

# Update2

Version 2.8

## Technical Reference Manual

---

### What is Update2:

Update2 is a maintenance tool, for the updating of files between different computers, or on the same computer between different folders and their subfolders.

Update2 was designed with the developer and the site administrator in mind.

Update2 executes instructions from a text-file, called a **script-file**. Script-files have a .UPD filename extension. As any Windows application, .UPD script-files are associated with the Update2 application: clicking on a .UPD script-file starts Update2, which executes the clicked .UPD script-file. The UPDATE2 script language is detailed through this document.

### Who needs Update2:

Update2 is typically used in situations such as the following:

- Your accounting data files are on your server, and you want to copy them daily onto the local hard disk of your station, as an additional backup. Moreover, on Tuesdays, files get copied to your C:\BACKDATA\TUE\ folder, and so on.
- You want to keep a copy of all your Word documents, letters and offers, on your removable drive or flashdisk, which you use to refresh your home copy of these documents. Symmetrically, any document modified or created at home must be transferred the same way back to the office. You want this to be done without having to copy all your files every time, but only those that have been changed.
- You are many programmers working on a common software project, or many engineers working on a common architectural project. A master copy of your project files (program sources, drawing files) is on the server, and everyone has a local copy on his local station. Every now and then, each programmer or engineer launches an update session, which copies his latest work on the server, and gets the latest work of other colleagues.
- Every time you start your application (such as PIMS), you want to be sure your station is working using the latest version. New versions are initially copied onto the server, so you want Update2 to start by updating your local copy of the application executable file from the server, then by launching this local copy.

### How to obtain Update2:

Download the Update2 freeware utility from our web-site <http://www.profiles-software.com>.

It is also listed on a number of freeware and utilities websites.

Or simply mail us at [update2@profiles-software.com](mailto:update2@profiles-software.com).

### Update2 History:

*Version 2.8 (January 2009)*

- Various cosmetic improvements and functional corrections.

---

### Update2 – by Profiles Software

version 2.8

page 1 / 15

[www.profiles-software.com](http://www.profiles-software.com)

support: [update2@profiles-software.com](mailto:update2@profiles-software.com)



PROFILES  
SOFTWARE

#### Version 2.6 (January 2008)

- Update2 auto-registration with Windows upon each startup was triggering security breach messages, as new operating systems are more and more protected against such interference. Registration with Windows has been revised, and can be now done on demand, from the About page.

#### Version 2.5 (June 2005)

- Update2 can accept now URL paths (such as \\server\data), not only drives and mapped drives (such as C:\data).

#### Version 2.4 (June 2005)

- upgraded so that it can handle files exceeding 2 Gb.

#### version 2.3 (December 2001)

- numerous cosmetic improvements
- more user control on interactive action
- minor bugs fixed

#### Version 2.2.1 (June 2001)

- When insufficient free-space reported, checkbox allows copying anyway.
- Added the /Protected command-line option

#### Version 2.2

- Resizable form.
- Pause before copy checkbox, now controlled by Key [BeforeCopy].
- Binary compare after copy, check-box and Key [CompareAfterCopy].
- Context-sensitive help available.

#### Version 2.1

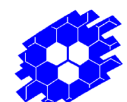
- Edit the script from Update2; reload the edited script if edited outside Update2.
- Stop between scan & copy, check copy/delete actions to be performed, choose to proceed or to abort.

#### Version 2.0

- Initial release of the Windows-based new version.

#### Version 1

- This DOS-based tool was initially developed in 1986, for the same purposes. Its last upgrade (2.6) was released in February 1991.



## Update2 Command-Line:

On its command line, Update2.exe expects the name of a xxxx.upd script-file, as the first command-line parameter.

It also accepts a left/right directory as command-line parameters, to be pre-pended to specified left/right directories.

The **/Protected** parameter (which can also be supplied as /P) puts Update2 in a protected mode, where only the provided script can be executed, it cannot be edited or reloaded, and no other script can be loaded.

The command line therefore looks like:

```
Update2 MyBackup.UPD R=c:\pims L=\\server\projects\pims /P
```

## Update2 script language:

### Script-file contents:

The Update2 script language is based on text-lines, to be considered as program-lines. In these lines, case (upper or lower) is always irrelevant and disregarded. Script files can be written using any simple text-editor or word-processor, typically NotePad or similar.

A script-file contains:

- Some global settings.
- A list of FileSpec-records, and their properties. FileSpec libraries may be also imported.
- Update-Session settings, then session execution. Multiple subsequent update-sessions may be executed.
- Finally, an application to be executed at end of the Update2 process (such as updated PIMS.app on network station).

### Key Assignment:

A key-assignment line is made of: a key, an equal-sign, a value: KEY=NEWVALUE.

In the specifications hereafter, we shall use the format:

```
definedkey = samplevalue (defaultvalue).
```

When the possible value must be one of a set, the possible values for the key shall be specified as follows: myanswer=[**Yes, No, Maybe**]. In such case, the first letter of the option is enough. If, however, more letters are specified, they must match the letter of the option. For instance: **M, ma, Mayb, maybe** are all legal and mean "Maybe", while **Moybe** is illegal.

When no default value is specified, this indicates that the default value is "empty".

### Commands:

Commands are a single word, on its own on the line, with no equal sign. They represent a command that Update2 must execute.

### Comment-lines:

Lines starting with semi-colons, such as **;comment**, are comment-lines, their contents are disregarded.

---

## Update2 – by Profiles Software

version 2.8

page 3 / 15

[www.profiles-software.com](http://www.profiles-software.com)

support: [update2@profiles-software.com](mailto:update2@profiles-software.com)



PROFILES  
SOFTWARE

## **Key-Values special contents:**

Within the value-part of a key-specification script-line, the following occurrences are converted into appropriate values:

### **Environment variables:**

Any text enclosed between percent signs is considered as an "environment variable", and replaced by the current value of this environment variable (just like it used to be in DOS batch-files).

For instance, if an environment variables contain the line **SERVER=F:**, any occurrence of **%SERVER%** in the value part of the script shall be replaced by **F:**.

In DOS, you used to define such environment variables by including lines stating SET VARIABLE=VALUE in batch-files, typically in the AUTOEXEC.BAT file.

In Windows, environment variables are defined in a form usually accessible by right-clicking on My Computer, and selecting Properties.

### **Timer Special Variables:**

These variables are mainly used when automatically creating backup copies. For instance one may ask for his files to be copied to a folder specified by F:\BACK\W{WEEK}. This way, on the 1<sup>st</sup> week of the year, files are copied to F:\BACK\W01, on the 2<sup>nd</sup> week they get automatically copied in F:\BACK\W02, and a complete historical back-track of these files is therefore kept.

- **{DAYNUM}** shall be replaced by the current 2-digit day-of-the-month: 01...31.
- **{DOW}** shall be replaced by the 3-letters English day-of-the-week of the current day, one of the following: MON, TUE, WED, THU, FRI, SAT, SUN.
- **{WEEK}** shall be replaced by the 2-digit number of the current week, 01...52. For defining the current week, week number 1 are the seven days from January 1<sup>st</sup> up to January 7<sup>th</sup>. Week 2 includes from January 8<sup>th</sup> up to January 14<sup>th</sup>, regardless of where Mondays or Sundays would fall.
- **{MTHNUM}** shall be replaced by the 2-digit number of the current month, 01...12.
- **{MTHTXT}** shall be replaced by the 3-letters English acronym for the current month, such as JAN, FEB, etc.
- **{YEAR2}** shall be replaced by the current 2-digit year number, without century.
- **{YEAR4}** shall be replaced by the current 4-digit year number, with century.



## FileSpec keys:

Every FileSpec is a record that associates to a given file wildcard-name, a number of properties that are to be applied to such files.

FileSpec-records are created sequentially, as they occur in the script-file. When processing a file, Update2 looks for a matching FileSpec, starting from latest created FileSpec back to the initial master default, which is \*.\* , and matches any file.

At any point in time, Update2 points to a given FileSpec record. When a FileSpec-related key is executed, it applies to the currently pointed-to FileSpec.

Here are the FileSpec-related keys.

### **FileSpec = STK\*.DOC**

Locates an existing FileSpec-record for the specified file-mask (stk\*.doc), and makes it the currently pointed-to FileSpec record. If no FileSpec-record exists for the given file-mask, it creates a new FileSpec-record with the default values for each FileSpec key.

Actually, when Update2 is first started, a master FileSpec is created for files \*.\* , so that it matches any file. Key values executed before the first FileSpec is specified apply to the initial \*.\* FileSpec.

### **Ignore = [No, Yes]**

If Ignore=Yes is specified, then Update2 will always ignore any such source files. No further comparison or copying shall be performed. This would be typically used to systematically ignore all ~\$.DOC files that Microsoft Word creates during editing.

FileSpecs \*.bak and \*.tmp are also typical candidates, that you would usually want to ignore.

### **Compare = [DateSize, Binary]**

Indicates the compare-method to be used with files matching this FileSpec. The default method is DateSize.

DateSize means files are considered similar if they have same date/time and same size.

Binary means that when date/time and size do match, the source and target files are also compared byte-by-byte, over their full length.

Same date/time and different size, or same size and different contents indicates a virus, or some buggy application (Delphi for instance may change the contents of its DFM files, while maintaining date/time unchanged). In such cases, the user is always asked whether to copy from source to target, from target to source, or to abort.

Note-1: time-stamps with a difference of 1 second are considered as equal, since Windows-95 has a resolution of 1-second, while DOS has a resolution of 2-seconds, even when started from within Windows.

Note-2: Update2 may encounter files that are dated in the future. If the encountered file is dated beyond present date/time by up to 24 hours, Update2 assumes this file was obtained by saving an e-mail attachment received from overseas. E-mail attachments time-stamps are usually affected by the time lag between the sending and the receiving country. In such case, Update2 always stops, asking what to do. If the file is time-stamped in the future, in excess of 24 hours, this can no longer be a time lag, and Update2 aborts at once, on the assumption that some clock must be out of tune within the network.



**Backup = ~P1 , ~P2 , BAK , TMP**

Specifies a comma-separated list of backup extensions to be used. This key has no default value.

The number of extensions in the list specifies how many backup generations are maintained, using which extensions. The sample list here means: 4 backup generations are kept, as follows: any TMP is deleted, then any BAK is renamed to TMP, then any ~P2 is renamed to BAK, then ~P1 is renamed to ~P2. Then the target file is renamed to ~P1 and the new file is copied.

All backup filespecs are implicitly excluded from the update process. In this example, all files \*.~P1, \*.~P2, \*.BAK and \*.TMP are implicitly excluded, and no attempt to update them will be made.

**MissingFile = [Ask,Create,Delete]**

Indicates how Update2 reacts when the target file is missing. It may Ask the operator, it may automatically Create the missing target file, or it may automatically Delete the Source file (used for instance in cleanup processes). By default, Update2 asks the operator.



## Global keys:

Global keys apply throughout the execution of a script-file, and define overall behavior.

### **Title = Daily updating of workfiles**

When a Title is specified, it is shown on the title bar of the Update2 window, to inform about the main action performed in this script. If no Title is specified, the full name of the script-file is used as a Title, until a first Message is encountered. The Message is then used as a Title.

### **Message = Now updating library source files**

As soon as a Message Key is specified, it is displayed in an appropriate area on the Update2 execution form, and remains displayed until replaced by another Message directive. Useful to hint the operator in clear language on which part updating operation is currently being performed.

### **FileSpecLibrary = c:\common\myfiles.upd**

Specifies another Update2 script-file, to be loaded as a FileSpec library. The specified script file is opened and read, but only FileSpec-related key-definitions are interpreted out of the script file. The script-file is considered as a library-file, and its FileSpec-related key-definitions are considered as if they were included within the current main script file. Every library script-file specified adds FileSpec definitions to the current executing Update2 environment, and multiple script library-files may be picked from different places, and “piled-up”.

This would allow “inheriting” FileSpec definitions from a known script file. In addition, script files may be written, that contain only FileSpec-related keys, to be used exclusively as library-files. Such scripts would contain no Commands.

The specified FileSpecLibrary file must exist, else an error will occur and the script will fail to run.

### **Include = c:\common\project2.upd**

Specifies another Update2 script-file, which is opened and read, and all its lines are considered as if they were included within the current script file. Typically, this allows generating a “master” script file, which contains only “Include” script-lines, and acts as a list of script files to be executed.

The specified Include file must exist, else an error will occur and the script will fail to run.

### **IncludeTry = c:\commercial\offers.upd**

Same as Include but, if the included file does not exist, Update2 will continue normally the process, without any error message. For instance, on a network, the line above executes only on computers where the **offers.upd** file exists, restricting it to people concerned with offers handling.

### **ExitFolder = c:\pims**

Specifies which must be the current folder when Update2 finishes processing the current script file. Where this line occurs in the script file is irrelevant. In pre-Windows days, this was also called the CWD, for Current Working Directory.



**ExitProgram = c:\fp25\foxprox -t c:\pims\abcd.app**

Specifies a complete command line, i.e. a program path and name, and possibly additional command-line parameters. Once Update2 is done processing the current script file, it launches any specified application, before terminating itself. Where this line occurs in the script file is irrelevant. Typical usage: update accounting files local copy, then launch accounting package.

**WhenFinish = [Stay,Close]**

Stay: Update2 does not close automatically after executing the script. Instead, it waits idle for you to click the Close button. This is the default behavior when you start Update2 then load the script from within Update2.

Close: after executing the script, Update2 waits 30 seconds before closing by itself. This delay allows you to take control, if you want to. One may want to take control after all work was done, in order to check the history of copied files. This is the default behavior when Update2 is launched by double-clicking a script file.

**CompareAfterCopy = [No,Yes]**

You may occasionally suspect problems with your operating system, your network or drive connections, or the like. The only ultimate way to make sure that the copied files were indeed copied properly is to perform a binary compare after copy on each file, after it was copied. A checkbox related to this key exists on the History page. Any manual checking overrides the value given for this key in the script.

"No" is the default value.

**BeforeCopy = [Nopause,Pause]**

You may ask Update2 to stop right after the scanning phase, allowing you to review the Actions History, then deciding whether you wish to proceed or to abort the Update2 process. A checkbox related to this key exists on the History page, any manual checking overrides the value given for this key in the script.

"NoPause" is the default value.





## Update-Session Keys and Commands:

An update-session introduces the mask of files that are the subject of this session, default left & right root-paths, right and left paths, excluded files, action options, updating mode, recursive updating, and what to do with missing folders. A Session is actually performed where the *Execute* command was specified.

```
SessionFiles = MSG*.DOC (*.*)
RightPath    = \DEV\SYS\PIMS
LeftPath     = \PIMS\DEV
RightRoot    = C:
LeftRoot     = \\SERVER\DEVELOP,10
```

All actions in an update-session occur between right-files and left-files, and/or vice-versa. To know where and which the right-files are, the full-path is built by concatenating the RightRoot key value currently in effect, the current RightPath, and the current SessionFiles. In the above example, the left-files are \\SERVER\DEVELOP\PIMS\DEV\MSG\*.DOC.

LeftRoot and RightRoot have a special, additional parameter, called the **Priority**. This priority works as follows: Where a LeftRoot or RightRoot key is encountered, it replaces the current LeftRoot or RightRoot only if its priority is equal or higher than the current one. After **LeftRoot=C:,20** was executed, any of **LeftRoot=D:,20** or **LeftRoot=E:,21** will change the value of LeftRoot to D: or E: respectively, while **LeftRoot=F:,19** keeps the value of LeftRoot unchanged.

When no priority is specified, such as in **LeftRoot=X:**, the default priority is 10.

If a R= parameter was supplied on the command-line, it acts as a RightRoot key, with a priority of 100. Similarly, the L= parameter acts as a LeftRoot with a priority of 100.

LeftPath and RightPath may start with the special NoRoot character #. If, for instance, LeftPath is specified as #VALUE, then LeftRoot is not used in building the full path. This shall be used in cases where the updated files are in a location which is unrelated to the current LeftRoot, such as LeftPath=#C:\LOCAL while LeftRoot=\\SERVER.

After performing an Execute command, one may specify a different SessionFiles setting, yet keeping the RightPath and LeftPath settings, and perform another Execute.

```
Exclude = *.wrk, My Doc*.*, TEMP\ ,/PROG*.IDX
```

This is a comma-separated list of file-masks to be excluded. Any file in the scanned SessionFiles, which filename matches one of the file-masks specified here, will be skipped.

To Exclude a complete subfolder, with all its files and nested subfolders, just add the subfolder name with a \ (backslash character) at the end of it. For instance, if you add **TEMP\** to the exclusion list, while the current source-path is **C:\USER**, then **C:\USER\TEMP\** contents are all excluded, whether files or subfolders. Note that **C:\USER\DATA\TEMP\** contents are not excluded.

The usage of the slash character / before any folder-name or file-mask means that contents are excluded only in the directly indicated folder, without excluding the contents of any of its subfolders. For instance, if the current source-path is **C:\USER**, then **/PROG\*.IDX** excludes **C:\USER\PROG\MYFILE.IDX**, but does not exclude **C:\USER\PROG\MORE\MYFILE.IDX**.

Exclude is reset to an empty list after each Execute.



## **UpdateMode = [Update, LMaster, RMaster, 1LTR, 2RTL]**

**Update:** Bi-directional updating. In a first pass, LTR (Left-To-Right), Left files are compared to Right files, and any newer files are copied. In a second pass, RTL (Right-To-Left), Right files are compared to Left, and any newer files are copied. At the end of the process, Left and Right are similar, and contain a mix of the newer files of both sides. Typically used in development environments, to merge the work of multiple workstations.

**1LTR, 2RTL:** only the LTR or the RTL pass of the above process is performed.

**LMaster, RMaster:** LMaster means that the Left-files are considered as a Master Copy, while the Right files are considered as a Local Backup. On such Local Backup, no action ever takes place (such as a genuine backup on –say– a Zip-drive), or any action is temporary and must be overwritten (Read-Only copy of Master accounting files, on Manager station, for follow-up). After this updating, files are copied from Left to Right, and files may be deleted from Right, until Right becomes an exact copy of Left, while Left remains unchanged. It may be thought-of as an optimized copy operation.

In these modes, the following settings are assumed, while the currently standing value is ignored: SessionFiles=\*.\*, and MissingFiles & MissingFolders are created and deleted as needed.

The default is Update, and this setting is reset to this default after each Execute.

## **SubFolders = [Included, Excluded, Yes, No]**

When Subfolders are Included, all the sub-folders of the source folder (whether right or left) are also scanned for matching SessionFiles. The default value is Included or Yes, and this setting is reset to this default after each Execute.

## **MissingFolder = [Userask, Auto, Skip]**

Indicates how Update2 reacts when a folder is missing. The missing folder may be the root source folder, where the source files are, or any folder on the target. The default value is Userask, and this setting is reset to this default after each Execute.

**Userask:** If the root source folder is missing, a message notifies the operator, and this part of the current Execute action is aborted. If a target folder is missing, Update2 ask the operator if we Skip, Create it on the target, Delete it from the source, or Abort the whole script.

**Auto:** If a target folder is missing, it is automatically created. If the root source folder is missing, then we have two cases. If the UpdateMode is Update, then this part of the current Execute action is skipped silently, since this UpdateMode is bi-directional and this source folder will be later a target folder, and will be created automatically. For any other value of UpdateMode, a message notifies the operator, and this part of the current Execute action is skipped.

**Skip:** If any of the source folder or the target folder is missing, the current Execute action is skipped silently.

## **Execute**

When this Command script-line is encountered, the Update2 process is executed with the standing parameters, and using the current FileSpec specifications. After each Execute, the following keys are reset to their default values: SessionFiles=\*.\*, Exclude=(none), MissingFolder=UserAsk, SubFolders=Included, UpdateMode=Update.



## Reset

When this Command script-line is encountered, the entire Update2 environment is “reset” to its initial state, as if Update2 were just started. This may be for instance used before “including” another script-file, to avoid inheriting any settings from the current script into the included script.



## Other Considerations:

### Error Handling:

Update2 may occasionally encounter system errors, such as the inability to access a network drive. In such case, Update2 will instantly abort execution at the point where the error is encountered, simply leaving what was done as done, and what was not as not done.

### Registering with Windows:

Upon startup, Update2 will check if it has been registered with Windows or not yet, or if the currently launched copy is the one registered; it will propose registration accordingly.

Just with any other application, you may be unable to register Update2 if you do not have Administrator rights on the current machine. You may register again anytime, by clicking the registration button from the About page.

Registering with Windows provides the following features:

#### Launching Update2:

Double-clicking or launching in any way an Update2 script-file, i.e. a file with the .upd extension, will automatically invoke the registered Update2 application, just as double-clicking on a file with the .txt extension would invoke Notepad.

#### Editing script:

Whether you right-click on the Update2 script-file and choose the Edit function, or whether you click the Edit button in the Script page of Update2, it will try to invoke the registered text editor.

Upon registering with Windows, Update2 will register the .upd extension for Edit, with the same editor as the one registered for .txt text-files.



## A sample \*.UPD script-file:

Probably the best way to explain how Update2 script-files are written and executed is to propose a sample script-file, and to comment it line by line.

Here is the sample script:

```
; Updating PIMS source files to Novell-server
;   may occasionally update to NT-server
;   using command-line parameter L=
ExitProgram    = c:\fp25\foxprox.exe -t DEMO.app
ExitFolder     = c:\pims
FileSpec       = *.PAS
Backup         = BAK,TMP
Compare        = DateSize
Missingfile    = Ask
FileSpec       = *.DBF
Backup         = DB1,DB2,DB3
MissingFile    = Create
SessionFiles   = A*.*
RightRoot      = C:
LeftRoot       = \\SERVER\SAMPLE, 20
RightPath      = \DEV\STK
LeftPath       = \SOURCE\DEV\STK
Exclude        = *.WRK, INDEX\*.IDX
UpdateMode     = Update
SubFolders     = Included
MissingFolder  = Ask
Execute
```



and here is the same sample script, commented line by line:

```
; Updating PIMS source files to Novell-server  
; may occasionally update to NT-server  
; using command-line parameter L=
```

These 3 lines are comment-lines, they are simply disregarded.

```
ExitProgram = c:\fp25\foxprox.exe -t DEMO.app
```

When all the script will have been executed, Update2 shall execute this complete command-line. The extension of the program to be executed must be provided (.exe here), since Update2 shall not attempt to provide any automatic extension.

```
ExitFolder = c:\pims
```

Before executing the program specified by ExitProgram, Update2 will make **C:\PIMS** to be the CWD, i.e. current working directory.

```
FileSpec = *.PAS
```

Update2 will look for a FileSpec-record for files which filename matches the \*.PAS specification. Since no such existing FileSpec-record shall be found, a new FileSpec-record will be created. The newly created FileSpec-record will be assigned default values.

```
Backup = BAK,TMP
```

When updated, \*.pas files shall be backed-up onto 2 generations, using extensions **BAK**, then **TMP**. This happens as follows: when **myfile.pas** is to be updated, any existing **myfile.tmp** gets deleted, then any existing **myfile.bak** gets renamed to **myfile.tmp**, then, depending on free space availability, one of the following may happen:

If enough free space is available, source **myfile.pas** gets copied as **\$update\$.\$\$\$** on target. Provided this copy is successful, target **myfile.pas** is then deleted, and **\$update\$.\$\$\$** is renamed as **myfile.pas** on target.

If we are short on free space, then **myfile.pas** gets copied directly from source to target, overwriting the initial one.

```
Compare = DateSize
```

Compare date & size, not contents. Files are claimed similar if they have same date/time and same size.

```
Missingfile = Ask
```

Ask user what to do if target file is missing: create target, delete source, or skip.

```
FileSpec = *.DBF
```

A FileSpec-record for \*.dbf files is created, with default values.

```
Backup = DB1,DB2,DB3
```

3-generations Backup will be performed on \*.dbf files.

```
MissingFile = Create
```

If target file \*.DBF is missing, create without asking.

```
SessionFiles = A*.*
```

This update-session will attempt updating all files that start with **A**.

```
RightRoot = C:
```

Right root is C: (priority 10) unless a command-line parameter **R=** (priority 100) is supplied.

## Update2 – by Profiles Software

version 2.8

page 14 / 15

[www.profiles-software.com](http://www.profiles-software.com)

support: [update2@profiles-software.com](mailto:update2@profiles-software.com)



PROFILES  
SOFTWARE

<b>LeftRoot = \\SERVER\SAMPLE, 20</b>	Left root is \\SERVER\SAMPLE, with priority 20, unless <b>L=</b> provided on command-line
<b>RightPath = \DEV\STK</b>	Merging RightRoot and RightPath indicates that the “Right” files are located in <b>C:\DEV\STK</b>
<b>LeftPath = \SOURCE\DEV\STK</b>	Merging LeftRoot and LeftPath indicates that the “Left” files are located in <b>F:\SAMPLE\SOURCE\DEV\STK</b>
<b>Exclude = *.WRK, INDEX\*.IDX</b>	Exclusions are all <b>*.WRK</b> files, as well as <b>*.IDX</b> files in the <b>INDEX</b> folder or any of its subfolders. <b>*.BAK</b> , <b>*.TMP</b> , <b>*.DB1</b> , <b>*.DB2</b> and <b>*.DB3</b> files are implicitly excluded, since these file-extensions are used by backup files.
<b>UpdateMode = Update</b>	Bi-directional updating. Newer files on Left overwrite older files with matching names on Right. Similarly, and in a second pass, newer files on Right overwrite older files with matching names on Left.
<b>SubFolders = Included</b>	When executing Right-To-Left, subfolders of <b>C:\DEV\STK</b> will be scanned too for files matching the <b>A*.*</b> (SessionFiles) pattern. Since <b>UpdateMode=Update</b> , when executing Left-To-Right, subfolders of <b>F:\SAMPLE\SOURCE\DEV\STK</b> will be also scanned.
<b>MissingFolder = Ask</b>	Ask if any target folder is missing, ask if the target folder should be created, or if the source folder should be disregarded, or deleted.
<b>Execute</b>	Execute this Session now. Since this is the last action in the script, after it is performed, <b>Update2</b> sets the current working directory as per <b>ExitFolder</b> , then launches the <b>ExitProgram</b> . <b>Update2</b> itself terminates its execution, and its icon disappears from the Windows desktop.

