresh) the list items which shows the content of directory, and is about to scan the directory for files to update itself.

If the directory contains too many files, the updating process can take rather long time. You can hook the OnBeginUpdate and OnEndUpdate events to amuse users by something while the FolderListView scans directory for files.

Example *(shows the avi movie while the dcFolderListView refresh the directory contents)*

```
{ Animate1.CommonAVI := aviFindFolder }
procedure TForm1.dcFolderListView1BeginUpdate(Sender: TObject);
begin
  Animate1.Active := True;
  Animate1.Visible := True;
end;

procedure TForm1.dcFolderListView1EndUpdate(Sender: TObject);
begin
  Animate1.Visible := False;
  Animate1.Active := False;
end;
```

See also

OnEndUpdate event and Refresh method.

8.15.4.2 OnDrawHeaderSection

Applies to

dcFolderListView component.

Declaration

```
type
  TacListViewDrawHeaderSection = procedure(Sender: TObject; Canvas:
TCanvas; Rect: TRect; ColumnIndex: Integer; Pressed: Boolean) of object;

property OnDrawHeaderSection: TacListViewDrawHeaderSection;
```

Description

The OnDrawHeaderSection event occurs when the header section needs to be drawn.

Write an OnDrawHeaderSection event handler to draw the contents for all the sections in the ListView. Draw on the header using the methods of the *Canvas* property. The *Rect* parameter indicates the location of the cell on the canvas. The *ColumnIndex* parameter indicate the column of the header that should be drawn. The *Pressed* parameter indicates whether the coulum is selected by mouse (pressed).

If the OnDrawHeaderSection handler is not assigned, the component will paint default content.

8.15.4.3 OnEndUpdate

Applies to

dcFolderListView component.

Declaration

© 1999-2001, UtilMind Solutions®

```
property OnEndUpdate: TNotifyEvent;
```

Description

The OnEndUpdate event occurs when the dcFolderListView has finished process of scanning for files witnin the current folder and has sucessfully refreshed itself.

If the directory contains too many files, the updating process can take rather long time. You can hook the OnBeginUpdate and OnEndUpdate events to amuse users by something while the FolderListView scans directory for files.

Example *(shows the avi movie while the dcFolderListView refresh the directory contents)*

```
{ Animate1.CommonAVI := aviFindFolder }
procedure TForm1.dcFolderListView1BeginUpdate(Sender: TObject);
begin
  Animate1.Active := True;
  Animate1.Visible := True;
end;

procedure TForm1.dcFolderListView1EndUpdate(Sender: TObject);
begin
  Animate1.Visible := False;
  Animate1.Active := False;
end;
```

See also

OnBeginUpdate event and Refresh method.

8.15.4.4 OnFolderChanged

Applies to

dcFolderListView component.

Declaration

```
type
  TdcFolderChangedEvent = procedure(Sender: TObject; Folder: String) of
object;

property OnFolderChanged: TdcFolderChangedEvent;
```

Description

The OnFolderChanged event occurs when user passes to another subfolder (i.e: clicking the folder icon in the list view), or when you call Back method (i.e: when user clicks the "Up" button to go to up-level folder).

The *Folder* parameter determines the directory in which user passed to. At once after occuring this event, the content of the list view will be automatically refreshed.

Note

The OnFolderChanged event will not occurs when you changing the Folder property programatically.

Example

```
procedure TForm1.dcFolderListView1FolderChanged(Sender: TObject;
  Folder: String);
begin
  dcFolderEdit1.Text := Folder;
```

© 1999-2001, UtilMind Solutions®

```
  UpBtn.Enabled := dcFolderListView1.IsBackPossible;
end;

procedure TForm1.UpBtnClick(Sender: TObject);
begin
  dcFolderListView1.Back;
end;
```

See also
Folder property and Back method.

8.15.4.5 OnMouseEnter

Applies to
dcFolderListView, dcFileEdit and dcFolderEdit components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the listview control.

See also
OnMouseLeave event.

8.15.4.6 OnMouseLeave

Applies to
dcFolderListView, dcFileEdit and dcFolderEdit components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the listview control.

See also
OnMouseEnter event.

8.15.4.7 OnSettingsChanged

Applies to
dcFolderListView component.

Declaration
```
property OnSettingsChanged: TNotifyEvent;
```

Description
The OnSettingsChanged event occurs when any system settings (system metrics or regional settings) is changed.

Write OnSettingsChanged event handler to refresh the list items when any system settings is changed. For example, to refresh items which contains date/time values when the date/time format is changed. (The format settings in your application will be already updated automatically.)

© 1999-2001, UtilMind Solutions®

8.15.4.8 OnToolTip

Applies to
dcFolderListView component.

Declaration
```
type
  TacListViewToolTipEvent = procedure(Sender: TObject; var ToolTipText:
String; Item: TListItem; LogicalCol: Integer) of object;

property OnToolTip: TacListViewToolTipEvent;
```

Description
The OnToolTip event occurs when the dcFolderListView is about to show the tooltip (hint) for some partially hidden list item.

The text for the tooltip, which is about to be displayed, transmitted to the event handler in the *ToolTipText* parameter. You can change it in the even handler.

To determinate the list item under mouse pointer — use *Item* parameter. To determinate whether this item was a subitem — read the column number from the *LogicalCol* parameter.

See also
ShowToolTip and ToolTipOptions properties.

8.16 dcFileEdit

8.16.1 TdcFileEdit

Overview
The dcFileEdit component is the enchanced TEdit component with the custom button at the right side of edit control, and can be used to select the filename

8.16.2 Properties

8.16.2.1 Align

Applies to
dcFileEdit, dcFolderEdit and dcFolderListView components.

Declaration
```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Description
The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form

© 1999-2001, UtilMind Solutions®

or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note

If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

8.16.2.2 Alignment

Applies to

dcFileEdit and dcFolderEdit components.

Declaration

```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property Alignment: TAlignment;
```

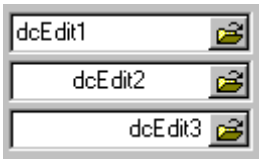
Description

The Alignment property controls horizontal placement of the text within the edit control. Set Alignment to specify how the text of the edit control is aligned within the client area (ClientRect property) of this control. Alignment can be one of the following values:

Value	Meaning
taLeftJustify	The text appears lined up along the left edge of the edit control;
taRightJustify	The text appears lined up along the right edge of the edit control;
taCenter	The text is horizontally centered in the edit control.

Snapshot

© 1999-2001, UtilMind Solutions®



8.16.2.3 AutoSelect

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property AutoSelect: Boolean; // True by default
```

Description
The AutoSelect property controls whether the text in the edit control should be automatically selected when user moves to the control pressing *Tab* key, activating input focus of the edit control.

8.16.2.4 Button

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
type
  TdcEditButton = class(TPersistent)
  published
    property Cursor: TCursor;
    property Flat: Boolean; // in Delphi 3 and later !!!
    property Glyph: TBitmap;
    property NumGlyphs: TNumGlyphs; // 1..4
    property Hint: String;
    property Width: Word;
    property Visible: Boolean; // False by default
  end;
```

Description
The dcFileEdit component is able to have a custom button at the right edge of edit control. The Button structure intended for specify settings for this button. To make button visible on edit control — set Button.Visible to True.

You can specify the glyph image and number of images, hint, cursor and width for button.

The button which at the right of edit control is just usual TSpeedButton component. You can even gain direct access to all button's properties using hidden **EditButton** property.
For example, to specify caption for this button you may use following code:
```
dcFileEdit1.EditButton.Caption := 'Test';
```

See also
OnButtonClick event.

8.16.2.4.1 Cursor

Applies to
dcFileEdit and dcFolderEdit components as subproperty of Button structure.

Declaration

© 1999-2001, UtilMind Solutions®

```
property Cursor: TCursor;
```

Description
The Cursor property specifies the image for mouse pointer when it passes over the custom button at the right side of edit control.

See also
CursorBorder property.

8.16.2.4.2 Flat

Applies to
dcFileEdit and dcFolderEdit components as subproperty of Button structure.

Declaration
```
property Flat: Boolean;
```

Description
The Flat property determines whether the button has a 3D border that provides a raised or lowered look.

Set Flat to True to remove the raised border when the button is unselected and the lowered border when the button is clicked or selected. When Flat is True, use separate bitmaps for the different button states to provide visual feedback to the user about the button state.

8.16.2.4.3 Glyph

Applies to
dcFileEdit and dcFolderEdit components as subproperty of Button structure.

Declaration
```
property Glyph: TBitmap;
```

Description
The Glyph property specifies the bitmap that appears on the custom button at the right side of edit control.

Set Glyph to a bitmap object that contains the image that should appear on the face of the button. Bring up the Open dialog box from the Object Inspector to choose a bitmap file (with a .BMP extension), or specify a bitmap file at runtime.

Glyph can provide up to four images within a single bitmap. All images must be the same size and next to each other in a horizontal row. Button displays one of these images depending on the state of the button.

Image position	Buttons state	Description
First	Up	This image appears when the button is unselected. If no other images exist in the bitmap, this image is used for all states.
Second	Disabled	This image usually appears dimmed to indicate that the button can't be selected.
Third	Clicked	This image appears when the button is clicked. The Up image reappears when the user releases the mouse button.
Fourth	Down	This image appears when the button stays down indicating that it remains selected.

© 1999-2001, UtilMind Solutions®

If only one image is present, Button attempts to represent the other states by altering the image slightly for each state, although the Down state is always the same as the Up state.

If the bitmap contains multiple images, specify the number of images in the bitmap with the NumGlyphs property.

Note
The lower left hand pixel of the bitmap is reserved for the "transparent" color. Any pixel in the bitmap which matches that lower left hand pixel will be transparent.

See also
NumGlyphs property.

8.16.2.4.4 NumGlyphs

Applies to
dcFileEdit and dcFolderEdit components as subproperty of Button structure.

Declaration
```
property NumGlyphs: TNumGlyphs; // 1..4
```

Description
The NumGlyphs is the number of images specified in the Glyph property.

Set NumGlyphs to the number of images provided by the bitmap assigned to the Glyph property. All images must be the same size and next to each other in a row. The Glyph property can provide up to four images.

See also
Glyph property.

8.16.2.4.5 Hint

Applies to
dcFileEdit and dcFolderEdit components as subproperty of Button structure.

Declaration
```
property Hint: String;
```

Description
The Hint property contains the text string that can appear when the user moves the mouse pointer over the button at right side of edit control.

8.16.2.4.6 Width

Applies to
dcFileEdit and dcFolderEdit components as subproperty of Button structure.

Declaration
```
property Width: Integer;
```

Description
The Width property controls the horizontal size of the button in pixels.

The button's height controlled by height of dcFileEdit control.

© 1999-2001, UtilMind Solutions®

8.16.2.4.7 Visible

Applies to
dcFileEdit and dcFolderEdit components as subproperty of Button structure.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the button appears at the right side of dcFileEdit control.

Use the Visible property to control the visibility of the button at runtime. If Visible is True, the button appears. If Visible is False, the button is not visible.

8.16.2.5 Color

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property Color: TColor;
```

Description
The Color property for dcFileEdit component is the background color of the edit control. Use Color to read or change the background color of the control.

If a control's ParentColor property is True, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

See also
ColorDisabled property.

8.16.2.6 ColorDisabled

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property ColorDisabled: TColor; // clBtnFace by default
```

Description
The ColorDisabled property for dcFileEdit component is the background color of the edit control at disabled state (when Enabled is False). Use the ColorDisabled property to specify color for disabled edit control.

See also
Color property.

8.16.2.7 CursorBorder

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property CursorBorder: TCursor;
```

© 1999-2001, UtilMind Solutions®

Description
The CursorBorder property specifies the image used to represent the mouse pointer when it passes over the non-client area of edit control.

See also
Cursor property of the button.

8.16.2.8 DlgDefaultExt

Applies to
dcFileEdit component.

Declaration
```
property DlgDefaultExt: String;
```

Description

8.16.2.9 DlgFilter

Applies to
dcFileEdit component.

Declaration
```
property DlgFilter: String;
```

Description

8.16.2.10 DlgOptions

Applies to
dcFileEdit component.

Declaration
```
type
 TOpenOption = (ofReadOnly, ofOverwritePrompt, ofHideReadOnly,
   ofNoChangeDir, ofShowHelp, ofNoValidate, ofAllowMultiSelect,
   ofExtensionDifferent, ofPathMustExist, ofFileMustExist,
   ofCreatePrompt, ofShareAware, ofNoReadOnlyReturn, ofNoTestFileCreate,
   ofNoNetworkButton, ofNoLongNames, ofOldStyleDialog,
   ofNoDereferenceLinks);
  TOpenOptions = set of TOpenOption;

property DlgOptions: TOpenOptions;
```

Description

8.16.2.11 DlgTitle

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property DlgTitle: String;
```

© 1999-2001, UtilMind Solutions®

Description
The DlgTitle property specifies the caption for the dialog box, which opens after clicking the button at the right side of edit control.

See also
Title property of the dcOpenDialog component.

8.16.2.12 DlgType

Applies to
dcFileEdit component.

Declaration
```
type
  TdcFileEditDlgTypes = (dtOpen, dtSave, dtOpenPicture, dtSavePicture);

property DlgType: TdcFileEditDlgTypes;
```

Description

8.16.3 Events

8.16.3.1 OnButtonClick

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property OnButtonClick: TNotifyEvent;
```

Description
The OnButtonClick event occurs when user clicks the button at the right side of edit control, to select the folder. The OnButtonClick for the dcFileEdit component occurs before displaying the "Open file" or "Save to file" dialog. If you would like to handle clicks on "Ok" and "Cancel" buttons of this dialog box — use OnDlgOk and OnDlgCancel events.

To make the custom button appear on the edit control — set Button.Visible to True, or to False if you'd like to hide the button.

See also
OnDlgOk and OnDlgCancel events.

8.16.3.2 OnDlgOk

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property OnDlgOk: TNotifyEvent;
```

Description
The DlgOk event occurs when user selects the file in the dialog box (which opens after clicking the button at the right side of edit control) and clicks the "Ok" button (confirms selection of the file).

See also
OnDlgCancel and OnButtonClick events.

© 1999-2001, UtilMind Solutions®

8.16.3.3 OnDlgCancel

Applies to
dcFileEdit and dcFolderEdit components.

Declaration
```
property OnDlgCancel: TNotifyEvent;
```

Description
The DlgCancel event occurs when user close the dialog box (which opens after clicking the button at the right side of edit control) without selection of the file, or clicks the "Cancel" button.

See also
OnDlgOk and OnButtonClick events.

8.16.3.4 OnMouseEnter

Applies to
dcFileEdit, dcFolderEdit and dcFolderListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the edit control.

See also
OnMouseLeave event.

8.16.3.5 OnMouseLeave

Applies to
dcFileEdit, dcFolderEdit and dcFolderListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the edit control.

See also
OnMouseEnter event.

8.17 dcFolderEdit

8.17.1 TdcFolderEdit

Overview
The dcFolderEdit component is the enchanced TEdit component with the custom button at the right side of edit control, and can be used to select the filename

Snapshoot

See also

© 1999-2001, UtilMind Solutions®

dcBrowseDialog and dcFileEdit components.

8.17.2 Properties

8.17.2.1 Align

Applies to
dcFolderEdit, dcFileEdit and dcFolderListView components.

Declaration

```
type
  TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);

property Align: TAlign;
```

Description
The Align property determines how the control aligns within its container (or parent control).

Use Align to align a control to the top, bottom, left, or right of a form or panel and have it remain there even if the size of the form, panel, or component that contains the control changes.

For example, to use a panel component with various controls on it as a tool palette, change the panel's Align value to alLeft. The value of alLeft for the Align property of the panel guarantees that the tool palette remains on the left side of the form and always equals the client height of the form.

The default value of Align is alNone, which means a control remains where it is positioned on a form or panel. Specify another value to align the control. Use one of these possible values:

Value	Meaning
alNone	The component remains where it was placed on the form or panel. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area.

Note
If Align is set to alClient, the control fills the entire client area so that it is impossible to select the parent form by clicking on it. In this case, select the parent by selecting the control on the form and pressing Esc, or by using the Object Inspector.

Any number of child components within a single parent can have the same Align value, in which case they stack up along the edge of the parent form or panel. The child controls stack up in z-order. To adjust the order in which the controls stack up, drag the controls into their desired positions.

© 1999-2001, UtilMind Solutions®

8.17.2.2 Alignment

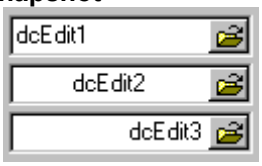
Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
type
  TAlignment = (taLeftJustify, taRightJustify, taCenter);

property Alignment: TAlignment;
```

Description
The Alignment property controls horizontal placement of the text within the edit control. Set Alignment to specify how the text of the edit control is aligned within the client area (ClientRect property) of this control. Alignment can be one of the following values:

Value	Meaning
taLeftJustify	The text appears lined up along the left edge of the edit control;
taRightJustify	The text appears lined up along the right edge of the edit control;
taCenter	The text is horizontally centered in the edit control.

Snapshot



8.17.2.3 AutoSelect

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property AutoSelect: Boolean; // True by default
```

Description
The AutoSelect property controls whether the text in the edit control should be automatically selected when user moves to the control pressing *Tab* key, activating input focus of the edit control.

8.17.2.4 Button

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
type
  TdcEditButton = class(TPersistent)
  published
    property Cursor: TCursor;
    property Flat: Boolean; // in Delphi 3 and later !!!
    property Glyph: TBitmap;
    property NumGlyphs: TNumGlyphs; // 1..4
    property Hint: String;
    property Width: Word;
    property Visible: Boolean; // False by default
```

© 1999-2001, UtilMind Solutions®

```
end;
```

Description

The dcFolderEdit component is able to have a custom button at the right edge of edit control. The Button structure intended for specify settings for this button. To make button visible on edit control — set Button.Visible to True.

You can specify the glyph image and number of images, hint, cursor and width for button.

The button which at the right of edit control is just usual TSpeedButton component. You can even gain direct access to all button's properties using hidden **EditButton** property.
For example, to specify caption for this button you may use following code:

```
dcFolderEdit1.EditButton.Caption := 'Test';
```

See also

OnButtonClick event.

8.17.2.4.1 Cursor

Applies to

dcFolderEdit and dcFileEdit components as subproperty of Button structure.

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property specifies the image for mouse pointer when it passes over the custom button at the right side of edit control.

See also

CursorBorder property.

8.17.2.4.2 Flat

Applies to

dcFolderEdit and dcFileEdit components as subproperty of Button structure.

Declaration

```
property Flat: Boolean;
```

Description

The Flat property determines whether the button has a 3D border that provides a raised or lowered look.

Set Flat to True to remove the raised border when the button is unselected and the lowered border when the button is clicked or selected. When Flat is True, use separate bitmaps for the different button states to provide visual feedback to the user about the button state.

8.17.2.4.3 Glyph

Applies to

dcFolderEdit and dcFileEdit components as subproperty of Button structure.

Declaration

```
property Glyph: TBitmap;
```

Description

© 1999-2001, UtilMind Solutions®

The Glyph property specifies the bitmap that appears on the custom button at the right side of edit control.

Set Glyph to a bitmap object that contains the image that should appear on the face of the button. Bring up the Open dialog box from the Object Inspector to choose a bitmap file (with a .BMP extension), or specify a bitmap file at runtime.

Glyph can provide up to four images within a single bitmap. All images must be the same size and next to each other in a horizontal row. Button displays one of these images depending on the state of the button.

Image position	Buttons state	Description
First	Up	This image appears when the button is unselected. If no other images exist in the bitmap, this image is used for all states.
Second	Disabled	This image usually appears dimmed to indicate that the button can't be selected.
Third	Clicked	This image appears when the button is clicked. The Up image reappears when the user releases the mouse button.
Fourth	Down	This image appears when the button stays down indicating that it remains selected.

If only one image is present, Button attempts to represent the other states by altering the image slightly for each state, although the Down state is always the same as the Up state.

If the bitmap contains multiple images, specify the number of images in the bitmap with the NumGlyphs property.

Note

The lower left hand pixel of the bitmap is reserved for the "transparent" color. Any pixel in the bitmap which matches that lower left hand pixel will be transparent.

See also

NumGlyphs property.

8.17.2.4.4 NumGlyphs

Applies to

dcFolderEdit and dcFileEdit components as subproperty of Button structure.

Declaration

```
property NumGlyphs: TNumGlyphs; // 1..4
```

Description

The NumGlyphs is the number of images specified in the Glyph property.

Set NumGlyphs to the number of images provided by the bitmap assigned to the Glyph property. All images must be the same size and next to each other in a row. The Glyph property can provide up to four images.

See also

Glyph property.

© 1999-2001, UtilMind Solutions®

8.17.2.4.5 Hint

Applies to
dcFolderEdit and dcFileEdit components as subproperty of Button structure.

Declaration
```
property Hint: String;
```

Description
The Hint property contains the text string that can appear when the user moves the mouse pointer over the button at right side of edit control.

8.17.2.4.6 Width

Applies to
dcFolderEdit and dcFileEdit components as subproperty of Button structure.

Declaration
```
property Width: Integer;
```

Description
The Width property controls the horizontal size of the button in pixels.

The button's height controlled by height of dcFolderEdit control.

8.17.2.4.7 Visible

Applies to
dcFolderEdit and dcFileEdit components as subproperty of Button structure.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property determines whether the button appears at the right side of dcFolderEdit control.

Use the Visible property to control the visibility of the button at runtime. If Visible is True, the button appears. If Visible is False, the button is not visible.

8.17.2.5 Color

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property Color: TColor;
```

Description
The Color property for dcFolderEdit component is the background color of the edit control. Use Color to read or change the background color of the control.

If a control's ParentColor property is True, then changing the Color property of the control's parent automatically changes the Color property of the control. When the value of the Color property is changed, the control's ParentColor property is automatically set to False.

See also
ColorDisabled property.

© 1999-2001, UtilMind Solutions®

8.17.2.6 ColorDisabled

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property ColorDisabled: TColor; // clBtnFace by default
```

Description
The ColorDisabled property for dcFolderEdit component is the background color of the edit control at disabled state (when Enabled is False). Use the ColorDisabled property to specify color for disabled edit control.

See also
Color property.

8.17.2.7 CursorBorder

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property CursorBorder: TCursor;
```

Description
The CursorBorder property specifies the image used to represent the mouse pointer when it passes over the non-client area of edit control.

See also
Cursor property of the button.

8.17.2.8 DlgOptions

Applies to
dcFolderEdit component.

Declaration
```
type
  TdcBrowseDialogOptions = set of (soShowEditBox, soShowFiles,
     soShowStatusText, soShowPathInStatus, soShowSysAncestors,
     soShowNetworkFiles, soSelectRootFolders);

property Options: TdcBrowseDialogOptions;
```

Description
The Options property specifies advanced options for folder-selection dialog. There are following possible options:

Value	Meaning
soShowEditBox	show edit box and allow to specify location typing the path in the edit box;
soShowFiles	show files and allow to select them;
soShowStatusText	show the status line;
soShowPathInStatus	show the path to selected folder in the status line;
soShowSysAncestors	show system shell folders (i.e: "Control Panel", "Printers" etc)
soShowNetworkFiles	show files in local area network;
soSelectRootFolders	allow to select root folders (i.e: C:\ or D:\).

© 1999-2001, UtilMind Solutions®

See also
DlgStatusText property;
Options property of the dcBrowseDialog component.

8.17.2.9 DlgSpecialLocation

Applies to
dcFolderEdit component.

Declaration
```
type
  TdcBrowseDialogSpecialLocation =
         (slNone, slDesktop,
          slPrograms, slControlPanel,
          slPrinters, slPersonal,
          slFavorites, slStartup,
          slRecent, slSendto,
          slRecycleBin, slStartMenu,
          slDesktopDirectory, slMyComputer,
          slNetwork, slNethood,
          slFonts, slTemplates);

property SpecialLocation: TdcBrowseDialogSpecialLocation;
```

Description
The SpecialLocation property used to specify the special shell folder to browse.

For example, if you would like to choose folders in "Start" menu, set SpecialLocation property to *slStartMenu*. If you wish to browse for printers and allow to choose only printers, set SpecialLocation property to *slPrinters*.

If you would like to allow to browse only within drives (and don't show any special shell directories), set SpecialLocation to *slMyComputer*.

See also
SpecialLocation property of the dcBrowseDialog component.

8.17.2.10 DlgStatusText

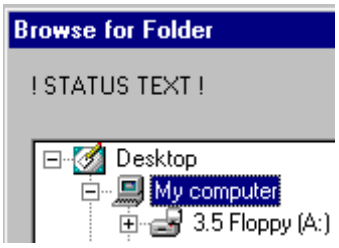
Applies to
dcFolderEdit component.

Declaration
```
property DlgStatusText: String;
```

Description
The DlgStatusText specifies the text in the status line of the "Browse for Folder" dialog box, which opens after clicking the button at the right side of edit control. The status line is the text that appears above the treeview with folders.

Snapshoot

© 1999-2001, UtilMind Solutions®



See also
StatusText property of the dcBrowseDialog component.

8.17.2.11 DlgTitle

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property DlgTitle: String;
```

Description
The DlgTitle property specifies the text for caption on the title bar of the dialog box, which opens after clicking the button at the right side of edit control.

See also
Title property of the dcBrowseDialog component.

8.17.3 Events

8.17.3.1 OnButtonClick

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property OnButtonClick: TNotifyEvent;
```

Description
The OnButtonClick event occurs when user clicks the button at the right side of edit control, to select the folder. The OnButtonClick for the dcFolderEdit component occurs before displaying the "Browse For Folder" dialog. If you would like to handle clicks on "Ok" and "Cancel" buttons of this dialog box — use OnDlgOk and OnDlgCancel events.

To make the custom button appear on the edit control — set Button.Visible to True, or to False if you'd like to hide the button.

See also
OnDlgOk and OnDlgCancel events;
dcBrowseDialog component.

8.17.3.2 OnDlgOk

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property OnDlgOk: TNotifyEvent;
```

© 1999-2001, UtilMind Solutions®

Description
The DlgOk event occurs when user selects the folder in the dialog box (which opens after clicking the button at the right side of edit control) and clicks the "Ok" button (confirms selection of the folder).

See also
OnDlgCancel and OnButtonClick events.

8.17.3.3 OnDlgCancel

Applies to
dcFolderEdit and dcFileEdit components.

Declaration
```
property OnDlgCancel: TNotifyEvent;
```

Description
The DlgCancel event occurs when user close the dialog box (which opens after clicking the button at the right side of edit control) without selection of the folder, or clicks the "Cancel" button.

See also
OnDlgOk and OnButtonClick events.

8.17.3.4 OnMouseEnter

Applies to
dcFolderEdit, dcFileEdit and dcFolderListView components.

Declaration
```
property OnMouseEnter: TNotifyEvent;
```

Description
The OnMouseEnter occurs when mouse pointer hovers the edit control.

See also
OnMouseLeave event.

8.17.3.5 OnMouseLeave

Applies to
dcFolderEdit, dcFileEdit and dcFolderListView components.

Declaration
```
property OnMouseLeave: TNotifyEvent;
```

Description
The OnMouseLeave occurs when mouse pointer leaves the edit control.

See also
OnMouseEnter event.

8.18 dcOpenDialog

8.18.1 TdcOpenDialog

Under construction, sorry for the inconvenience. We'll update the help soon. Please contact us: info@appcontrols.com

© 1999-2001, UtilMind Solutions®

8.18.2 Properties

8.18.2.1 DefaultExt

Applies to
dcOpenDialog and dcSaveDialog components.

Declaration
```
property DefaultExt: string;
```

Description
The DefaultExt property specifies a file extension that is appended automatically to the selected file name, unless the selected file name already includes a registered extension. If the user selects a file name with an extension that is unregistered, DefaultExt is appended to the unregistered extension.

Extensions longer than three characters are not supported. Do not include the period (.) that divides the file name and its extension.

See also
FileName property.

8.18.2.2 FileName

Applies to
dcFileInfo, dcFileCRC, dcFileTail, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration
```
property FileName: String;
```

Description
The FileName property specifies and returns the name and complete directory path of selected file. The value of FileName is the same as the first item in the Files property.

To make a file name appear by default in the dialog's edit box, assign a value to FileName in the Object Inspector or in program code. Programmatic changes to FileName have no effect while the dialog is active.

Use ExtractFilePath and ExtractFileName functions to sepearate the file path and filename.

See also
Files and DefaultExt properties.

8.18.2.3 Files

Applies to
dcOpenDialog and dcSaveDialog components.

Declaration
```
property Files: TStrings;
```

Description
The Files property is a list of selected items returned by file-selection dialog. Files is the TStrings instance that contains *each* selected file name with its full directory path.

To let users select multiple file names, set the *ofAllowMultiSelect* flag in Options.

© 1999-2001, UtilMind Solutions®

See also
FileName property.

8.18.2.4 Filter

Applies to
dcOpenDialog and dcSaveDialog components.

Declaration
```
property Filter: String;
```

Description
The Filter property determines the file masks (filters) available in the dialog. The file-selection dialog includes a drop-down list of file types under the edit box. When the user picks a file type from the list, only files of the selected type are displayed in the dialog.

To configure file masks at design time, click on the ellipsis marks (...) to the right of the Filter property in the Object Inspector. This opens the Filter editor. In the left column of the Filter editor, under Filter Name, type a brief description of each file type that will be available at runtime. In the right column, under Filter, type the file mask corresponding to each description. For example, the description "Text files" might appear to the left of the mask "*.TXT", and the description "Pascal source files" might appear to the left of the mask "*.PAS". Since the description appears in the drop-down list at runtime, it is often helpful to show the mask explicitly in the description (for example, "Text files (*.txt)").

To create file masks in program code, assign a value to the Filter property that consists of a description and a mask separated by a vertical bar (pipe) character. Do not include spaces around the vertical bar. For example,

```
dcOpenDialog1.Filter := 'Text files (*.txt)|*.TXT';
```

Multiple filters should be separated by vertical bars. For example,

```
dcOpenDialog1.Filter := 'Text files (*.txt)|*.TXT|Pascal files
(*.pas)|*.PAS';
```

To include multiple masks in a single filter, separate the masks with semicolons. This works both in the Object Inspector and in program code. For example,

```
dcOpenDialog1.Filter := 'Pascal files|*.PAS;*.DPK;*.DPR';
```

If no value is assigned to Filter, the dialog displays all file types.

See also
FilterIndex property.

8.18.2.5 FilterIndex

Applies to
dcOpenDialog and dcSaveDialog components.

Declaration
```
property FilterIndex: Integer;
```

Description
The FilterIndex property determines which of the file types in Filter is selected by default when the dialog opens. Set FilterIndex to 1 to choose the first file type in the list as the default, or set FilterIndex to 2 to choose the second file type as the default, and so forth. If the value of FilterIndex

© 1999-2001, UtilMind Solutions®

is out or range, the first file type listed in Filter is the default.

See also
Filter property.

8.18.2.6 InitialDir

Applies to
dcOpenDialog and dcSaveDialog components.

Declaration
```
property InitialDir: String;
```

Description
The InitialDir property determines the current (default) directory displayed in the file-selection dialog when it opens. For example, to point the dialog at the WINDOWS\SYSTEM directory, set the value of InitialDir to C:\WINDOWS\SYSTEM.

If no value is assigned to InitialDir, or if the specified directory does not exist, the initial directory is controlled by the global ForceCurrentDirectory variable. If ForceCurrentDirecotry is True, the dialog opens with the current working directory displayed. Otherwise, the dialog opens with either the current working directory or the My Documents directory, depending on the version of Windows.

See also
InitialDirSaver property.

8.18.2.7 InitialDirSaver

© 1999-2001, UtilMind Solutions®

8.18.2.8 Options

Applies to
dcOpenDialog and dcSaveDialog components.

Declaration

```
type
  TOpenOption = (ofReadOnly, ofOverwritePrompt, ofHideReadOnly,
ofNoChangeDir, ofShowHelp, ofNoValidate, ofAllowMultiSelect,
ofExtensionDifferent, ofPathMustExist, ofFileMustExist, ofCreatePrompt,
ofShareAware, ofNoReadOnlyReturn, ofNoTestFileCreate, ofNoNetworkButton,
ofNoLongNames, ofOldStyleDialog, ofNoDereferenceLinks,
ofEnableIncludeNotify, ofEnableSizing);
  TOpenOptions = set of TOpenOption;

property Options: TOpenOptions;
```

Description
The Options property determines the appearance and behaviour of the file-selection dialog. Use the Options property to customize the appearance and functionality of the dialog. The possible values of Options are

Value	Meaning
ofReadOnly	Selects the Open As Read Only check box by default when the dialog opens.
ofOverwritePrompt	Generates a warning message if the user tries to select a file name that is already in use, asking whether to overwrite the existing file. (Use with T[dc]SaveDialog and TSavePictureDialog.)
ofHideReadOnly	Removes the Open As Read Only check box from the dialog.
ofNoChangeDir	After the user clicks OK, resets the current directory to whatever it was before the file-selection dialog opened.
ofShowHelp	Displays a Help button in the dialog.
ofNoValidate	Disables checking for invalid characters in file names. Allows selection of file names with invalid characters.
ofAllowMultiSelect	Allows users to select more than one file in the dialog.
ofExtensionDifferent	This flag is turned on at runtime whenever the selected filename has an extension that differs from DefaultExt. If you use this flagin an application, remember to reset it.
ofPathMustExist	Generates an error message if the user tries to select a file name with a nonexistent directory path.
ofFileMustExist	Generates an error message if the user tries to select a nonexistent file. (only applies to Open dialogs).
ofCreatePrompt	Generates a warning message if the user tries to select a nonexistent file, asking whether to create a new file with the specified name.
ofShareAware	Ignores sharing errors and allows files to be selected even when sharing violations occur.
ofNoReadOnlyReturn	Generates an error message if the user tries to select a read-only file.
ofNoTestFileCreate	Disables checking for network file protection and inaccessibility of disk drives. Applies only when the user tries to save a file in a create-no-modify shared network directory.
ofNoNetworkButton	Removes the Network button (which opens a Map Network Drive dialog) from the file-selection dialog. Applies only if the ofOldStyleDialog flag is on.

© 1999-2001, UtilMind Solutions®

ofNoLongNames	Displays 8.3-character file names only. This option is only valid if Options also includes ofOldStyleDialog.
ofOldStyleDialog	Creates the older style of file-selection dialog.
ofNoDereferenceLinks	Disables dereferencing of Windows shortcuts. If the user selects a shortcut, assigns to FileName the path and file name of the shortcut itself (the .LNK file), rather than the file linked to the shortcut.
ofEnableIncludeNotify	Sends include notification messages to the dialog. (Windows 2000 and later.)
ofEnableSizing	Allows users to resize the dialog.

8.18.2.9 Title

Applies to
dcOpenDialog, dcSaveDialog and dcBrowseDialog components.

Declaration
```
property Title: String;
```

Description
The Title property specifies the text in the dialog's title bar. If no value is assigned to Title, the dialog has the title "Open".

8.18.3 Methods

8.18.3.1 Execute

Applies to
dcOpenDialog, dcSaveDialog and dcBrowseDialog components.

Declaration
```
function Execute: Boolean; // returns True when user clicks OK
```

Description
The Execute method opens the file-selection dialog, returning True when the user selects a file and clicks Open. If the user clicks Cancel, Execute returns False.

8.19 dcSaveDialog

8.19.1 TdcSaveDialog

Under construction, sorry for the inconvenience. We'll update the help soon. Please contact us: info@appcontrols.com

8.19.2 Properties

8.19.2.1 DefaultExt

© 1999-2001, UtilMind Solutions®

8.19.2.2 FileName

Applies to
dcFileInfo, dcFileCRC, dcFileTail, dcShellIcon, dcShellLink, dcShellProperties, dcOpenDialog, dcSaveDialog components.

Declaration
```
property FileName: String;
```

Description

8.19.2.3 Files

© 1999-2001, UtilMind Solutions®

8.19.2.4 Filter

© 1999-2001, UtilMind Solutions®

8.19.2.5 FilterIndex

© 1999-2001, UtilMind Solutions®

8.19.2.6 InitialDir

© 1999-2001, UtilMind Solutions®

8.19.2.7 InitialDirSaver

© 1999-2001, UtilMind Solutions®

8.19.2.8 Options

© 1999-2001, UtilMind Solutions®

8.19.2.9 Title

Applies to
dcSaveDialog, dcOpenDialog and dcBrowseDialog components.

Declaration
```
property Title: String;
```

Description

8.19.3 Methods

8.19.3.1 Execute

Applies to
dcSaveDialog, dcOpenDialog and dcBrowseDialog components.

Declaration
```
function Execute: Boolean; // returns True when user clicks OK
```

Description
The Execute method opens the file-selection dialog, returning True when the user selects a file and clicks Open. If the user clicks Cancel, Execute returns False.

8.20 dcBrowseDialog

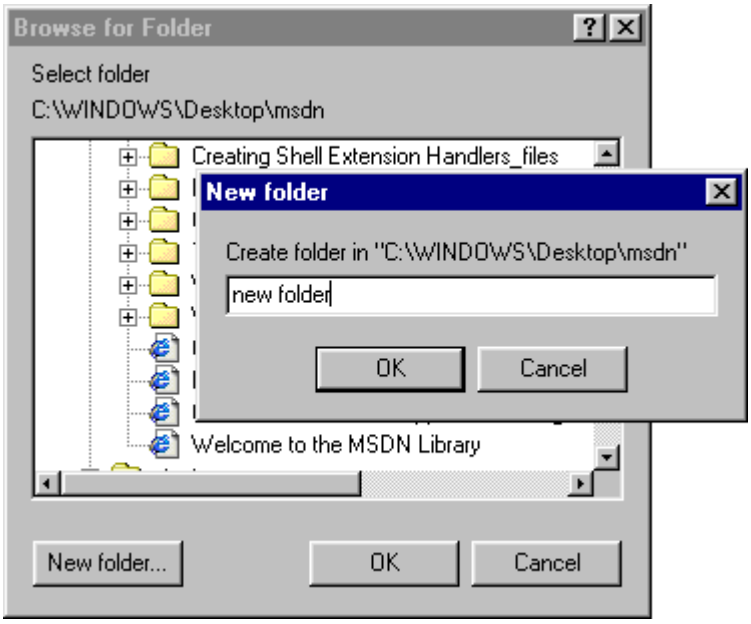
8.20.1 TdcBrowseDialog

Overview
The TdcBrowseDialog makes an "Browse For Folders" dialog box available to your application. The purpose of the dialog box is to let a user select local or netrwork folders.

new! In v3.2 has been added "Create New Folder" button to the dialog box. This button allows user to create new folder on the fly.

Snapshot

© 1999-2001, UtilMind Solutions®



How to use ?

You may select the starting folder at the Folder property and kind of location to browse at the BrowseFor and SpecialLocation properties. Choose browsing preferences in Options. If you would like to allow user to create new folder on browsing, set NewFolder.Visible property to True.

Use the Execute method to display this dialog box. The selected folder will returned in the Folder property.

To test an apperance of the dcBrowseDialog at design-time — drop the component on your form and click the Folder property in the Object Inspector.

See also

dcFolderEdit, dcOpenDialog and dcSaveDialog components.

8.20.2 Properties

8.20.2.1 BrowseFor

Applies to

dcBrowseDialog component.

Declaration

```
type
  TdcBrowseFor = (forAnything, forFolder, forComputer, forPrinter);

property BrowseFor: TdcBrowseFor;
```

Description

The Folder property specifies the starting location to browse and determines the selected folder or shell object after browsing (after the Execute method).

See also

SpecialLocation property.

© 1999-2001, UtilMind Solutions®

8.20.2.2 Folder

Applies to
dcDiskScanner, dcFolderMonitor and dcBrowseDialog components.

Declaration
```
property Folder: String;
```

Description
The Folder property specifies and determines the folder (or other shell object) selected in the folder-selection dialog.

Example
```
with dcBrowseDialog1 do
 begin
  Folder := Edit1.Text;
  if Execute then
    Edit1.Text := Folder;
 end;
```

8.20.2.3 Options

Applies to
dcBrowseDialog component.

Declaration
```
type
  TdcBrowseDialogOptions = set of (soShowEditBox, soShowFiles,
     soShowStatusText, soShowPathInStatus, soShowSysAncestors,
     soShowNetworkFiles, soSelectRootFolders);

property Options: TdcBrowseDialogOptions;
```

Description
The Options property specifies advanced options for folder-selection dialog. There are following possible options:

Value	Meaning
soShowEditBox	show edit box and allow to specify location typing the path in the edit box;
soShowFiles	show files and allow to select them;
soShowStatusText	show the status line (use the status line to show current path (*saShowPathInStatus*), or specify the status text in the OnSelected event handler);
soShowPathInStatus	show the path to selected folder in the status line;
soShowSysAncestors	show system shell folders (i.e: "Control Panel", "Printers" etc)
soShowNetworkFiles	show files in local area network;
soSelectRootFolders	allow to select root folders (i.e: C:\ or D:\).

See also
SpecialLocation and StatusText properties;
OnSelected event.

8.20.2.4 NewFolder

Applies to
dcBrowseDialog and dcFolderEdit components.

© 1999-2001, UtilMind Solutions®

Declaration

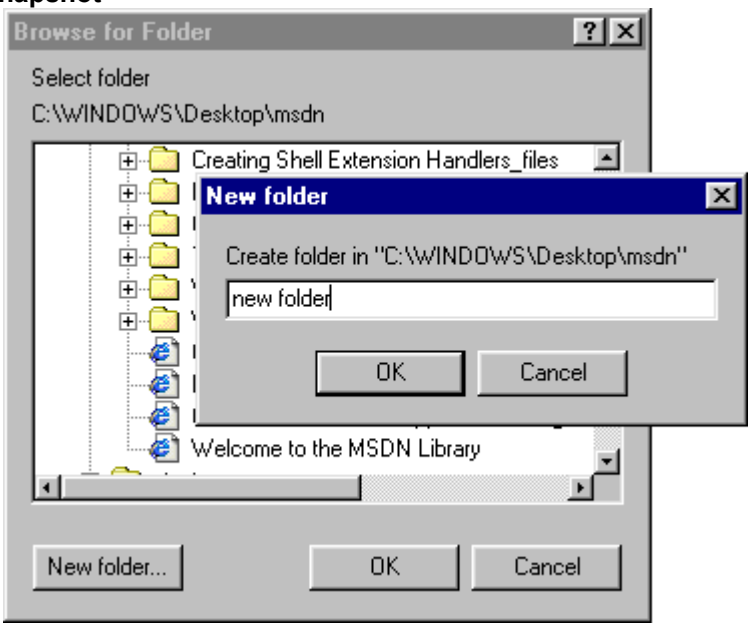
```
type
  TdcNewFolderBtn = class
  published
    property Caption: String;
    property Prompt: String;
    property Visible: Boolean;
  end;
```

Description

The NewFolder structure used to add the "New folder..." button to the browse-for-folder dialog and specify it's caption and prompt line for input box.

To add button to the dialog box — just set Visible property to True.

Snapshot



8.20.2.4.1 Caption

Applies to

dcBrowseDialog and dcFolderEdit components as subproperty of NewFolder structure.

Declaration

```
property Caption: String;
```

Description

The Caption property specifies the text that appears on the button used to create new folder. in the directory structure. Default text used for caption is "New folder...", but you can change this string to other (i.e: to translate it to another language).

See also

Prompt and Visible properties.

© 1999-2001, UtilMind Solutions®

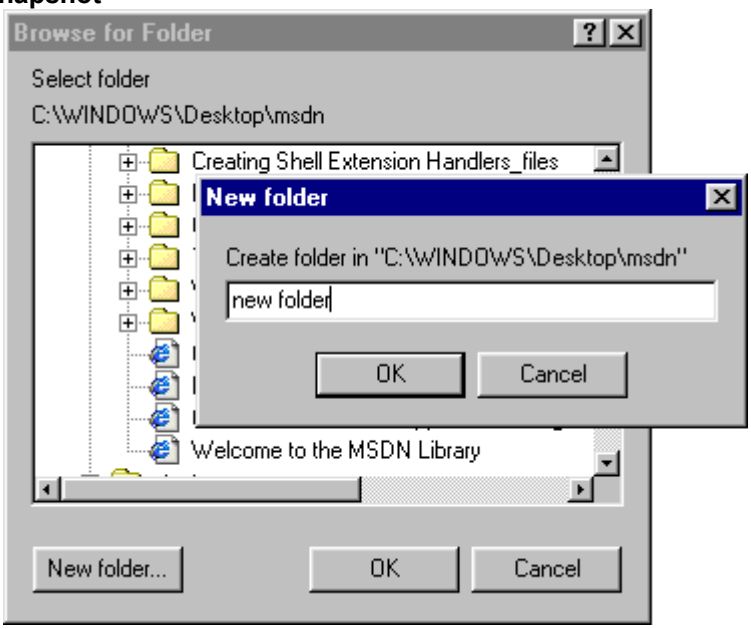
8.20.2.4.2 Prompt

Applies to
dcBrowseDialog and dcFolderEdit components as subproperty of NewFolder structure.

Declaration
```
property Prompt: String;
```

Description
The Prompt property specifies the text that appears in "prompt" line of the input box which pops up asking the name for new folder. Default text for prompt line is "Create folder in", but you can change this string to other (i.e: to translate it to another language).

Snapshot



See also
Caption and Visible properties.

8.20.2.4.3 Visible

Applies to
dcBrowseDialog and dcFolderEdit components as subproperty of NewFolder structure.

Declaration
```
property Visible: Boolean;
```

Description
The Visible property controls whether the "New folder" button should be visible in the dialog box. Set Visible to True to let user to create new folders on browsing.

See also
Caption and Prompt properties.

© 1999-2001, UtilMind Solutions®

8.20.2.5 Position

Applies to
dcBrowseDialog component.

Declaration
```
type
  TdcBrowseDialogPosition = (bpDefault, bpScreenCenter);

property Position: TdcBrowseDialogPosition;
```

Description
The Position property controls the placement of the folder-selection dialog when it appears on screen after calling the Execute method.

Value	Meaning
dpDefault	dialog box appears in default position;
dpScreenCenter	dialog box appears in center of screen (default).

You can also control the initial placement of the dialog box specifying the top-left coordinates in OnInitialized event handler.

See also
Execute method and OnInitialized event.

8.20.2.6 SpecialLocation

Applies to
dcBrowseDialog component.

Declaration
```
type
  TdcBrowseDialogSpecialLocation =
         (slNone, slDesktop,
          slPrograms, slControlPanel,
          slPrinters, slPersonal,
          slFavorites, slStartup,
          slRecent, slSendto,
          slRecycleBin, slStartMenu,
          slDesktopDirectory, slMyComputer,
          slNetwork, slNethood,
          slFonts, slTemplates);

property SpecialLocation: TdcBrowseDialogSpecialLocation;
```

Description
The SpecialLocation property used to specify the special shell folder to browse.

For example, if you would like to choose folders in "Start" menu, set SpecialLocation property to *slStartMenu*. If you wish to browse for printers and allow to choose only printers, set SpecialLocation property to *slPrinters*.

If you would like to allow to browse only within drives (and don't show any special shell directories), set SpecialLocation to *slMyComputer*.

See also

© 1999-2001, UtilMind Solutions®

BrowseFor property.

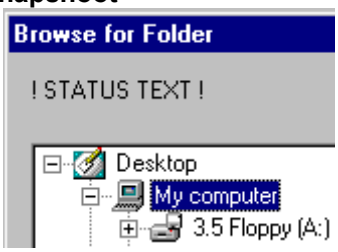
8.20.2.7 StatusText

Applies to
dcBrowseDialog component.

Declaration
```
property StatusText: String;
```

Description
The DlgStatusText specifies the text in the status line of the folder-selection dialog. The status line is the text that appears above the treeview with folders.

Snapshoot



See also
DlgStatusText property of the dcFolderEdit component.

8.20.2.8 Title

Applies to
dcBrowseDialog, dcOpenDialog and dcSaveDialog components.

Declaration
```
property Title: String;
```

Description
The DlgTitle property specifies the text for title bar of the folder-selection dialog. If not value assigned to Title, the dialog has the title "Browse for Folder".

See also
DlgTitle property of the dcFolderEdit component.

8.20.3 Methods

8.20.3.1 Execute

Applies to
dcBrowseDialog, dcOpenDialog and dcSaveDialog components.

Declaration
```
function Execute: Boolean; // returns True when user clicks OK
```

Description
The Execute method opens the "Browse for Folder" dialog, to let user to select the shell object from the list. The Execute returns True when user selects the folder and clicks "Ok" or, if user if user clicks "Cancel" it returns False.

© 1999-2001, UtilMind Solutions®

8.20.4 Events

8.20.4.1 OnChanged

Applies to
dcBrowseDialog component.

Declaration
```
type
  TdcBrowseDialogChanged = procedure(Sender: TObject; Folder: String) of
object;

property OnChanged: TdcBrowseDialogChanged;
```

Description
The OnChanged event occurs when user selected the directory and *confirmed* selection. Selected directory returned in *Folder* parameter.

See also
OnSelected event.

8.20.4.2 OnInitialized

Applies to
dcBrowseDialog component.

Declaration
```
type
  TdcBrowseDialogInitialized = procedure(Sender: TObject; Wnd: hWnd; var
posLeft, posTop: Integer) of object;

property OnInitialized: TdcBrowseDialogInitialized;
```

Description
The OnInitialized event occurs before appearing the dialog on screen and intended to
- get access to the dialog handle (*Wnd* parameter), to add some custom controls to the dialog, for instance;
- set the placement of the dialog box on screen (specifying *posLeft* and *posTop* parameters, left and top position of the dialog box on screen). Alternatively, to specify initial placement of the dialog box you can use Position property.

See also
OnChanged and OnSelected events;
Position property

8.20.4.3 OnSelected

Applies to
dcBrowseDialog component.

Declaration
```
type
  TdcBrowseDialogSelected = procedure(Sender: TObject; Folder: String;
var OKEnabled: ShortInt; var StatusText: String) of object;

property OnSelected: TdcBrowseDialogSelected;
```

© 1999-2001, UtilMind Solutions®

Description
The OnSelected event occurs when user selects the folder in the dialog box.

Parameters

Folder	returns the path to selected folder;
OKEnabled	used to enable or disable "OK" button of dialog box (to allow or disallow user to confirm selection);
StatusText	used to retrieve or specify the status text (if *soShowStatusText* option is True).

See also
OnChanged event and Options property.

8.21 dcSystemImageList

8.21.1 TdcSystemImageList

Overview
The dcSystemImageList gives access to the system image list. That's the image list which all icons (32x32 or 16x16) for various file types that Explorer uses. All these icons represents the files within the Windows shell.

How to use ?
Just point the *Images* property of any component, which uses the ImageList's to display images, to the dcSystemImageList. The icons will be taken from the list of system images.

For example, if you have the ListView which contains the list of files, you can use the dcSystemImageList as source for file icons.

To retrieve *small* icons (16x16) — set the ImageSize property to `is16x16`. To retrieve *large* icons (32x32) — set the ImageSize property to `is32x32`.

Example

You can use the dcSystemImageList together with the dcDiskScanner or dcMultiDiskScanner components. For example, you need to find some certain files and show them in the ListView + use the system image list as source for images that represents the icons for found files.

Then point the SmallImages property of the ListView to the dcSystemImageList component with ImageSize equal to is16x16 (and/or LargeImages property to the dcSystemImageList where the ImageSize = is32x32). Hook the OnFileFound event of the dc[Multi]DiskScanner component, and use *SysImageIndex* parameter to specify the image index for the list items.

```
procedure TForm1.dcDiskScanner1FileFound(Sender: TObject; FileName,
   FileType: String; FileSize: Extended; FileTime: TDateTime;
   FileAttributes: TdcScanAttributes; LargeIcon, SmallIcon: TIcon;
   SysImageIndex: Integer; TotalFiles: Integer; TotalSize: Extended);
var
  ListItem: TListItem;
begin
  ListItem := ListViews.Items.Add;
  ListItem.Caption := FileName;
  ListItem.ImageIndex := SysImageIndex;
end;
```

Notes
1. You can use/retrieve the icons of this list, but can not modify (add/delete) them.
2. You don't need to use the dcSystemImageList with dcFolderListView, because the dcFolderListView automatically uses the system images internally.

© 1999-2001, UtilMind Solutions®

See also
dcDiskScanner and dcMultiDiskScanner component.

8.21.2 Properties

8.21.2.1 ImageSize

Applies to
dcSystemImageList component.

Declaration
```
type
  TdcSystemImageSize = (is16x16, is32x32);

property ImageSize: TdcSystemImageSize;
```

Description
The ImageSize property specifies the size of icons in the system image list. The size of the icons can be 16x16 or 32x32.

If you are using dcSystemImageList as source for images in the ListView, then point SmallImages property of the ListView to the dcSystemImageList with ImageSize equal to `is16x16`, and LargeImages to the dcSystemImageList where ImageSize = `is32x32`.

8.22 dcIcon2Bitmap

8.22.1 TdcIcon2Bitmap

Overview
The dcIcon2Bitmap is utility component that will help you to convert icon images (TIcon) to bitmaps (TBitmap). In example, it may be very useful if you would like to store bitmaps instead of icons, returned by various components of DiskControls pack, in the TImageList collection.

How to use ?
Assign the icon image to the Icon property and read the converted bitmap image from Bitmap property.

8.22.2 Properties

8.22.2.1 Bitmap

Applies to
dcIcon2Bitmap component.

Declaration
```
property Bitmap: TBitmap; // Read only!
```

Description
The Bitmap property determines the bitmap image, converted from icon, specified in the Icon property.

See also
Icon property.

© 1999-2001, UtilMind Solutions®

8.22.2.2 Icon

Applies to
dcIcon2Bitmap component.

Declaration
```
property Icon: TIcon;
```

Description
The Icon property specifies the icon image that you would like to convert to bitmap. After specifying the icon, you can read the converted bitmap image from Bitmap property.

See also
Bitmap property.

© 1999-2001, UtilMind Solutions®

Index

- A -

AddPath 48

- C -

Components Overview 10
Contact Information 21

- D -

dcBrowseDialog 168
dcDiskInfo 70
dcDiskScanner 21
dcFileAssociation 58
dcFileCRC 80
dcFileEdit 139
dcFileInfo 76
dcFileOperations 53
dcFileTail 82
dcFolderEdit 147
dcFolderListView 110
dcFolderMonitor 50
dcIcon2Bitmap 177
dcMultiDiskScanner 34
dcOpenDialog 156
  Execute 161
dcSaveDialog 161
  Execute 168
dcShellIcon 94
dcShellLink 97
dcShellProperties 102
dcSystemImageList 176
dcVersionInfo 86
dcVirtualDrives 66
DiskControls pack 10

- E -

Example 30
  dcDiskInfo 75
  dcFileAssociation 65
  dcMultiDiskScanner 47
  dcShellLink 102
  ExcludeList 47
  IncludeList 47
  OnFileFound 33
  TdcFileOperations 58

- H -

How to install new file association? 65

- I -

Installation Instructions 16
Introduction 10

- L -

License Agreement 19

- O -

Overview 10

- R -

Registration Information 18

- S -

Search rules designer 47

- T -

TdcBrowseDialog 168
  BrowseFor 169
  Execute 174
  Folder 170
  NewFolder 170
  OnChanged 175
  OnInitialized 175
  OnSelected 175
  Options 170
  Position 173
  SpecialLocation 173
  StatusText 174

© 1999-2001, UtilMind Solutions®

TdcBrowseDialog 168
Title 174
TdcDiskInfo 70
AvailableDrives 70
BytesPerSector 71
Disk 71
DiskFree 71
DiskSize 72
DriveType 72
FileSystem 72
FreeClusters 73
LargeIcon 73
SectorsPerCluster 73
SerialNumber 73
SerialNumberStr 74
SmallIcon 74
SysImageIndex 74
TotalClusters 75
VolumeLabel 75
TdcDiskScanner 21
Busy 22
Execute 29
Files 23
FindAllFolders 22
Folder 23
IncludeHiddenSubfolders 23
IncludeSubfolders 24
Matches 24
OnFileFound 30
OnScanDone 32
OnScanFolder 32
SearchAttributes 24
SearchMask 26
SearchSize 26
SearchTime 26
Stop 29
Suspended 27
ThreadPriority 27
UseIcons 28
WaitThread 28
TdcEditButton 149
Cursor 150
Flat 150
Glyph 150
Hint 152
NumGlyphs 151
Visible 152
Width 152
TdcFileAssociation 58
AccessMode 60
ExecutableFile 61
ExtDescription 61
EXTENSION 61
FileDescription 62
IconIndex 62
IconLocation 62
InstallFileType 64
LargeIcon 63
MIMEType 63
ParamString 63
SmallIcon 64
UninstallFileType 65
TdcFileAssociationDefCommandName 60
TdcFileCRC 80
CRC 81
FileName 81
Method 81
TdcFileEdit 139
Align 139
Alignment 140
AutoSelect 141
Button 141
Color 144
ColorDisabled 144
CursorBorder 144
DlgDefaultExt 145
DlgFilter 145
DlgOptions 145
DlgTitle 145
DlgType 146
OnButtonClick 146
OnDlgCancel 147
OnDlgOk 146
OnMouseEnter 147
OnMouseLeave 147
TdcFileInfo 76
DisplayName 76
FileAttributes 77
FILENAME 77
FileSize 78
FileType 78
LargeIcon 78
SmallIcon 79
SysImageIndex 79
TimeCreated 79
TimeLastAccessed 80
TimeLastModified 80
TdcFileOperations 53

© 1999-2001, UtilMind Solutions®

TdcFileOperations 53
Copy 56
Delete 56
DestFolder 53
Example 58
Execute 56
FileList 54
Move 57
OnError 58
Operation 54
Options 55
ProgressTitle 56
Rename 57
TdcFileTail 82
CheckChanges 85
FileName 82
FileSize 83
LastLines 83
LastLinesCount 83
MonitorType 83
OnFileAppended 85
OnFileChanged 86
ThreadPriority 84
TdcFolderEdit 147
Align 148
Alignment 149
AutoSelect 149
Button 149
Color 152
ColorDisabled 153
CursorBorder 153
DlgOptions 153
DlgSpecialLocation 154
DlgStatusText 154
DlgTitle 155
OnButtonClick 155
OnDlgCancel 156
OnDlgOk 155
OnMouseEnter 156
OnMouseLeave 156
TdcFolderListView 110
Back 128
Columns 111
CreateFolder 128
CreateShortcut 129
Cursors 115
DeleteSelectedFiles 129
FileAttributes 116
FileAttributesByListItem 130
FileMask 117
FileNameByListItem 130
FileSizeByListItem 131
FileTimeByListItem 131
FileTypeByListItem 131
Folder 118
FolderMonitor 118
GetColumnAt 132
GetItemAt 132
HeaderMenu 119
HeaderStyle 120
IsBackPossible 133
IsScrollBarsVisible 133
KBStr 121
ListItemByFileName 133
OnBeginUpdate 135
OnDrawHeaderSection 136
OnEndUpdate 136
OnFolderChanged 137
OnMouseEnter 138
OnMouseLeave 138
OnSettingsChanged 138
OnToolTip 139
Options 121
Refresh 134
RegistrySaver 122
SelectAll 135
ShowAllFolders 123
ShowFileExtensions 124
ShowFolderProperties 134
ShowHeaderMenu 124
ShowItemProperties 135
ShowSortMark 124
ShowToolTips 125
SortColumn 125
SortDirection 125
SortMarkAlign 126
SortMarkKind 127
ToolTipOptions 127
UnselectAll 135
TdcFolderMonitor 50
Active 50
Filter 51
Folder 51
OnChange 52
ThreadPriority 52
TdcIcon2Bitmap 177
Bitmap 177
Icon 178

© 1999-2001, UtilMind Solutions®

TdcListColumn 111
Alignment 112
Caption 112
Visible 112
Width 113
TdcListColumns 111
Attributes 111
FileType 113
Modified 113
Name 114
Size 114
TdcListViewCursors 115
Border 115
Divider 115
Header 116
ScrollHorz 116
ScrollVert 116
TdcListViewHeaderMenu 119
AscendingItem 119
DescendingItem 120
PopupMenu 120
TdcListViewRegistrySaver 122
Enabled 122
RegKey 123
RegLocation 123
TdcMultiDiskScanner 34
Busy 35
ExcludeList 35
Execute 42
Files 35
IncludeList 36
Matches 37
OnExcludingBegin 42
OnExcludingEnd 43
OnFileFound 43
OnFolderNotExist 45
OnScanDone 45
OnScanFolder 46
SearchAttributes 38
SearchSize 39
SearchTime 39
Stop 42
Suspended 40
ThreadPriority 40
UseIcons 41
WaitThread 41
TdcNewFolderBtn 170
Caption 171
Prompt 172
Visible 172
TdcOpenDialog 156
DefaultExt 157
FileName 157
Files 157
Filter 158
FilterIndex 158
InitialDir 159
InitialDirSaver 159
Options 160
Title 161
TdcSaveDialog 161
DefaultExt 161
FileName 162
Files 162
Filter 163
FilterIndex 164
InitialDir 165
InitialDirSaver 166
Options 167
Title 168
TdcShellIcon 94
DisplayName 95
FILENAME 95
FileType 95
LargeIcon 96
Options 96
SmallIcon 96
SysImageIndex 97
TdcShellLink 97
FILENAME 98
HotKey 98
IconIndex 99
IconLocation 99
LargeIcon 99
LinkTarget 99
LoadFromFile 101
Parameters 100
RunAs 100
SaveToFile 102
ShortDescription 100
SmallIcon 101
WorkingDirectory 101
TdcShellProperties 102
FileName 103
MenuAlignment 104
MenuOptions 104
OnAfterMenuItemClick 108
OnBeforeMenuItemClick 108

© 1999-2001, UtilMind Solutions®

TdcShellProperties 102
OnDelete 109
OnRename 109
ShowContextMenu 105
ShowContextMenuByFile 106
ShowProperties 107
ShowPropertiesByFile 108
TdcSystemImageList 176
TdcVersionInfo 86
Comments 87
CompanyName 87
FileDescription 88
FileFlags 88
FileFlagsMask 88
FileMajorVersion 88
FileMinorVersion 89
FILENAME 89
FileVerBuild 90
FileVerMajor 90
FileVerMinor 90
FileVerRelease 91
FileVersion 91
FileVersionFloat 91
InternalName 92
LanguageName 92
LegalCopyright 92
LegalTrademark 92
OriginalFilename 92
ProductMajorVersion 93
ProductMinorVersion 93
ProductName 93
ProductVersion 93
ProductVersionFloat 94
TranslationValue 94
TdcVirtualDrives 66
AddDrive 68
DeleteDrive 69
Drive 67
GetTargetByDrive 69
Target 67
UnusedDrives 67
VirtualDrives 68
TDiskScanList 48
TDiskScannerPath 49

- V -

Version History 13

© 1999-2001, UtilMind Solutions®