DF SDK User Manual

Table of Contents

1. Overview

1.1. What is DF SDK?

DF SDK™ version 1.0.2.2

Copyright © 2002-2008 by AgenSoft. All rights reserved.

Contacting Agensoft

DF SDK is a software development kit which provides an effective approach for creating software patches and updates, and integrating of the patch applying and/or building process directly into you own software product(s). Unlike other available products, DF SDK does not simply create incremental updates and re-package files, but analyzes each file at the byte level and builds the difference for updating the target file on the end-user's system.

DF SDK does not deal with any specific data structures, it operates with files as a binary data, and that why it is designed to work well on files of any type including executables, libraries, data files and others.
It can be used to build a difference for two binary files with the following reconstruction of the new version file(s) using an old file and the output difference file (df-file).
DF SDK is available as a standard dynamic link library (DLL), and can be used almost with any Windows development language. The DF SDK API is designed to be simple enough to integrate it into your existing products/solutions, and to provide a high-grade of performance and flexibility. DF SDK comes with sample programs, and includes examples which can simplify patch creation and applying process.

Keeping your customers on the most current version optimizes the performance of your product and significantly reduces your end-user support costs.

DF SDK can be used as an efficient instrument in solving of the following problems:

- Software updating/patching;
- Differential backup;
- Version control.

DF SDK package consists of two main components, which are:

- **LIBDF library (REDISTRIBUTABLE PACKAGE)**, functionality of this library provides comparing of two binary files and creating a difference-file (df-file), and reconstructing a new version file using an old file and created difference-file;
- **LIBDFP library (REDISTRIBUTABLE PACKAGE)**, this library contain all functionality needed to reconstruct a new version file using an old file and created df-file (only functions listed in the "Difference Applying" category are included so that the size of this library is significantly smaller then that of *LIBDF* library).

NOTE: Only libdf.dll and/or libdfp.dll modules may be freely redistributed, e.g. within your own software product(s). You may not redistribute any other documents, code examples, etc. included into the full-version distribution package (obtained after ordering)!

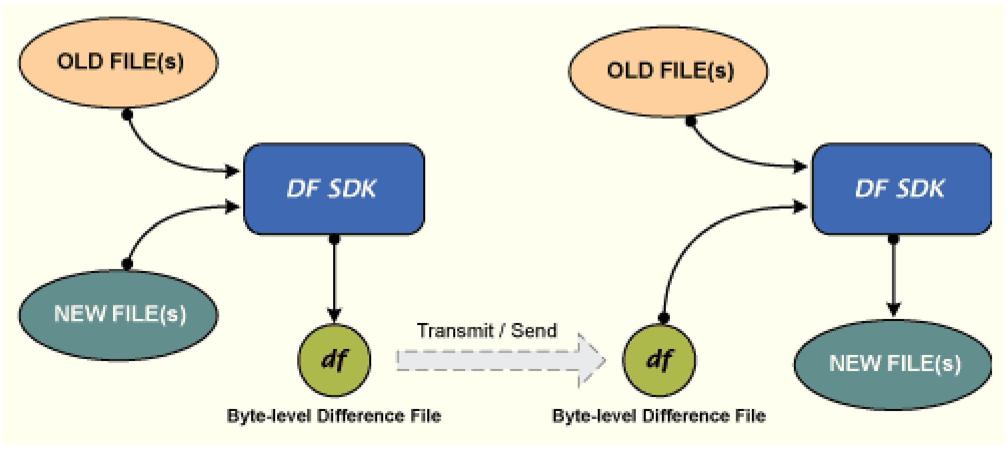
NOTE: Source code is not included into the the full-version distribution package (obtained after standard ordering)! It can be obtained only upon request for additional payment.

Key features of DF SDK v1.0.2.2 :

- Comparing files with size up to 2^63 Bytes (~8589934592 Gb).
- High speed of file comparing and df-file reconstruction
- Improved quality of file-comparing, resulting in smaller output df-files
- bzip2-compression is used and as a result the output difference-file (df-file) is smaller by size
- Flexible control under ratio "time/result" with the help of different comparing methods selection
- Setting utilized resources constraints
- Support for all Windows versions in use today -- Windows 95, 98, 2000, 2003, XP, Me, and NT 4.0 SP6.

DF SDK is developed to effectively accomplish a "binary files updating/patching" task.
If you are familiar with such standard utilities of the UNIX system as *diff* and *patch* then we could say that functionality of DF SDK can accomplish the same task though there is a significant difference: DF SDK effectively operates not only with text but with all binary files.

The idea behind DF SDK is simple: when modified file(s) must be transmitted, send only the changes (byte level differences) stored in one reliable self-installing update module rather than the entire software installation package. DF SDK lossless byte-level compression is not content dependent, so it may be used whenever data is changed at one location and must be efficiently updated and/or archived at another.



NOTE: DF SDK does not deal with any specific data structures, it operates with files as a binary data. Databases or files of other formats can be updated only as binary files (**warning:** database update can be implemented only if it is not changed on the end-user's machine at the moment).

System requirements :

- Microsoft Windows 95, 98, NT4, 2000, ME, XP, 2003 or above
- Processor 486 and above, 16 Mb RAM
- Microsoft Internet Explorer 5.0 or later - *only to view this chm help-file*
- Display resolution 800x600 hi-color (16 bits) - *only to view chm help-file*

1.2. About Software Patching

Preface...

Onrush of the modern software development, and also wide Internet availability, have made usual the frequent release of new versions of software programs. Prompt bugs / mistakes correction and feature adding are those major factors which compel manufacturers to modify their software products on a frequent basis. And if the overall size of the distribution package of a software product is quite sizeable, then the necessity to download the full new version distribution package can become an expensive and tiresome procedure for end-users.

Most often, the size of the brought changes from the version to version is significantly less than the size of the full distribution package, so many developers could thought that it would be really a good idea to have an opportunity to implement software updating by delivering to end-users only those, presumably small changes by size which distinguish the new version from the version already installed on the end-user's machine. Well, an attempt to reduce the update data size as much as possible is praiseworthy. However, how to make it really effective? In case of the text data, everything is clear. There are a lot of excellent algorithms for text files comparing. And even more the text data can be compressed effectively. But if you need to build an update package for executable files (exe, dll) then you will find out that these algorithms are completely powerless to make something worthwhile. Effective comparison of executable files or any other binary files requires absolutely other approaches. We'll use "binary file" phrase to define files which structure is obviously unknown. Such files are considered as an arbitrary octet-byte stream.

So, when an update must be provided for the end-users of an application, a developer has several choices:

Re-Packaging

Create a complete installation package for the application and make it available for download, or issue new media that contains the updated software. This is the simplest, and the most common approach used. But for large packages, this can appear to be inappropriate for some customers who need the update immediately, so that you require them to download a large file (or even a collection of files), in spite of the fact that only a relatively small number of changes have been made to specific parts of the application files. You may also send new media to your customers, incurring additional costs related to delivering the updated software.

Incremental Updates

Although many companies refer to this method as a "patch", an incremental update of the package involves redistributing only those files which have changed since the previous release. While this approach is an improvement over re-packaging of the application, it can introduce several problems of its own. This method requires that a developer creates a new distribution package which consists of only modified files. If a new version consists of changes made to several large files, or contains a significant number of modified files, then benefits of incremental updating are reduced significantly. And moreover, incremental updating has the potential for introducing problems where critical system libraries or shared components are updated. The developer must make sure that only the appropriate versions of the target files are being overwritten when the update is applied. If the update takes no notice that a file which needs to be overwritten is not of the correct version, it can result in application fail unexpectedly or even general system instability.

Patching

Creating a patch is the process of two files comparing, the original and an updated file, and the following returning only of the bytes that have been changed. The resulting patch file consists only of the changes from within each individual file with the help of byte-level differencing technology used by our patching engine, resulting in a significantly smaller update size. When multiple files need to be updated, the collection of individual patches are then collected in a single file called a patch package.

This patch package is then applied on the end-user's machine and the files are updated to the current version. The patch application process also ensures that the correct files are updated and that the updates are being applied correctly.

Clearly, the most advantageous choice for both the developer and the users is to pack the update as a patch, rather than re-deploying of the entire distribution package or creating an installation package for incremental updating. This approach minimizes the amount of time the developer spends creating and deploying the update, and significantly reduces the size of the update and the amount of downtime that the customer experiences while the application is being updated. And as a result, using patching you can to significantly reduce end-user support costs.

DF SDK

DF SDK is developed to effectively accomplish a *"binary files updating/patching"* task.

If you are familiar with such standard utilities of the UNIX system as *diff* and *patch* then we could say that functionality of DF SDK can accomplish the same task though there is a significant difference: DF SDK effectively operates not only with text but with all binary files.

1.3. What's new in this version?

DF SDK v1.0.2.2 Version History

[Legend]

[+ Added feature]

[x Improved/changed feature]

[– Bug fixed]

===

Version 1.0.2.2 (25 Dec 2006)

[–] df-file applying: critical bug fixed - function dfOpenDfFile could return EDF_DF_DAMAGED for correct df-files in some cases.

===

Version 1.0.2.1 (11 Dec 2006)

[–] df-file building: critical comparing algorithm bug fixed, which could in some cases lead to significant increase of algorithm work time and worse result.

[x] Miscellaneous help system corrections

===

Version 1.0.2 (19 Oct 2006)

[–] df-file applying: significant bug fixed

[x] Miscellaneous help system corrections

===

Version 1.0.1 (10 Oct 2006)

First Public Release

===

1.4. Binary files comparing

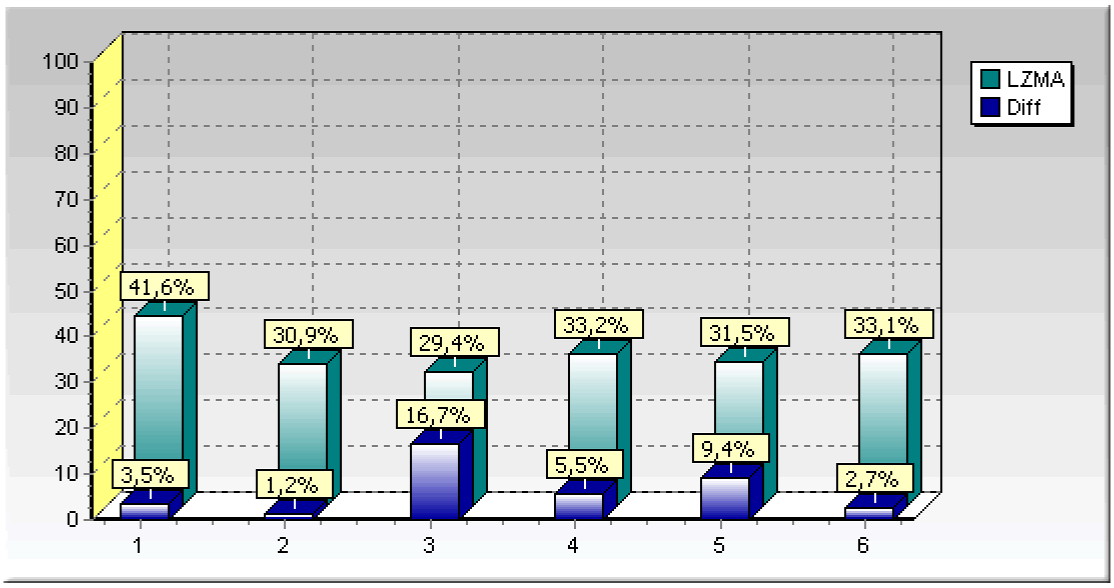
Preface...
Construction of a small-size patch is closely related to the number of changes of the new version versus an old one, and also to the efficiency of the methods, capable to determine these changes and present them in a compact form. Well-known algorithms of text data comparing can't help in comparing of the binary data when the structure of files is completely unknown in advance, and the nature of these changes is poorly predicted. Attempt to apply LCS-search algorithms (Largest Common Subsequence) also have shown their incompetence. Comparison of binary files requires a special approach which is successfully implemented in DF SDK. The differencing algorithm used in DF SDK provides creating of the smallest difference-files available in the industry.

The basis of the algorithm is the ability to find concurrence in compared files at a level of octet-byte subsequences. Such byte-oriented nature of the algorithm allows to make the efficiency of its work independent of a file format. The size of a resulting difference file, in view of time spent for its construction is assumed as the main criterion of patch-building algorithm efficiency.

Key features of new algorithm:

- Significant improvement of algorithm quality parameters (generated difference files are smaller, work speed is greater);
- Special optimization providing considerable raise of comparison quality for executable modules (exe, dll);
- Comparing files with size up to 2^63 bytes;
- Flexible control under ratio "time/result" with the help of different comparing methods selection;
- Setting utilized resources constraints (memory size, process priority);
- Caching of comparing results. Storage of comparing results in intermediate files allows to significantly reduce the time of patch building.

The table below contains the examples (implemented for well-known software products main exe-files), illustrating the efficiency of df-files in comparison with LZMA-compression. Percentage values show the ratio of the LZMA-compressed file and df-file sizes to the size of a new file.



File	**New file size,** bytes	**LZMA-compression**		**DF-file size**	
		bytes	%	bytes	%
RAR.exe [3.40 beta3] – [3.42]	**297 472**	123 733	41.6	**10 291**	**3.5**
Winamp.exe [5.04] – [5.05]	**980 480**	303 275	30.9	**12 240**	**1.2**
TheBat.exe [3.4.0.933] – [3.5.0.1013]	**8 955 464**	2 632 520	29.4	**1 493 827**	**16.7**
HelpMan.exe [3.4] – [3.5] (Help&Manual)	**5 139 456**	1 704 393	32.2	**284 577**	**5.5**
SPECCTRA.exe [10.2] – [15.2] (SPECCTRA ShapeBased Automation Software	**13 307 978**	4 190 654	31.5	**1 245 494**	**9.4**
Fireworks.exe [7.0.0.288] – [7.0.2.295]	**14 696 448**	4 861 690	33.1	**400 728**	**2.7**

1.5. Acknowledgments

DF SDK v1.0.2.2 includes the following sources, which are used with the permission of their authors for redistribution.

- -

bsdiff
Description: bsdiff and bspatch are tools for building and applying patches to binary files
Author: Colin Percival, Computing Lab, Oxford University
Web: http://www.daemonology.net/bsdiff/

- -

bzip2 and libbzip2
Description: Freely available, patent free, high-quality data compressor
Author: Julian R. Seward
Web: http://sources.redhat.com/bzip2/

- -

2. Library LIBDF

2.1. Introduction

Library LIBDF is designed to solve a problem which can be briefly described as "binary files updating/patching".

If you are familiar with such standard utilities of the *UNIX* system as *diff* and *patch* then we could say that functionality of DF SDK can accomplish the same task though there is a significant difference: DF SDK effectively operates not only with text but with all binary files.

However if diff or patch sound nothing to you, let's describe what can be implemented using LIBDF by the following example:

> Assume that you have some file which is being changed in the course of time. It can be an executable file of your program, or any other file independent on its format and content (except archive formats which will be described later). Let's assume that you have two version of one of those files which we can call *old* file and *new* file. Using functions of LIBDF library we can implement comparing of a new versus old file, find differences between them and save the data describing these differences in a special difference file - *df-file*. Later on using other functions of LIBDF library we can reconstruct a new file having only an old file and df-file.

In case of a slight difference between an old file and a new one (i.e. there are few changes brought into a new file in comparison with an old one) and the size of the result df-file is much smaller then that of a new file then **LIBDF can appear to be an effective tool to solve the following problems:**

- **Software updating/patching** to ensure your end-users always have the latest version of your software product;
- **Differential backup** to store your backups in a reliable yet space-saving way;
- **Version control** for reliable yet efficient control under versions of your files (of any type).

2.2. How it works

Library LIBDF consists of Functions and data Structures which can be divided into 2 main categories.

- First category ("Comparing and Diff") contains functions and corresponding structures used to compare files and build difference files (df-files).
- Second category ("Difference Applying") contains functions used for df-files applying (i.e. reconstruction of a new (revised) file using an old file and df-file). LIBDFP library contains functions only from this category and has the smaller size comparing to LIBDF library.
- Shared Structures, used by functions from both categories above.

It is necessary to mark that almost all function of LIBDF library return an value of the type int, which encodes the output of function execution. "0" value (EDF_SUCCESS) means that function is executed successfully. Otherwise the returned value is a specific error code. The full list of error codes is provided in the "Error codes" section of this manual.

Comparing and Diff

Functions

dfOpenCmpByNames
dfOpenCmpByHandles
dfCompare
dsTestCmpResult
dfBuildN
dfBuildH
dfCloseCmp

Structures

TDfCmp
TDfCmpOptions
TDfCmpResult
TDfBldOptions

Difference Applying

Functions

dfOpenDfFile
dfCloseDfFile
dfGetDfInfo
dfApplyN
dfApplyH
dfTouch
dfGetExtraDataSize
dfGetExtraData

Shared Structures

TDfInteract
TDfChangePhase
TDfProgress

2.3. Getting Started

2.3.1. *Comparing Files*

Assume that you have 2 files, which you would like to compare and in case they are different to build a difference file (df-file).
To start th comparing process it is necessary to create the so called `comparing context`. Comparing context is a special data structure `TDfCmp`, which contains all necessary information concerning the compared files, and also includes additional fields. (See description of TDfCmp structure for details).

Creation of the comparing context can be implemented using one of the following functions:

```
int dfOpenCmpByNames ( _TCHAR const * szOldFileName, _TCHAR const *
szNewFileName, int bExtractPEVersions, PTDfCmp * ppCmp );
```

or

```
int dfOpenCmpByHandles ( FD_T hOldFile, FD_T hNewFile, PTDfCmp * ppCmp );
```

In the first function compared files are set as names of files and in the second by files descriptors, which are probably created to the moment of comparing start.
Pointer on the successfully created *comparing context* is saved in the variable on which a `ppCmp` parameter points.
After the comparing context is successfully created we are ready to start comparing our files.

To compare files use function `dfCompare`:

```
int dfCompare ( TDfCmp * pCmp, TDfCmpOptions * pCmpOpt, TDfInteract *
pInteract, TDfCmpResult * pCmpRes );
```

Besides the comparing context let's take other 3 parameters:
`pCmpOpt` – pointer on the structure `TDfCmpOptions`, which contains parameters of comparing.
Each from compared files can be divided into to main parts in the view of df-file building: file name, file attributes, last modification date/time and file content. Comparing of files is implemented by comparing of all 4 parts. Files are considered different from each other if at least one part differs. In case of file content difference there is an option used to set necessity of identical parts search for the following df-file building. (See structure TDfCmpOptions for detailed description of comparing options).

Process of content comparing and search for identical parts can be a long operation (especially for big files).
`pInteract` – parameter which can be used to provide the operation progress representation (for instance as a progress indicator)
Fields of structure `TDfInteract` on which `pInteract` points used to set addresses of necessary callback-functions (see `TDfInteract` structure description for details).

`pCmpRes` – points on `TDfCmpResult` structure where the result of file comparing (per each part) will be saved in case of successful dfCompare function termination.

The following function can be used for generalization of the comparing results to the simple conclusion: files are equal or not.

```
int dfTestCmpResult ( TDfCmpResult * pCmpRes );
```

By analysing fields of the structure `TDfCmpResult` using or own tools or by using the function `dfTestCmpResult` you can made a conclusion whether compared files are different or not and if they are then what particular parts among 4 available are different and finally you make a decision if it necessary to build a df-file. Note that difference file can be built even if files are identical. This feature can useful in some special applications.

If a decision to build df-file is made then the following function is called:

```
int dfBuildN ( _TCHAR const * szDfFile, TDfCmp * pCmp, TDfBldOptions *
pBldOpt, TDfInteract * pInteract );
```

or

```
int dfBuildH ( FD_T hDfFile, TDfCmp * pCmp, TDfBldOptions * pBldOpt,
TDfInteract * pInteract );
```

These functions differ from each other only in specifying the way the result df-file will be saved: in the file with specified name or using already created file descriptor.
`pCmp` – using this parameter you can pass the pointer on the comparing context which participated in the successful `dfCompare` function call (see dfBuildN, dfBuildH functions descriptions for details).
`pBldOpt` – pointer on the structure containing parameters of difference file building and also additional application data, which can be saved in the result df-file.
`pInteract` – this parameter allows an application program to get information on the progress of difference file building process.
Successful termination of function `dfBuildN` (or `dfBuildH`) says that df-file is successfully built.

The next chapter of this manual is devoted to the following df-file applying >>

2.3.2. DF-file Applying

Now let's take up the sequence of actions needed to successfully restore a new file having an old file and a difference file (df-file).

The easiest way to apply df-file is to use the following function:

```
int dfApply ( _TCHAR const * szDfFile, _TCHAR const * szOldFileName, _TCHAR
const * szNewFileName );
```

You need to set names of those 3 files that participate in df-file applying, and namely: name of df-file, name of an old file and name of a new file. And that's all.

However to get extended features on df-file applying you need to use other functions.

```
int dfOpenDfFile ( _TCHAR const * szDfFile, TDfHandle * phDf );
```

This function opens df-file which name is set by `szDfFile` parameter and performs verification of its integrity. In case of successful call of `dfOpenDfFile` the value of the variable of type `TDfHandle` on which `phDf` points will be used as a special descriptor of the successfully opened df-file. Pay attention that this descriptor is valid only within LIBDF functions context and have nothing to do with common file descriptors, created for instance by system function `CreateFile`.

```
TDfCmp const * dfGetDfInfo ( TDfHandle hDf );
```

Using this function you can extract information from df-file about compared files which is identical to that represented by the *comparing context* at the moment of `dfBuildN` (or `dfBuildH`) calling.

Restoring of a new file is implemented using one of the following functions: dfApplyN or dfApplyH.
Difference between these functions lies in the method of specifying files which participate in the operation: names of files or already created common files descriptors and also in the additional parameter of df-file applying.

```
int dfApplyN ( TDfHandle hDf, _TCHAR const * szOldFileName, _TCHAR const *
szNewFileName, int iApplyFlags, TDfInteract * pInteract );
```

```
int dfApplyH ( TDfHandle hDf, FD_T hOldFile, FD_T hNewFile, int iApplyFlags,
TDfInteract * pInteract );
```

It is significant to note that iApplyFlags parameter can be used to cancel updating of any file component.
(See dfApplyN, dfApplyH functions descriptions for details)

2.4. Comparing and Diff

2.4.1. Functions

2.4.1.1. dfOpenCmpByNames

Summary

dfOpenCmpByNames creates new comparing context for two files, names of these file are set in parameters `szOldFileName` and `szNewFileName`.

Syntax

```
int dfOpenCmpByNames (

    _TCHAR const *   szOldFileName,
    _TCHAR const *   szNewFileName,
    int              bExtractPEVersions,
    PTDfCmp *        ppCmp );
```

Arguments

`szOldFileName`
[Input]
A pointer to a null-terminated string that specifies the name of an old file.
Can be NULL or empty string.

`szNewFileName`
[Input]
A pointer to a null-terminated string that specifies the name of a new file.
Can be NULL or empty string.

`bExtractPEVersions`
[Input]
Flag which specifies whether an attempt to obtain a version number (which may be stored in the PE-file resources) will be maid.
If value of `bExtractPEVersions` is not equal to "0", then this attempt is maid and if succeed, then pointers on file version numbers are saved (in a text format) in the fields of the difference context `szOldVersionID` and `szNewVersionID`.

`ppCmp`
[Output]
Pointer on variable of `PTDfCmp` type, which contains a pointer on the comparing context created in case of successful function termination.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_OPEN_ERROR (in case of error opening of an old file)
EDF_NEW_OPEN_ERROR
EDF_OUT_OF_MEMORY
EDF_SYSERROR
EDF_FATAL

Remarks

Filling of the context structure fields during the process of *comparing context* creating is processed subject to values of attributes of the compared files (see `TDfCmp` structure description for details).

Successfully created *comparing context* can be used later on when calling functions `dfCompare`, `dfBuildN` and `dfBuildH`.

Name of an old **or** new file can be skipped (i.e. set to NULL or left an empty string), but not both concurrently.

NOTE: Use function dfCloseCmp to delete the context and to free associated resources.

See also

TDfCmp, dfOpenCmpByNames, dfCompare, dfCloseCmp

2.4.1.2. dfOpenCmpByHandles

Summary

dfOpenCmpByNames creates new *comparing context* for two files, descriptors for these files are set in parameters `hOldFile` and `hNewFile`.

Syntax

```
int dfOpenCmpByHandles (

    FD_T       hOldFile,
    FD_T       hNewFile,
    PTDfCmp *  ppCmp );
```

Arguments

`hOldFile`
[Input]
Descriptor of an old file, created by function `CreateFile()`.
May be INVALID_FD_VALUE value, if old file is skipped.

`hNewFile`
[Input]
Descriptor of a new file, created by function `CreateFile()`.
May be INVALID_FD_VALUE value, if new file is skipped.

`ppCmp`
[Output]
Pointer on variable of `PTDfCmp` type, which contains a pointer on the *comparing context* created in case of successful function termination.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_OUT_OF_MEMORY
EDF_SYSERROR
EDF_FATAL

Remarks

Filling of the context structure fields during the process of difference context creating is processed subject to values of attributes of the compared files (see `TDfCmp` structure description for details). Fields `szOldFileName`, `szNewFileName`, `szOldVersionID` and `szNewVersionID` are left empty. You can set values for these fields on your own.

Successfully created *comparing context* can be used later on when calling functions `dfCompare`, `dfBuildN` and `dfBuildH`.
One of files whether an old **or** a new one can be skipped (value of the descriptor INVALID_FD_VALUE), but not both concurrently.

NOTE: Use function `dfCloseCmp` to delete the context and to free associated resources.

See also
`TDfCmp`, `dfOpenCmpByNames`, `dfCompare`, `dfCloseCmp`

2.4.1.3. *dfCompare*

Summary

dfCompare performs comparing of files, which are set by the *comparing context* and saves them in the context for further utilization upon df-file building using functions `dfBuildN` or `dfBuildH`.

Syntax

```
int dfCompare (
   PTDfCmp                pCmp,
   TDfCmpOptions   *   pCmpOpt,
   TDfInteract   *     pInteract,
   TDfCmpResult   *    pCmpRes );
```

Arguments

`pCmp`
[Input]
Pointer on the comparing context.

`pCmpOpt`
[Input]
Pointer on the structure with comparing parameters (see description of the structure `TDfCmp` for details).
If NULL then default parameters values are taken.

`pInteract`
[Input]
Pointer on the structure of interaction with application program (see description of the structure `TDfInteract` for details).
Can be set NULL.

`pCmpRes`
[Output]
Contains the result of files comparing in case of successful execution of the function of field structure, which pCmpRes points on (see description of the structure TDfCmpResult for details).

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_OUT_OF_MEMORY
EDF_FATAL
EDF_SYS_ERROR
EDF_READ_ERROR
EDF_ABORTED

Remarks

Function `dfCompare` can be called for an unlimited times with different parameters during all lifecycle of the comparing context.

See also

`TDfCmp`, `TDfCmpOptions`, `TDfInteract`, `TDfCmpResult` `dfOpenCmpByNames`, `dfOpenCmpByHandles`, `dfCompare`, `dfTestCmpResult`, `dfBuildN`, `dfBuildH`

2.4.1.4. *dfTestCmpResult*

Summary

dfTestCmpResult analyses results of comparing, produced with function `dfCompare` (structure `TDfCmpResult`) and makes a conclusion whether compared files are equal or not.

Syntax

```
int dfTestCmpResult ( TDfCmpResult *  pCmpRes );
```

Arguments

`pCmpRes`
[Input]
Pointer on a structure containing results of file comparing (produced with function `dfCompare`).
Can not be NULL.

Return Values

Returns "0", if files are considered identical.
Returns "1", if files are considered different by at least one of components (name, attributes, last modification date, content).
Returns "-1", if it is impossible to make a conclusion whether file are equal or not (as far as when creating of the comparing context one of files was skipped).

See also

`TDfCmpResult`, `TDfCmpOptions`
`dfCompare`

2.4.1.5. dfBuildN, dfBuildH

Summary

dfBuildN, dfBuildH functions are used to build a df-file based on the results of file comparing, presented by the *comparing context*.

Syntax

```
int dfBuildN (
    PTDfCmp                pCmp,
    _TCHAR const   *       szDfFile,
    TDfBldOptions   *      pBldOpt,
    TDfInteract   *        pInteract );
```

```
int dfBuildH (
    PTDfCmp                pCmp,
    _TCHAR const   *       hDfFile,
    TDfBldOptions   *      pBldOpt,
    TDfInteract   *        pInteract );
```

Arguments

pCmp
[Input]
Pointer on the comparing context.

szDfFile
[Input]
Name of the result df-file.
If specified file already exists then it is replaced.

hDfFile
[Input]
File descriptor, in which the result file is written. Previous file content will be lost.
Cannot be INVALID_FD_VALUE.

pBldOpt
[Input]
Pointer on the structure containing parameters of df-file building.
If NULL, then default parameters values are taken.
(See description of structure TDfBldOptions)

pInteract
[Input]
Pointer on the structure of interaction with application program (see description of the structure TDfInteract).
Can be set NULL.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_OUT_OF_MEMORY
EDF_FATAL

EDF_SYSERROR
EDF_OPEN_ERROR
EDF_READ_ERROR
EDF_WRITE_ERROR
EDF_ABORTED
EDF_DF_UNSUPP_COMPRESSOR

Remarks

`dfBuildN` and `dfBuildH` can be called immediately after context creation or after successful call of `dfCompare`.

If `dfBuildN` or `dfBuildH` are called after `dfCompare` terminates with an error, then EDF_BAD_PARAM is returned.

See also

`TDfCmp`, `TDfBldOptions`, `TDfInteract`
`dfCompare`

2.4.1.6. dfCloseCmp

Summary

dfCloseCmp deletes the *comparing context*, created by function `dfOpenCmpByNames` or `dfOpenCmpByHandles` and frees up associated resources.

Syntax

```
int dfCloseCmp ( PDfCmp pCmp );
```

Arguments

pCmp
[Input]
Pointer on the comparing context.
May be NULL.

Return Values

If function succeeds, the return value is zero (EDF_SUCCESS).
If function fails, the return value is EDF_FATAL

See also

`TDfCmp`, `dfOpenCmpByNames`, `dfOpenCmpByHandles`

2.4.2. Structures

2.4.2.1. TDfCmp

Summary:

TDfCmp structure can be used in 2 cases within library LIBDF :

- Representation of the comparing context;
- Representation of the df-file content information.

And all fields of this structure have the same definite value independently on its usage at that.

Syntax:

```
struct TDfCmp
{
   FILEPOS_T     nOldFileSize;
   FILEPOS_T     nNewFileSize;
   _TCHAR *      szOldFileName;
   _TCHAR *      szNewFileName;
   UINT32_T      dwOldFileAttr;
   UINT32_T      dwNewFileAttr;
   FILETIME_T    ftOldLastWriteTime;
   FILETIME_T    ftNewLastWriteTime;
   _TCHAR *      szOldVersionID;
   _TCHAR *      szNewVersionID;
   UINT32_T      nFlags;
   MD5_T         md5Old;
   MD5_T         md5New;
   void *        lpInternal;
};
```

Fields:

nOldFileSize – size of an old file. "-1" value if old file was skipped.
nNewFileSize – size of a new file. "-1" value if new file was skipped.

szOldFileName – pointer on a null-terminated string - name of an old file. NULL if name of an old file was not set.
szNewFileName – pointer on a null-terminated string - name of a new file. NULL if name of a new file was not set.
If using function `dfOpenCmpByNames` to create context, then fields `szOldFileName` and `szNewFileName` are initialized only by names of files (without path).
If using function `dfOpenCmpByHandles` to create context, then fields `szOldFileName` and `szNewFileName` are set equal to NULL value

dwOldFileAttr – old file attributes.
dwNewFileAttr – new file attributes.

ftOldLastWriteTime – old file last modification date/time.
ftNewLastWriteTime – new file last modification date/time.

szOldVersionID – pointer on a null-terminated string, representing version ID of an old file.
szNewVersionID – pointer on a null-terminated string, representing version ID of a new file.

nFlags – flags field.
Free combination of the following flags:

DFHF_CHG_NAME	Set if file names are different.
DFHF_CHG_DATETIME	Set if last modification date/time of files is different.
DFHF_CHG_ATTR	Set if attributes of files are different.
DFHF_CHG_CONTENT	Set if content of files is different.
DFHF_OLD_MD5	Value of old file's MD5 was calculated and saved in the field md5Old.
DFHF_NEW_MD5	Value of new file's MD5 was calculated and saved in the field md5New.
DFHF_EXTRA_DATA	Flag of 'extra data' presence in df-file.
DFHF_TE	df-file was built without opportunity to recover content of a new file (DF_CNTUPD_NONE). Such df-file cannot be applied.

`md5Old` – value of old file's MD5 if it was calculated (if the flag DFHF_OLD_MD5 is set in the flags field)

`md5New` – value of new file's MD5 if it was calculated (if the flag DFHF_NEW_MD5 is set in the flags field)

`lpInternal` - reserved for internal use. **Must not be changed by user!**

NOTE:

Values of fields `enOldFileSize, nNewFileSize, szOldFileName, szNewFileName, dwOldFileAttr, dwNewFileAttr, ftOldLastWriteTime, ftNewLastWriteTime, szOldVersionID, szNewVersionID` are set during creation process of the *comparing context* (functions `dfOpenCmpByNames` and `dfOpenCmpByHandles`) and may be changed later on by user at any moment (for string values it is allowed to change both directly strings themselves and values of field-pointers).
Other fields must not be changed by user!

Flags field `nFlags` is set when calling function `dfCompare` and must not be changed by user.

Location and time for calculation of values `md5Old` and `md5New` is not exactly determined (it can be functions: `dfCompare`, `dfBuildN` or `dfBuildH`) and depends on the compared files and on parameters of called functions. And moreover values of `md5Old` и `md5New` can be left uncalculated at all, if it is not required for the task of df-file applying. To ensure calculation of values of `md5Old` and `md5New` it is necessary to utilize function `dfBuildN` and `dfBuildH` and set flags `iForceOldMD5` and `iForceNewMD5` of the `TDfBldOptions` structure.

2.4.2.2. TDfCmpOptions

Summary:

TDfCmpOptions structure contains fields which define parameters of `dfCompare` function operation.

Syntax:

```
struct TDfCmpOptions
{
   int         iContentCmp;
   int         iContentUpdateData;
   int         iEqSearchMethod;
   int         iMaxMemoryUsageUnit;
   unsigned    nMaxMemoryUsage;
   int         iFileNameCmp;
   int         iFileNameCaseSensitive;
   int         iDateTimeCmp;
   unsigned    nDateTimePrecision;
   int         iAttrCmp;
};
```

Fields:

`iContentCmp` – defines whether files content will be compared or not, and if it will not then whether consider it identical or not.

Possible values:

-1	Files content is considered identical (no comparing or search for identical parts is performed).
0	***Perform comparing and if content is different then operate according to `iContentUpdateData` options (default value).***
1	Consider files content absolutely different (no comparing or search for identical parts is performed).

On the first glance it can be considered nonsensical or even absurd to take files content as equal if is not equal in fact, however in this way a mechanism of content updating method selection or content difference ignoring. So that if function `dfCompare` considers that compared files content is identical (as a result of this flag setting or actual files content comparing) then corresponding df-file, which is built by the following call of `dfBuildN`/ `dfBuildH` will not contain content updating data and in case of its applying - content of an old file will not be changed.

The same is with fields `iFileNameCmp`, `iDateTimeCmp` and `iAttrCmp` (see description below) with only one difference that they concern names of files, last modification date/time and file attributes.

`iContentUpdateData` – this parameter defines the method of difference representation within content of compared files.
Ignored if iContentCmp is "-1".

Possible values:

DF_CNTUPD_NONE	Perform comparing of files content in case of parameter iContentCmp value is equal to 0 and save in df-file only information on the result of file comparing (content is different or not). In case of presence of difference in content search for identical parts is not performed.

	The result df-file cannot be used for old file updating in this case. Used for results of file comparing caching.
DF_CNTUPD_SMART_DIFF	A special algorithm which minimizes the size of data, necessary for content differences encoding is used for identical content parts search ***(default value)***. Equivalent to value of `DF_CNTUPD_SAVE_ENTIRE` if `iContentCmp` is "1".
DF_CNTUPD_B2B_DIFF	Difference in file content is organized as per-byte difference of parts of the largest length (equal to the size of the smallest among compared file) starting from the zero byte (rarely used). Equivalent to value of `DF_CNTUPD_SAVE_ENTIRE` if `iContentCmp` is "1".
DF_CNTUPD_SAVE_ENTIRE	The whole content of new file will be saved in df-file. Can be used to get rid of the requirement of old file constancy at the applying side.

`iEqSearchMethod` – defines the method of identical file content parts search process **only if `iContentUpdateData=DF_CONTENT_SMART_DIFF`. Otherwise field value is ignored.**

Possible values:

DF_METHOD_DEFAULT (in this version of LIBDF it is equal to value of DF_METHOD_GOOD)
DF_METHOD_FASTEST
DF_METHOD_FAST
DF_METHOD_NORMAL
DF_METHOD_GOOD
DF_METHOD_BEST
DF_METHOD_PARANOID

WARNING! DF_METHOD_BEST и DF_METHOD_PARANOID methods can be very slow for big files!

`iMaxMemoryUsageUnit` – defines units for parameter `nMaxMemoryUsage` parameter value setting:
"0" - in % from total system RAM (default value);
"1" - in MBytes.

`nMaxMemoryUsage` – defines max RAM size which can be used during identical content parts search process (actual only if `iContentUpdateData=DF_CONTENT_SMART_DIFF`).
If equal to "0" then default value is taken: 80% from the total system RAM.
If set in MBytes, then allowed values range is: [1..2000].

`iFileNameCmp` – defines whether to perform comparing of names of files or not, and if not then whether to consider them identical or not.

Possible values:

-1	Consider names of files equal.
0	***Perform comparing (default value).***
1	Consider names of files different.

`iFileNameCaseSensitive` – flag which defines what method will be used for files names comparing - case sensitive or not.
If "0" - comparing method is not case sensitive, otherwise - case sensitive.
Ignored if iFileNameCmp is not equal to "0".
Default value: "0".

`iDateTimeCmp` – defines whether to perform comparing of files last modification date/time values or not, and if not then whether to consider them equal or not.

Possible values:

-1	Consider equal.
0	***Perform comparing (default value).***
1	Consider different.

`nDateTimePrecision` – defines precision of comparing of last modification date/time values of files in milliseconds.
Ignored if `iDateTimeCmp` is not "0".
Default value: "0".

`iAttrCmp` – defines whether to perform comparing of attributes of files or not, and if not then whether to consider them equal or not.

Possible values:

-1	Consider attributes of files equal.
0	***Perform comparing (default value).***
1	Consider attributes of files different.

2.4.2.3. TDfCmpResult

Summary:

TDfCmpResult structure is assigned to return the results of `dfCompare` function operation.

Syntax:

```
struct TDfCmpResult
{
   int   iContentDiff;
   int   iFileNameDiff;
   int   iDateTimeDiff;
   int   iAttrsDiff;
};
```

Fields:

`iContentDiff` – this field contains the result of file content comparing.

Possible values:

-1	Old file content differs from that of a new one.
0	Old file content is identical to that of a new one.
1	One of file (old or new) was skipped during comparing context creation process.

`iFileNameDiff` – this field contains the result of files names comparing.

Possible values:

-1	Names of files are considered different.
0	Names of files are considered identical.
1	One of file (old or new) was skipped during comparing context creation process.

`iDateTimeDiff` – this field contains th result of comparing of files last modification date/time.

Possible values:

-1	Values of Last modification date/time of files are considered different.
0	Values of Last modification date/time of files are considered identical.
1	One of file (old or new) was skipped during comparing context creation process.

`iAttrsDiff` – this field contains th result of comparing of files attributes.

Possible values:

-1	Attributes of files are considered different.
0	Attributes of files are considered identical.
1	One of file (old or new) was skipped during comparing context creation process.

2.4.2.4. *TDfBldOptions*

Summary:

TDfBldOptions structure is assigned to transfer parameters of df-file building to functions `dfBuildN` and `dfBuildH`.

Syntax:

```
struct TDfBldOptions
{
   int              iCompression;
   int              iForceOldMD5;
   int              iForceNewMD5;
   void const *     lpExtraData;
   size_t           nExtraDataSize;
};
```

Fields:

`iCompression` – sets method of df-file data compression. Only data which describes difference between compared files content is compressed.

Possible values:

-1	do not compress.
0	***Compression method(s) selected automatically so that minimize the result total size of df-file (currently only bzip2 available).***
1	bzip2

`iForceOldMD5` – flag setting of which ensures that as a result of successful call of `dfBuildN` (or `dfBuildH`) an MD5 will be calculated for an old file and saved in the field `md5Old` of the comparing context (see `TDfCmp` structure description).

`iForceNewMD5` – flag setting of which ensures that as a result of successful call of `dfBuildN` (or `dfBuildH`) an MD5 will be calculated for a new file and saved in the field `md5New` of the comparing context (see `TDfCmp` structure description).

`lpExtraData` – pointer on optional data (memory area with the size of `nExtraDataSize` bytes), which will be saved in a separate section of df-file (at the end of file).
Necessity of saving of additional data in df-file and data itself is determined by application.
May be set NULL value (no additional data).

`nExtraDataSize` – number of bytes of additional data.

2.5. Difference Applying

2.5.1. *Functions*

2.5.1.1. *dfOpenDfFile*

Summary

dfOpenDfFile function is used to open existing df-file ant test its integrity.

Syntax

```
int dfOpenDfFile (
    _TCHAR const *   szDfFile,
    TDfHandle *      phDf );
```

Arguments

`szDfFile`
[Input]
Name of df-file.

`phDf`
[Output]
Pointer on the variable of type `TDfHandle` in which a special df-file descriptor is placed. This descriptor is valid only as a parameter in functions of LIBDF library.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_OUT_OF_MEMORY
EDF_FATAL
EDF_SYSERROR
EDF_OPEN_ERROR
EDF_READ_ERROR
EDF_DF_DAMAGED
EDF_DF_UNSUPP_VERSION

Remarks
After completion of the work with df-file it must be closed by `dfCloseDfFile` function call.

See also
dfCloseDfFile, dfGetDfInfo, dfApplyN, dfApplyN, dfTouch

2.5.1.2. dfCloseDfFile

Summary

dfCloseDfFile closes df-file, opened by `dfOpenDfFile` function, and frees up all associated resources.

Syntax

```
  int dfCloseDfFile ( TDfHandle hDf );
```

Arguments

`hDf`
[Input]
Special df-file descriptor, created by `dfOpenDfFile` function.
May by NULL.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_FATAL

See also

dfOpenDfFile

2.5.1.3. dfGetDfInfo

Summary

dfGetDfInfo function returns information on df-file content.

Syntax

```
TDfCmp const * dfGetDfInfo ( TDfHandle hDf );
```

Arguments

`hDf`
[Input]
Special df-file descriptor, created by `dfOpenDfFile` function.
Can not be NULL.

Return Values

The function returns the pointer on the `TDfCmp` structure, which fields contain information on df-file content.
(See `TDfCmp` structure description for details)
The pointer is valid during all lifecycle of `hDf` descriptor.

See also
dfOpenDfFile, dfCloseDfFile

2.5.1.4. dfApplyN, dfApplyH

Summary

Functions **dfApplyN** and **dfApplyH** are used to apply a df-file, i.e. to restore a new file having an existing old file and df-file.

Syntax

```
int dfApplyN  (

    TDfHandle          hDf,
    _TCHAR const *     szOldFileName,
    _TCHAR const *     szNewFileName
    int                iApplyFlags,
    TDfInteract *      pInteract );
```

```
int dfApplyH  (

    TDfHandle      hDf,
    FD_T           hOldFile,
    FD_T           hNewFile,
    int            iApplyFlags,
    TDfInteract *  pInteract );
```

Arguments

`hDf`
[Input]
Special df-file descriptor, created by `dfOpenDfFile` function.

`szOldFileName`
[Input]
Name of an old file.
May be skipped if an old file was skipped when creating df-file. (Field `nOldFileSize` of `TDfCmp` structure is equal to "-1").

`szNewFileName`
[Input]
Name of a new file.
Can be NULL value or empty string for verification purpose.

`iApplyFlags`
[Input]
Flags which define the procedure of df-file applying.
Possible values are: "0" or free combination of the following flags:

DFAF_DONT_CHG_DATETIME	Do not change last modification date/time of the new file. If new file already exists then its last modification date/time is not changed. Otherwise an old file's last modification date/time is taken.
DFAF_DONT_CHG_ATTR	Same as for DFAF_DONT_CHG_DATETIME, but for the value of new file's attributes.
DFAF_DONT_CHG_CONTENT	Same as for DFAF_DONT_CHG_DATETIME, but for the new file content.
DFAF_DEL_OLD_FILE	This flag prescribes to delete an old file after df-file applying in case of

	old and new files are not identical. Valid only within dfApplyN function.

Default value: "0".

`pInteract`
[Input]
Pointer on the structure of interaction with application program (see description of the structure TDfInteract).
May be NULL.

`hOldFile`
[Input]
Old file descriptor.
Can be INVALID_FD_VALUE value, if old file was skipped when creating df-file. (Field `nOldFileSize` of `TDfCmp` structure is equal to "-1").

`hNewFile`
[Input]
New file descriptor.
Can be INVALID_FD_VALUE value or empty string for verification purpose.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_OUT_OF_MEMORY
EDF_FATAL
EDF_SYSERROR
EDF_READ_ERROR
EDF_WRITE_ERROR
EDF_ABORTED
EDF_TMP_CREATE_ERROR
EDF_DF_UNSUPP_COMPRESSOR
EDF_ALREADY_PATCHED
EDF_OLD_SIZE_UNEXPECTED
EDF_OLD_MD5_ERROR
EDF_NEW_MD5_ERROR
EDF_REPLACE_ERROR

Remarks

`szOldFileName` and `szNewFileName` may point on the same file. Then in case of successful df-file applying an old file is replaced by a new one.

`hOldFile` и `hNewFile` must represent different files when updating file.

At the moment of df-file applying a new file may already exist. Its replacement is implemented only after ensuring that new file restoration completed successfully (restoration is performed to a temporary file which then replaces the already existing one). That is why you can you can ensure that in case of any error the existing file will not be changed.
An moreover if the restored file replaces the already existing file and the existing file is on NTFS volume, then an attempt to pass the access permissions from the existing file to the restored one will me made.
It is necessary to mark that replacement of the already existing file is implemented only in case of file content updating.

These functions can determine the situation when an attempt to update the already patched file is made (in case of file content updating). Such situation can occur if you try to update the already updated file. In this case the function returns EDF_ALREADY_PATCHED error code.

See also

dfCloseDfFile, dfGetDfInfo, dfApplyN, dfApplyH

2.5.1.5. dfTouch

Summary

dfTouch performs updating of all file parts except its content.

Syntax

```
int dfTouch (
    TDfHandle          hDf,
    _TCHAR const *     szNewFileName,
    int                iApplyFlags );
```

Arguments

`hDf`
[Input]
Special df-file descriptor, created by `dfOpenDfFile` function.

`szNewFileName`
[Input]
Name of a new file.
Can be NULL, if new file was skipped when creating df-file.

`iApplyFlags`
[Input]
Flags which define the procedure of df-file applying.
Possible values are: "0" or free combination of the following flags:

DFAF_DONT_CHG_DATETIME	Do not change last modification date/time of the new file. If new file already exists then its last modification date/time is not changed. Otherwise an old file's last modification date/time is taken.
DFAF_DONT_CHG_ATTR	Same as for DFAF_DONT_CHG_DATETIME, but for the value of new file's attributes.

Default value: "0".

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_FATAL
EDF_SYSERROR
EDF_NEW_OPEN_ERROR

Remarks

Updating of the new file is performed only in case of this updating is defined in df-file and not masked by flags in `iApplyFlags` parameter.
If a new file was skipped when creating df-file, then function does nothing and returns EDF_SUCCESS.

See also
dfCloseDfFile, dfGetDfInfo, dfApplyN, dfApplyH

2.5.1.6. dfGetExtraDataSize

Summary

dfGetExtraDataSize provides extracting of the size of additional data which was saved in df-file when it was created.

Syntax

```
 int dfGetExtraDataSize (
     TDfHandle     hDf,
     size_t *      pnExDataSize );
```

Arguments

hDf
[Input]
Special df-file descriptor, created by dfOpenDfFile function.

pnExDataSize
[Output]
Pointer on the variable where the requested value is placed.
Can not be NULL.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
EDF_BAD_PARAM
EDF_FATAL

Remarks

If df-file does not contain additional data then function returns EDF_SUCCESS and *pnExDataSize is "0".

See also
dfGetExtraData, dfGetDfInfo

2.5.1.7. *dfGetExtraData*

Summary

dfGetExtraData provides extracting of additional data which was saved in df-file when it was created.

Syntax

```
int dfGetExtraData (
    TDfHandle    hDf,
    void *       buffer,
    size_t       size,
    size_t *     pnCopied );
```

Arguments

`hDf`
[Input]
Special df-file descriptor, created by `dfOpenDfFile` function.

`buffer`
[Output]
Pointer on the memory area where additional data is written.
Can not be NULL.

`size`
[Input]
Size of the output buffer, on which the `buffer` parameter points on.
Can be smaller then the additional data size.

`pnCopied`
[Output]
Pointer on the variable, in which the number of bytes of additional data written in `buffer` is returned.
Can not be NULL.

Return Values

If the function succeeds, the return value is zero (EDF_SUCCESS).

If the function fails, the return value is one of the following:
- EDF_BAD_PARAM
- EDF_OUT_OF_MEMORY
- EDF_FATAL
- EDF_SYSERROR
- EDF_READ_ERROR

Remarks

If df-file does not contain additional data then this function returns EDF_SUCCESS and `*pnCopied` is "0".

See also
dfGetExtraDataSize, dfGetDfInfo

2.6. Shared Structures

2.6.1. TDfChangePhase

Summary

Pointer on the structure **TDfChangePhase** is passed as the only parameter of the callback-function of process phase change notification.

Syntax

```
struct TDfChangePhase
{
    EnDfPhase     enPhase;
    int           bBegin;
    void *        lpPhaseData;
    void *        lpUserData;
};
```

Fields

enPhase – execution phase identifier (ID). Currently the following phases are defined:

Phase ID	Description	Function, which can use it
eaphContentCmp	File content comparing	dfCompare
eaphOldMD5	Calculating of old file's MD5	dfBuildN, dfBuildH
eaphNewMD5	Calculating of new file's MD5	dfBuildN, dfBuildH
eaphGenDf	df-file building	dfBuildN, dfBuildH
eaphApplyDf	df-file applying	dfApplyN, dfApplyH

bBegin – flag of beginning/ending of the phase execution.
Possible values:
"1" – phase beginning;
"0" – phase ending.
For each phase executed a callback-function `OnDfChangePhase` will be called 2 times: before phase beginning and after phase ending.
Notification of phase changing is called in such a way that two different phases cannot cross by time, i.e. if a notification of the 1st phase beginning comes then a notification of its ending will be called before a notification of the 2nd phase beginning comes.

lpPhaseData – pointer on additional data concerned with the current phase.
Currently the value of `lpPhaseData` is defined only for `eaphOldMD5` and `eaphNewMD5` phases: at their completion (bBegin is "0") the field `lpPhaseData` points on the calculated value of MD5 of an old and new file accordingly. In other cases field lpPhaseData contains NULL.

lpUserData – value passed to the field `lpUserData` of `TDfInteract` structure.
Can be used by application on its own.

2.6.2. TDfInteract

Summary

Execution time of some of LIBDF functions can appear to be prolonged enough especially in case of big files processing.
That is why in such functions the following features are provided: application program notification of execution phases, phase execution progress, and function interruption.
TDfInteract structure is used particularly fo these purposes, its pointer can be passed to the following functions: `dfCompare, dfBuildN, dfBuildH, dfApplyN, dfApplyH`.

Syntax

```
struct TDfInteract
{
    PFN_OnDfChangePhase   fnOnDfChangePhase;
    PFN_OnDfProgress      fnOnDfProgress;
    int *                 pAbortFlag;
    void *                lpUserData;
};
```

Fields

`fnOnDfChangePhase` – pointer on a callback-function which will be called when changing function execution phases.

Function must have the following prototype:

```
void OnDfChangePhase( TDfChangePhase const * pChgPhase );
```

The only function parameter `pChgPhase` points on the structure `TDfChangePhase` which fields are filled by the calling function (see TDfChangePhase structure description for details).
Can be NULL.

`fnOnDfProgress` - pointer on a callback-function which is called to indicate the progress of current phase execution.

Function must have the following prototype:

```
void OnDfProgress( TDfProgress const * pPrgs );
```

The only function parameter `pPrgs` points on the structure `TDfProgress`, which fields are filled by the calling function (see `TDfProgress` structure description for details).
Can be NULL.

`pAbortFlag` – pointer on the variable of int type which value is periodically checked by executed function. If the value of this variable becomes different from "0", then execution of the function is immediately terminated and function returns EDF_ABORTED error code.
Can be NULL.

`lpUserData` – pointer on user data.
Can be used by application on its own.

2.6.3. TDfProgress

Summary

Pointer on the structure **TDfProgress** which is passed the only parameter of the callback-function of the current phase execution progress.

Syntax

```
struct TDfInteract
{
    EnDfPhase     enPhase;
    float         fProgress;
    float         fEqProcent;
    void *        lpUserData;
};
```

Fields

enPhase – ID of the current phase.

fProgress – degree of current phase execution completion expressed in %.

fEqProcent – value of this field is actual only for `eaphContentCmp` phase. Contains the value equal to the sum of sizes of found identical parts in the old and new files, expressed in % relative to the size of the new file. Can be used for indication of how many identical parts have been found in the content of compared files.

lpUserData – value passed to the field `lpUserData` of `TDfInteract` structure.
Can be used by application program on its own.

2.7. Error Codes

Below is the full list of error codes which can be returned by functions of LIBDF library and their brief description.

ERROR	Code	Description
EDF_SUCCESS	0	Successful termination of operation
EDF_BAD_PARAM	1	Incorrect function parameters
EDF_OUT_OF_MEMORY	2	Not enough memory for operation
EDF_FATAL	3	Internal library error
EDF_SYS_ERROR	4	Unexpected system error Error code can be obtained by `GetLastError()` call.
EDF_OPEN_ERROR	5	Open file error
EDF_READ_ERROR	6	Read file error
EDF_WRITE_ERROR	7	Write file error
EDF_DF_DAMAGED	8	Patch damaged or not df-file
EDF_ABORTED	9	Operation aborted by user
EDF_NEW_OPEN_ERROR	100	New file open error
EDF_TMP_CREATE_ERROR	101	Temporary file creation error
EDF_DF_UNSUPP_VERSION	111	Unsupported version of the df-file
EDF_DF_UNSUPP_COMPRESSOR	112	Unknown compression method
EDF_ALREADY_PATCHED	120	Attempt to update the already updated file
EDF_OLD_SIZE_UNEXPECTED	121	Impossible to update content Unexpected size of the old file
EDF_OLD_MD5_ERROR	122	Impossible to update file content Old file MD5 error *** Indicates that old file content differs from that one which was used to create df-file
EDF_NEW_MD5_ERROR	123	New file MD5 error *** This error indicates that new file content was reconstructed incorrectly (mismatched MD5 of new file calculated when building df-file and when reconstructing a new file). Reasons can be as following: 1. df-file corrupted, though this corruption was not recognized 2. LIBDF library error
EDF_REPLACE_ERROR	124	New file replace error *** Impossible to perform replacement of already existing new file (it can be occupied by another process or out of system resources). You can call `GetLastError()` for details.
EDF_NO_CNTUPD_DATA	125	Impossible to update file content: during comparing process the field `iContentUpdateData` of structure `TDfCmpOptions` was set to the value DF_CNTUPD_NONE.

3. Frequenlty asked questions

1. General Questions:

• 1.1 Can I get the software on a disk or CD media?

You can include CD with installation file if you order DF SDK software at ShareIt!. It is shipped via Air Mail, within 2 business weeks. CD-ROM is free if buying a Commercial or a Corporate license, and 9.95$ value for Shareware license. At the subscription time in the future, if you lose your copy of the software due to a hard drive failure you can download the latest trial version from our site. If for any reason your registration code becomes invalid or lost - just send an e-mail to Sales Support with details of your previous registration. You will receive a new code via e-mail at no charge as soon as we validate your old registration data.

• 1.2 Will I get 'SPAM' if I give you my e-mail address?

No. We never, ever give out customer information or e-mail addresses in any manner.

• 1.3 Can I order via fax machine or phone, purchase with a check?

- Our sales agency now offers both a Toll-Free Voice Registration Service and a Fax Registration Service for buying our products. To order software using the Toll-Free Voice Registration Service, please check links within online secure order forms after selecting an appropriate license from our Order page. You must have DF SDK Product ID ready.
- Our sales agency gladly accepts payment by check or money order. However, please allow adequate time for the funds to be processed by our bank. For information about where to send your check or money order, please check links within online secure order forms after selecting an appropriate license at our Order page. You must have DF SDK Product ID ready.

• 1.4 What are your product IDs at RegSoft and ShareIt?

Please, visit our online FAQs page at our public website www.agensoft.com/faq.html

- **1.5 How long does it take to get my registration code or full version after I purchase a product online?**

After our sales agency receive your online credit card order, it may take up to several hours to authorize your transaction. As soon as your charge is authorized, you will receive an authorization e-mail with your Tracking ID# as well as instructions on how to obtain the full registered version the product you ordered. It is important that the customer check his or her e-mail to obtain the charge authorization and instructions. In the unlikely event that your credit card is declined, you will receive an e-mail stating the reason. If you do not receive any e-mail within 48 hours - there may be a problem with your order. In that case, kindly contact us by e-mail at sales@agensoft.com with your name, Tracking ID# and the approximate date and time of your order to obtain the status of your order.

- **1.6 What is the policy on updates...how much do they cost?**

All minor updates are free as of this writing (subject to change). Minor updates are those where the software version number to the right of the decimal change (minor updates usually slightly differs from each other), but the digit to the left of the decimal stays the same. For example, updates from 3.0 to 3.1 are free, however, 2.x to 4.0 will be on a cost basis.
Additionally, if a new major version does get issued, it is offered to our current customers at a discounted rate (50% discount) - it means that to register a major version update you will have to pay only 50% of its total price for each license to be renewed (for instance, $50 USD for one license renewal if the price is $100) to renew your registration.

- **1.7 What are the advantages for distributing patches?**

Distributing the changes as the "patch" has several significant advantages over distributing new version of the product. First, you don't have to make new CD-ROM's (or multiple floppy disks) and new boxes.
The patches are usually small and easy to distribute on single floppy or over the Internet. Due to the differential nature of the patches, you can also distribute your patches freely (from your web page for example), because it is impossible to install patch without previous (bought and registered) version of product.

- **1.8 What is the difference between patching and incremental updating?**

Incremental update contains all the files which have been changed between two versions.
DF-files made with DF SDK consists only of the changes from within each individual file with the help of byte-level differencing technology used by our patching engine, resulting in a significantly smaller update size.

2. Technical Questions:

- **2.1 What is the main idea of building update modules?**

The main idea of building patches is that patch file represents only information concerning changes made to an old version software product files relatively to a new version software product files. And if these changes are not significant relatively to total size of new version files than such delivering of update module (patch file/patch module) can become more effective method of update delivering.
Software has its bugs. These bugs are often discovered after the official release of the product. You are getting bug reports from your users. It's terrible that you spent your money to create fancy box, to record CD-ROM's and to distribute your software and several days after they appear. So, you should use "patch" which just contains description of changes you have made to your product since the official (or just previous) release. What's more, the difference between previous and current version to keep the patches as small as possible.

- **2.2 Why patch file size for EXE-files and DLL-files update sometimes appears to be rather large... ?**

This is so indeed, similar effect can occur sometimes. At recompilation of source code with insignificant changes regarding source code of the old version output file of newer version (EXE, DLL, etc.) most likely will have significant difference comparing with old version files. First of all, it is a result of some features of program binary code representation in Win32/PE executable files.

Here are some recommendations to reduce the size of output patch file in this case:

- Try to reduce files size by moving unchanged parts of the program (viz. their invariability from version to version or their insignificant changes) to dll-modules.
- Do not apply Exe-compression utilities to deflate executable files (if there is no extreme necessity).

But nevertheless PatchFactory considers all these features of EXE and DLL files and provides optimal patch building in these cases.

- **2.3 I need to update a database with things like adding columns or so. Does your software could help me to patch the database using sql or it can replace old files...?**

  DF SDK does not deal with any specific data structures, it operates with files and directories. Databases or files of other formats can be updated only as binary files (**warning:** database update can be implemented only if it is not changed on the end-user's machine).

- **2.4 Does the DF SDK require any third-party libraries to run?**

  DF SDK is entirely self-contained: it does not use any 3rd party libraries, and only requires libraries which are part of the base operating system.
  With DF SDK, you have the assurance that customers with different hardware and software configurations will always be able to update your software.

To obtain the latest FAQ you can visit our public website at http://www.agensoft.com.
If any question is not covered here, ask us: support@agensoft.com regarding technical issues (such as bug reports, feature suggestions, etc.)
and at sales@agensoft.com regarding sales issues (ordering problems, partnership suggestions, etc.)
You can also use our online email-form (preferably) to contact us.
We'll get in touch with you as soon as possible (usually within two business days).
* Do not forget to provide us with necessary technical information (Windows version, Detailed description of your problem,
and your registration information, if you are a registered user).

4. Registration and Support

4.1. License Agreement

END-USER LICENSE AGREEMENT

IMPORTANT: This End-User License Agreement ("EULA") is a legal agreement between you, either as an individual or a single entity ("Customer"), and AgenSoft Inc. ("AGENSOFT") for all AGENSOFT products which may include executable programs, redistributable modules, controls, dynamic link libraries, source code, demos, intermediate files, media, printed materials, and "online" or electronic documentation ("SOFTWARE PRODUCT(S)" or "SOFTWARE").

AGENSOFT grants to you as an individual, a personal, nonexclusive license to install and use the SOFTWARE PRODUCT(S) for the sole purposes of designing, developing, testing, and deploying application programs which you create. By installing, copying, or otherwise using the SOFTWARE PRODUCT(S), you agree to be bound by the terms of this EULA. If you do not agree to any part of the terms of this EULA, DO NOT INSTALL, USE, EVALUATE, OR REPLICATE IN ANY MANNER, ANY PART, FILE OR PORTION OF THE SOFTWARE PRODUCT(S).

All SOFTWARE PRODUCT(S) are licensed, not sold. If you are an individual, you must acquire an individual license for the SOFTWARE PRODUCT(S) from AGENSOFT or its authorized resellers. If you are an entity, you must acquire and assign an individual license for each Customer within your organization using and/or developing with the SOFTWARE PRODUCT(S) or acquire the appropriate license type (if applicable) from AGENSOFT or its authorized resellers.

If the SOFTWARE PRODUCT(S) you have obtained is marked as a "TRIAL" or "EVALUATION," you may install one copy of the SOFTWARE PRODUCT(S) for testing purposes for a period of 30 calendar days from the date of installation ("Evaluation Period"). You may not attempt to bypass or disable any of these limitations. These restrictions may only be removed by purchasing a license to use the SOFTWARE PRODUCT(S) from AGENSOFT. Upon expiration of the Evaluation Period, the SOFTWARE PRODUCT(S) must be uninstalled and all copies destroyed.

RIGOROUS ENFORCEMENT OF INTELLECTUAL PROPERTY RIGHTS. If the licensed right of use for this SOFTWARE PRODUCT(S) is purchased by you with any intent to reverse engineer, decompile, create derivative works, and the exploitation or unauthorized transfer of, any AGENSOFT intellectual property and trade secrets, to include any exposed methods or source code where provided, no licensed right of use shall exist, and any PRODUCT(s) created as a result shall be judged illegal by definition of all applicable law. Any sale or resale of intellectual property or created derivatives so obtained will be prosecuted to the fullest extent of all local, federal and international law.

1. LICENSE GRANT
AGENSOFT grants you a license to use the version of this SOFTWARE PRODUCT(S) according to the license type you have purchased. You may not modify the SOFTWARE PRODUCT(S) or disable any licensing or control features of the SOFTWARE PRODUCT(S) except as an intended part of the SOFTWARE PRODUCT(S)'s programming features. This license is not transferable to any other company, entity, or individual. You may not publish any registration information (serial numbers, registration keys, etc.) or pass it to any other company, entity, or individual. Permission to distribute the SOFTWARE is not transferable, assignable, saleable, or franchisable. Each entity wishing to distribute the package must independently satisfy the terms of the distribution license.

2. COPYRIGHT
Except for the licenses granted by this agreement, all right, title, and interest in and to the SOFTWARE PRODUCT(S) (including, but not limited to, all copyrights in any executable programs, modules, controls, libraries, electronic documentation, text and example programs), any printed materials and copies of the SOFTWARE PRODUCT(S) are owned by AGENSOFT. The SOFTWARE PRODUCT(S) is protected by copyright laws and international treaty provisions. Therefore you must treat the SOFTWARE PRODUCT(S) like any other copyrighted material except that you may (a) make one copy of the Software solely for backup or archival purposes, or (b) transfer the Software to a single hard disk, provided you keep the original solely for backup or archival purposes. You may not copy any printed materials that may accompany the SOFTWARE PRODUCT(S).

3. REDISTRIBUTION
a) In addition to the rights granted in Section 1, AGENSOFT grants you the right to use and modify the source code version of those portions of the Software designated as "sample code" for the sole purposes of designing, developing, and testing your software product(s), and to reproduce and distribute the sample code, along with any

modifications thereof, only in object code form, provided that you comply with Section 3.c.

b) In addition to the rights granted in Section 1, AGENSOFT grants you a non-exclusive, royalty-free right to reproduce and distribute the object code version of any portion of the SOFTWARE PRODUCT(S), along with any modifications thereof, in accordance with the above stated conditions.

c) If you redistribute the sample code or redistributable components, you agree to: (i) distribute the redistributables in object code only, in conjunction with and as a part of a software application product developed by you which adds significant and primary functionality to the Software; (ii) not use AGENSOFT's name, logo, or trademarks to market your software application product; (iii) include a valid copyright notice on your software product ; (iv) indemnify, hold harmless, and defend AGENSOFT from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your software application product; (v) not permit further distribution of the redistributables by your end user.

AGENSOFT PRODUCT(s) may include certain files ("Redistributable(s)") intended for distribution by you to the users of software applications which you create. Redistributables include, for example, those files identified in printed or on-line documentation as redistributable files, or those files preselected for deployment by an install utility provided with the SOFTWARE PRODUCT(S) (if any). In all circumstances, the Redistributables for the SOFTWARE PRODUCT(S) are only those files specifically designated as such by AGENSOFT.

Subject to all of the terms and conditions in this EULA, you may reproduce and distribute copies of the Redistributables, provided that such copies are made from the original copy of the Redistributables included with the SOFTWARE PRODUCT(S) or modified versions of the Redistributables which are provided to you by AGENSOFT or those which you create. Copies of Redistributables may only be distributed with and for the sole purpose of executing application programs permitted under this EULA that you have created using the SOFTWARE PRODUCT(S).

AT NO TIME MAY CUSTOMER CREATE ANY TOOL, REDISTRIBUTABLE, OR SOFTWARE PRODUCT(S) THAT DIRECTLY OR INDIRECTLY COMPETES WITH AGENSOFT SOFTWARE PRODUCT(S) WHICH UTILIZES ALL OR ANY PORTION OF THE SOFTWARE PRODUCT(S) contained within this installation.

Distribution by the CUSTOMER of any design-time tools (EXE's, OCX's or DLL's), executables, and source code distributed to Customer by AGENSOFT as part of this SOFTWARE PRODUCT(S) and not explicitly identified as a redistributable file is strictly prohibited. The Customer shall not develop software applications that provide an application programming interface to the SOFTWARE PRODUCT(S) or the SOFTWARE PRODUCT(S) as modified.

The Customer may NOT distribute the SOFTWARE PRODUCT(S), in any format, to other users for development or application compilation purposes. Specifically, if Customer creates a control using the SOFTWARE PRODUCT(S) as a constituent control, Customer may NOT distribute the control created with the SOFTWARE PRODUCT(S) (in any format) to users to be used at design time and or for ANY development purposes.

Customer MAY NOT REDISTRIBUTE any SOFTWARE PRODUCT(s) files if using an evaluation, trial, Not for Resale, or demo version of the SOFTWARE PRODUCT(s).

4. TRANSFER

You may NOT permanently or temporarily transfer ANY of your rights under this agreement to any individual or entity without prior written approval from AGENSOFT. Regardless of any modifications which you make and regardless of how you might compile, link, and/or package your programs, under no circumstances may the libraries, Redistributables, and/or other files of the SOFTWARE PRODUCT(S) (including any portions thereof) be used for developing programs by anyone other than you. Only you as the licensed Customer have the right to use the libraries, redistributables, or other files of the SOFTWARE PRODUCT(S) (or any portions thereof) for developing programs created with the SOFTWARE PRODUCT(S). In particular, you may not share copies of the Redistributables with other co-developers. You may not reproduce or distribute any AGENSOFT documentation without AGENSOFT's explicit permission.

If you are an entity (Company), you must acquire and assign a license to each Customer within your organization using and or developing with the SOFTWARE PRODUCT(S). With written notification to AGENSOFT, Company may transfer the license obtained for a Customer to another Customer employed or otherwise engaged by Company if the initial Customer is no longer employed or engaged by Company or is reassigned to another function within Company and no longer develops software applications using the SOFTWARE PRODUCT(S). In addition, with written notification to AGENSOFT, Company may transfer its license of the SOFTWARE PRODUCT(S) to a successor Company.

5. TRADE SECRETS AND CONFIDENTIALITY

a) The Software contains information or material which is proprietary to AGENSOFT ("Confidential Information"), which is not generally known other than by AGENSOFT, and which you may obtain knowledge of through, or as a result of the relationship established hereunder with AGENSOFT. Without limiting the generality of the foregoing, Confidential Information includes, but is not limited to, the following types of information, and other information of a similar nature (whether or not reduced to writing or still in development): designs, concepts, ideas, inventions, specifications, techniques, discoveries, models, data, source code, object code, documentation, diagrams, flow charts, research, development, methodology, processes, procedures, know-how, new product or new technology information, strategies and development plans (including prospective trade names or trademarks).

b) Such Confidential Information has been developed and obtained by AGENSOFT by the investment of significant time, effort and expense, and provides AGENSOFT with a significant competitive advantage in its business.

c) You agree that you shall not make use of the Confidential Information for your own benefit or for the benefit of any person or entity other than AGENSOFT, except for the expressed purposes described in the paragraph hereof entitled "REDISTRIBUTION", in accordance with the provisions of this Agreement, and not for any other purpose.

d) You agree to hold in confidence, and not to disclose or reveal to any person or entity, the Software, other related documentation, your product registration key file or serial number (if any) or any other Confidential Information concerning the Software other than to such persons as AGENSOFT shall have specifically agreed in writing to utilize the Software for the furtherance of the expressed purposes described in the paragraph hereof entitled "REDISTRIBUTION", in accordance with the provisions of this Agreement, and not for any other purpose.

e) You acknowledge the purpose of this section entitled "TRADE SECRETS AND CONFIDENTIALITY" is to protect AGENSOFT’s ability to limit the use of the data and the Software generally to licensees, and to prevent use of Confidential Information concerning the Software by other developers or vendors of software.

6. OTHER RESTRICTIONS

You may not rent, lease or transfer the Software. You may not reverse engineer, decompile or disassemble the Software, except to the extent applicable law expressly prohibits the foregoing restriction. Without prejudice to any other rights, AGENSOFT may terminate this License Agreement if you fail to comply with the terms and conditions of the agreement. In such event, you must destroy all copies of the SOFTWARE PRODUCT(S).

7. LIMITED WARRANTY

If within 30 days of your purchase of this SOFTWARE PRODUCT(S), you become dissatisfied with the Software for any reason, you may return the software to AGENSOFT (or your dealer, if you did not purchase it directly from AGENSOFT) for a refund of your purchase price. To return the Software, you must contact AGENSOFT and obtain a return material authorization (RMA) number. AGENSOFT will not accept returns of opened or installed software without an RMA number. Returns are subject to the deduction from your purchase price of a 20% restocking fee and all shipping costs. You agree that this limited warranty does not apply to a Software source code license as specified in section 5.e.

8. TECHNICAL SUPPORT

Technical support is usually provided via email only. At any time, if the unmodified SOFTWARE PRODUCT(S) fails to satisfy the performance described in the documentation, AGENSOFT will try to issue a workaround to help the Customer solve the technical problems. However, it is understood that AGENSOFT may not always be able to solve such problems. A service fee for such workarounds may be charged by AGENSOFT. In such case, AGENSOFT will notify the Customer with a quote in prior to issuing the workarounds.

9. NO OTHER WARRANTIES

AGENSOFT DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SOFTWARE PRODUCT(S), THE ACCOMPANYING WRITTEN MATERIALS, AND ANY ACCOMPANYING HARDWARE.

10. LIMITATION OF LIABILITY

IN NO EVENT SHALL AGENSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITH LIMITATION, INCIDENTAL, CONSEQUENTIAL, SPECIAL, OR EXEMPLARY DAMAGES OR LOST PROFITS, BUSINESS INTERRUPTION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OR INABILITY OF THIS SOFTWARE PRODUCT(S), EVEN IF AGENSOFT HAS BEEN ADVISED OF SUCH DAMAGES.

APART FROM THE FOREGOING LIMITED WARRANTY, THE SOFTWARE PRODUCT(S) ARE PROVIDED

"AS-IS", WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. THE ENTIRE RISK AS TO THE PERFORMANCE OF SOFTWARE PRODUCT(S) IS WITH THE PURCHASER. AGENSOFT DOES NOT WARRANT THAT THE OPERATION OF THE PROGRAMS WILL BE UNINTERRUPTED OR ERROR-FREE. AGENSOFT ASSUMES NO RESPONSIBILITY OR LIABILITY OF ANY KIND FOR ERRORS IN THE SOFTWARE PRODUCT(S), OF/FOR THE CONSEQUENCES OF ANY SUCH ERRORS.

11. GOVERNMENT-RESTRICTED RIGHTS

Use, duplication, or disclosure by the U.S. Government of the computer software and documentation in this package shall be subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7013 (Oct 1988) and FAR 52.227-19 (Jun 1987). The Contractor is AGENSOFT.

12. GOVERNING LAW

This agreement shall be governed by the laws of Moscow, Russian Federation, excluding the application of its conflicts of law rules and shall inure to the benefit of AGENSOFT and any successors, administrators, heirs and assigns. Any action or proceeding brought by either party against the other arising out of or related to this agreement shall be brought only in a STATE or FEDERAL COURT of competent jurisdiction located in Russian Federation, Moscow. The parties hereby consent to in personal jurisdiction of said courts. The United Nations Convention on Contracts for the International Sale of Goods is specifically disclaimed.

In addition to any other relief granted the prevailing party shall be entitled to recover its attorney’s fees and costs. THE PARTIES EXPRESSLY WAIVE THE RIGHT TO A TRIAL BY JURY. The parties acknowledge that any breach of this agreement may result in irreparable harm to AGENSOFT Inc., thereby entitling AGENSOFT to injunctive relief for any such breach in addition to any other rights or remedies that AGENSOFT may have. This Agreement is the entire agreement between you and AGENSOFT and supersedes any other communications or advertising with respect to the SOFTWARE PRODUCT(S). If any provision of this License is held invalid, the remainder of this License shall continue in full force and effect.

4.2. How to register

➢ **Please visit our public web-site for the latest Ordering information: http://www.agensoft.com**

As soon as we are notified that your order has been processed, a full version of DF SDK (or special download link) will be sent to you via email within 24 hours or one business day (usually within 12 hours and depends on the country you reside in). Ordering gives you the right to use DF SDK, receive technical support and use features available only for registered users.

We offer a 30-day money back guarantee on all our software. If you are not completely satisfied with your purchase, we will refund its total price excepting shipping fees within 30 days of receipt.

➢ **To buy DF SDK you can use one of our secure payment processing services:** *RegNow* or *ShareIt!* (services of Digital River), which are industry leaders in E-Commerce Payment Processing Services. All data exchanged during the payment process is SSL-secured and protected with a high level of encryption.

Our registration services currently accept the following bank cards: **Visa**, **Mastercard/Eurocard**, **American Express** , **Discover/Novus**, **JCB**, **Diners Club**. You can also order by **Cheque**, **Cash**, **Fax**, **Postal mail**, **Fax&Phone**, **Bank/Wire Transfer**, **Purchase Order** or **Paypal** - check links on the online secure order form. Don't forget to supply exact personal information and a valid e-mail address to which we can send your registration and update information.

The simplest and cheapest way to purchase DF SDK software is to order it online.
Customer information is considered confidential and will not be shared or distributed to any third party.

To get more info on DF SDK discounts and special offers visit DF SDK order page at
http://www.agensoft.com/order.html

To allow for all types of developers to use our software, we offer three types of licenses:	
• **Discounted Personal License**	offered as a service to the industry, primarily for single person companies with little revenue (such as shareware authors). The software is licensed to the name of the individual purchasing the license.
• **Standard Commercial License**	intended for small companies. The software is licensed to the name of the company purchasing the license. Maximum number of employees using the software is limited to 5. It also entitles an organization to receive high-priority support (with guaranteed answer within 2 business days) via email. If you would like to obtain more extended services or more employees to use the software - consider buying a Premium Corporate license.
• **Premium Corporate License**	intended for large companies and corporates with many employees. The software is licensed to the name of the company purchasing the license. And besides this type of license entitles an organization to receive one copy of the distribution software and to duplicate the software for any number of people or workstations within the corporation. It also entitles an organization to receive high-priority support (with guaranteed answer within 1 business day) via email and free major version upgrades during lifetime of the product.

NOTE: Source code can be provided only with Commercial or Corporate license for additional payment.

NOTE: All licenses include the royalty-free distribution of modules marked as redistributables both for you and your customers for an unlimited number of applications, free minor version updates with special 50% discount for major upgrades during lifetime of the product and high-priority technical support via email and public support forums.

NOTE: The price for Personal, Commercial and Corporate licenses as well as other service terms are subject to change without any notice. To get the latest information about our pricing and discounting

policy and other services, you can *visit our public web-site at* www.agensoft.com.

If you do not receive a responce from us within a reasonable amount of time (usually two business days for credit card payments or two weeks for other payments), please notify us by email at sales@agensoft.com. We request your apologize for any inconvenience caused by those delays if any.

If you have any problems after you placed an order, visit our Ordering Problem and FAQ pages or send an e-mail to our Sales Support Team at sales@agensoft.com. Please include your name, e-mail address, and detailed description of your problem. We make every effort to reply to all e-mail inquiries within two to four hours with a maximum of 48 hours or two business days.

Competitive Upgrade - up to 50% OFF for migration to DF SDK. If you have previously purchased an updating solution and have experienced any options, performance, or reliability issues, AgenSoft offers you the chance to upgrade from your current vendor's software to PatchFactory. Please forward a copy of the order confirming your purchase of that updating software at sales@agensoft.com and we'll gladly e-mail you the special instructions to obtain your upgrade to PatchFactory.

Purchasing a DF SDK license entitles you to the following:

- Royalty-free distribution of DF SDK redistributable modules in your applications
- Free minor version updates during lifetime of the product
- Free Technical Support (via email and public Support Forums)
- Access to all Technical Support resources (including hidden area available only for registered users)
- Discounted License Upgrade and Registration Renewal price

* *"Free minor version updates"* means that by purchasing version 1.0, you have also purchased 1.1, 1.2 and all other 1.x versions, but not version 2.0 and later. However, you will get free major version update if it has passed less than 3 months since the date of your purchase.

4.3. Update and support

- **Update**

As a registered user of DF SDK you will be able to receive all minor updates of DF SDK free of charge.
It means that by purchasing version 1.1, you have also purchased 1.2, 1.3 and all other 1.x versions, but not version 2.0 and later. However, at the present time there we are not planning to release a new major version update and it is entirely possible that only minor versions will ever be published.

Update information: www.agensoft.com/updates
Download information: www.agensoft.com/download.html

- **Support**

If you have a question regarding the operation of DF SDK, please take a moment to review the most common Frequently Asked Questions (FAQs). You may find there the answer you are looking for! If there is no answer do not hesitate to contact us. We'll help you and respond to your message as soon as possible (usually within 24 to 48 hours).

Support information: www.agensoft.com/support

You can use our online email-form (preferably) to contact us. Just fill all necessary fields and click on "Send Email" button. Do not forget to select the correct Question Category for your message and write the detailed description of your problem - it can significantly shorten the respond time.

All comments and suggestions concerning improvements to DF SDK are appreciated!

When asking for technical support please inform us about the following:
- DF SDK version installed (from the "About" dialog including build number)
- Full OS version installed (including Service Pack installed and Localization)
- Detailed description of your problem (with a screenshot if possible)
- Your registration information (if you are a registered user)

NOTE: before requesting technical support, please ensure that you are using the latest version of DF SDK.

5. Contact information

Contacting AgenSoft...

Please refer to the E-mail addresses below for further information.

- info@agensoft.com
  General information and feedback. Any suggestions and comments will be welcomed!

- support@agensoft.com
  Customer technical support service or bug report.

- sales@agensoft.com
  Purchase or registration code affairs. We will be happy to help you.
  Software distribution, promotion in software compilations, or business cooperation. We are open to various levels of cooperation.

- webmaster@agensoft.com
  Exchanging links or placing advertisement.

You can also use our online email-form (preferably) to contact us.

Visit our public web-site www.agensoft.com for further information.