
Using Editlists



R I M A G E™

Corporate Headquarters:

Rimage Corporation
7725 Washington Avenue South
Minneapolis, MN 55439
USA
800-553-8312 (toll free US)
Service: +1 952-946-0004 (International)
Fax: +1 952-944-6956

European Headquarters:

Rimage Europe GmbH
Albert-Einstein-Str. 26
63128 Dietzenbach Germany
Tel: +49-(0) 6074-8521-0
Fax: +49-(0) 6074-8521-21

Rimage Corporation reserves the right to make improvements to the equipment and software described in this document at any time without any prior notice. Rimage Corporation reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Rimage Corporation to notify any person or organization of such revisions or changes.

This document may contain links to web sites that were current at the time of publication, but may have moved or become inactive since. This document may contain links to sites on the Internet that are owned and operated by third parties. Rimage Corporation is not responsible for the content of any such third-party site.

©2007, Rimage Corporation

Rimage™ is a trademark of the Rimage Corporation. SDK™ is a trademark of the Rimage Corporation. Dell® is registered trademark of Dell Computer Corporation. FireWire® is a registered trademark of Apple Computer, Inc.

All other trademarks and registered trademarks are the property of their respective owners.

Contents

Introduction	1
Version 2 Editlist Extensions	1
Comments	1
File Name Quoting	1
UNC Filename Support	1
Wildcard Support	2
File Renaming	2
Defining a Version 2 Editlist	3
Base Path	3
CD Path (Subdirectories)	3
Files	3
File Source Override	4
Combining CD Path and Filenames	4
Source to Destination File Rename	4
Version 2+ Extensions	5
Regular Expression File Matching	7
Rules Governing Filters	7
Editlist Example	7
Regular Expression Matches	8
Example Version 1 (Old Style) Editlist	9
Example Version 2 Editlist	10
XML Editlist	11

Introduction

The imaging server can create images using either a parent directory or an editlist as the source for the files to be imaged. The editlist is a flexible way of describing the source and destination for the files that will comprise the image. The editlist is a line oriented ASCII file consisting of records separated by carriage return and linefeed (CR/LF). An editlist can be created using simple programming statements in a user's application program, or a manual editor program. The rules for records in the editlist file follow, and an example of an editlist is given to clarify usage. This document describes version 2 and 3 of the editlist which make a few changes and add some extensions to the format.

Version 2 Editlist Extensions

Comments

In Version 1 of the editlist any text within an edit file record enclosed in double quotes (") is treated as a comment. When a double quote is the only character on a line it is the beginning of a comment block that is ended by another line containing only a double quote.

For example: the following example does not match the previous sentence.

```
" This is a comment line"
or
D:/    " Base directory"
or
"
  Now we are going to do some really
  Good stuff to the program
"
```

Comments can appear anywhere in the file as long as they are placed at the end on any particular line.

In Version 2 of the editlist, the comment notation has changed because of the need to use the double quote to surround file names which contain spaces or tabs. In a Version 2 editlist, any line or part of a line which follows a semi-colon is considered a comment.

For example:

```
; This is a comment line
"D:\\" ; Base directory
```

Comments can appear anywhere in the file as long as they are the last thing on any particular line.

File Name Quoting

The file names in the Version 1 editlist were not quoted because in an 8.3 filename scheme blanks are invalid characters. Version 2 allows long filenames so the filenames, Base Paths, and CD Paths are surrounded by double quotes.

UNC Filename Support

Version 2.0 and later of the Editlist supports UNC (Universal Naming Convention) paths in any filename or source override path. UNC paths are recognized because they begin with the sequence '\\ or double backslashes.

For example:

```
"\\NETWORK\RESOURCE\Filepath"
```

Wildcard Support

The Version 2 editlist supports the use of file spec type wildcard characters in the filename tokens.

For example:

```
"*.DAT"           ; Wild card characters used  
or  
"G:\MYFILES\*.FIL" ; Wild card and renamed
```

File Renaming

The Version 2 editlist supports source to destination file renaming.

For example:

```
"MYDATA.TXT"      "G:\MYFILES\DATA.TXT"
```

File DATA.TXT from the source directory G:\MYFILES will be renamed to MYDATA.TXT in the current CD directory. The original file, the file on the fixed disk, is not renamed.

Defining a Version 2 Editlist

Base Path

A filename token beginning with a letter followed immediately by a colon ':', or a double slash (UNC Universal Naming Convention) and ending with either a colon ':' or a backslash '\' is assumed to specify a base drive and optional path in usual MS-DOS fashion (the colon must follow drive letters). These statements specify the drive and path that is the base or parent directory of the source of files to be placed on the CD-R. Examples of valid drive designators are:

```
"D:" ; uses the root of drive d as the file source.
"C:" ; uses the root of drive c as the file source.
"E:\WINDOWS\SYSTEM\" ; uses the windows\system subdirectory of drive e as
the file
; source.
"\\NET\RESOURCE\Filepath\" ; Unc Path
```

A new base directory can be started at anytime by simply specifying the new root in the file sequence.

CD Path (Subdirectories)

Lines that begin and end with a single backslash '\' specify subdirectories stemming from the base directory listed in the previous section and the destination directories of the CD-R. A subdirectory path can be specified in the usual MS-DOS way, but the string *must* be ended with another backslash. Examples are:

```
"\RECORDS\" ; indicates the list of files is to be read from the \records\
; subdirectory of the base directory (by default) and placed into the
; \records directory in the resulting image file.

"\COUNTY\COURT\1992\" ; indicates the list of files is to be read from the
\county\court\1992
; subdirectory of the base directory (by default) and placed into the
; \county\court\1992 subdirectory in the resulting image file.
```

 **Note:** A base source directory must be specified before the destination directory in the editlist file.

Files

Files to be copied from the source disk to the CD-R are identified on lines that are not Base Paths or CD Path (subdirectories) Files are sorted in ASCII collating order (without regard to case) while they are being copied to the CD-R.

 **Note:** Both a Base Source directory and the CD Path (destination subdirectory) must be specified before listing the files to copy to the CD-R image.

Files can be listed as either an absolute file name or a file spec including normal DOS wildcards.

For example:

```
"*.C" ; indicates all files with .C extension.
"MAIN.C" ; indicates only the main.c file.
```

Long file names are allowed in the Version 2 editlist. The previous version editlist did not have a facility to handle long file names and, in fact, inhibited their use by using the double quote character for comments. Long filenames are included similar to the following example:

```
"My Document.Doc" ; indicates the file 'My Document.Doc' on the long file name
system.
```

File Source Override

Each file is assumed to be in the same directory hierarchy in the resulting CD-R image as it was originally in the source. In a “parent directory” scheme this works quite well. Version 2 editlists can include file overrides if the source file does not exist in the parent directory hierarchy. When a file source override is used the entire path is specified for the filename.

For example:

```
"G:\DOCS\My Document.Doc"
or
"G:\DOCS\*.Doc"
or
"\\RIMAGE\NT_D\DOCS\*.Doc"
```

This feature allows files to be included, in the image, without the extra step of copying them to a parent directory before imaging. The file source override file path must be available to the imaging server software just like the base directory path and files.

Combining CD Path and Filenames

As a quick step you can put the name of a source directory on the same line as a CD Path. When you do this the Imager will change the CD Path and get the file or files listed in the source option.


For example:

```
"\DATA\" "G:\MYFILES\DATA\*.*"
"G:\MYFILES\DATA2\*.*"
```

is the same as:

```
"\DATA\"
"G:\MYFILES\DATA\*.*"
"G:\MYFILES\DATA2\*.*"
```

In this example, the program changes the CD Path to \DATA\ and gets all of the files from the G:\MYFILES\DATA directory into that subdirectory. Any subsequent files will also be copied to that directory until the CD Path is changed again.

 **Note:** While you can list a CD Path on every line, you really should only list the CD Path when it changes. This will speed up the imager slightly because it won't have to continually check to see if the directory exists.

Source to Destination File Rename

You can rename files as they are copied to the destination CD directory. To do so you format the line for the entry similar to the following:

```
"MYDATA.TXT" "G:\MYFILES\DATA\DATA.TXT"
```

When this line item is encountered by the processor the file from the source directory, "G:\MYFILES\DATA\DATA.TXT" is renamed to MYDATA.TXT in the current base path of the CD directory.

 **Note:** File renaming does not work with wild cards.

For example, the following lines:

```
"MYDATA.TXT" "G:\MYFILES\DATA\DATA.*"
"MYDATA.*" "G:\MYFILES\DATA\DATA.*"
```

will not work with this process. To rename files, each file must be listed directly.

Version 2+ Extensions

A number of extensions have been added to the editlist handling. These extensions are designed to handle the new options for VCD, the directing of files to either Mac or PC on a hybrid image, and a few other specialized operations. All of these options consist of the ':' character followed by a keyword. There may also be several additional arguments for some keywords.

All of these commands may be on a line by themselves. They take effect from that point on as the editlist is processed. The commands that are checked as **prefix** may also be used in front of a normal editlist file specifier. They will still affect later entries unless over-ridden, but their main use is to make the file more readable. As an example placing :MAC at the start of a line makes it clear that this file goes to the Mac side. Both QD and QDJ do this when creating editlists.

The following table lists the keywords and how they work.

Editlist Command	Prefix	Description	Comments
:PC	✓	Send file(s) to PC image.	The PC side is the ISO part of a hybrid image.
:MAC	✓	Send file(s) to Mac image.	The Mac side is the HFS part of a hybrid image.
:DEFAULT	✓	Send files to both Mac and PC.	Files will be in both the ISO and the HFS parts of a hybrid.
:BOTH	✓	Send files to both Mac and PC.	Same as above, :DEFAULT.
:UNC		Use UNC paths.	This is the default. The command is still accepted but has no effect. It was useful in the past when UNC's weren't always supported correctly.
:SUBDIRECTORIES		Expand folders.	Tells the editlist class to expand any folders found when using wildcards. This is set for the messaging version of IS, so it isn't useful anymore.
:EMPTYDIRECTORIES		Place empty folders on CD.	When set empty directories will be included on the CD image. This is set for the messaging version of IS and isn't useful anymore.
:INCLUDE_PARENT		Add parent folder path to CD.	This is a rarely used option that puts the source path minus the drive onto the CD path. In other words, the parent folder name is included on the CD. This is not the normal operation, where the parent folder does not appear on the CD.
:OPT ZIPFILENAME=filename		Changes the zip file name.	This is used to change the default name "comp.zip" to "filename" when using the zip option in IS.
FILES_AT_BASEPATH		Don't add current CD path to source path.	This is an extremely rarely used option that allows all files to be on the base source path rather than on a tree that is duplicated on the CD. In short, it collapses the source tree before putting the files on the CD.
:OPT ENCODING=UTF(8 16)		UTF encoding	The type of encoding is usually determined by the BOM (byte order mark) in the editlist. This command can force it to UTF16 or UTF8. It must be on the first line in the file.

Editlist Command	Prefix	Description	Comments
:OPT REGEX=(YES NO TRUE FALSE)		Allow regular expressions for filename matching	Choose T or Y to enable, or F or N to disable full regular expression matching in the INCLUDE and EXCLUDE options. N or F only allows DOS style wildcard matching in filenames.
:OPT INCLUDE=list		Include file specifiers	This is used to limit the files that are matched in the source files. It may be either DOS wildcards or regular expressions depending on the REGEX option above. Note that the source file must include a * for this to work. See the example later under regular expressions.
:OPT EXCLUDE=list		Exclude file specifiers	This can be used to specify files using DOS wildcards or regular expressions that are to be excluded from matching.
:DVD_SLIDE		Start a slide show entry	Image files after this in the editlist will be added to the DVD movie and will also be added to the file system.
:DVD_SLIDE_ONLYE		Start a slide show entry only	Image files after this will be added to the DVD movie but will not be added to the DVD file system. Note however that the DVD movie subsystem (Yes Video) may choose to add the original slides to the DVD file system anyway.
:DVD_SLIDE_END		Stop slide show entries	Files after this will not be added to the DVD movie. They will simply be placed on the output file system.
:DVD_VIDEO		Start adding DVD movie	Video clips after this will be added to the DVD movie. They will not be added to the file system.
:DVD_VIDEO_END		Stop adding DVD movie files	This returns the editlist to normal processing. Movie clips will be placed on the file system but will not be added to the DVD movie.
:OPT ROTATE=(0 90 180 270)		Rotate slides or movie clips	This sets the rotation of the following slides or movie clips. Use :OPT ROTATE=0 to set back to no rotation.
:VOLUME_GROUP_START		Start a spanning group	Files following this will be kept on the same volume in a spanned volume set if there is enough room on the media. If not, then more volumes as necessary will be used.
:VOLUME_GROUP_END		Finish a spanning group	This closes a volume group. Files that are not part of a volume group will not be placed on any media that has files from a volume group. In other words, each volume group gets its own set of media and files not belonging to a volume group will not be placed on any volumes that have volume group files.

Regular Expression File Matching

The :OPT REGEX can be used to control what kind of wild card matching is done. If it is set to FALSE, then DOS wildcards like ? and * are used. If it is set to TRUE, then full regular expressions are matched as described below. This is useful when using the INCLUDE and EXCLUDE filter options.

A filter specifies which content should be retrieved from a folder in cases that the user desires that only a subset of all content in a folder is desired for inclusion on the resulting disc.

For example, the user chooses a folder named "C:\MyFolder" for inclusion on a CD and this folder contains the following files:

- File1.jpg
- File2.jpg
- File3.cpp
- File4.cpp
- File5.h

Perhaps the user only wants to retrieve files with the ".cpp" extension since they are programming language code files that should be backed up to CD. A Filter allows the user to specify that only ".cpp" will be included on the CD from the folder in question. DOS *.cpp could be used for this.

Rules Governing Filters


- A filter is a string that follows the conventions of DOS wildcarding:
- Wildcards are characters that can be used to stand-in for unknown characters in file names. In card games, a wildcard is a card that can match up with any other cards. In DOS, wildcard characters can match up with any character that is allowable in a file name. There are two wildcards in DOS:
 - * = matches up with any combination of allowable characters. The asterisk character, *, can stand in for any number of characters. An example:
 - *.doc - This matches any file that ends with .doc.
 - ? = matches up with any single allowable character. The question mark wildcard, ?, stands in for any single character.
 - ab?.txt - This filter finds any file with a three-letter name, of which the first two letters are **ab**, and with a **txt** extension. So files like abz.txt and ab2.txt are included on the disc.
 - *ab?.do? - This looks in the folder for files that have anywhere from 1 to 5 beginning characters, followed by **ab** followed by one character, and which have an extension of do followed by any one character.

Editlist Example

```
:OPT INCLUDE="*.cpp|*.h|*.rc"  
"MyFiles\" "C:\MyFiles\*.*"
```

It is also possible to specify filter strings that exclude matches, meaning "I do NOT want files included that meet this criteria. For instance, the user can create the filter '*.doc' to exclude all doc files.

```
:OPT EXCLUDE="*.doc"  
"MyFiles\" "C:\MyFiles\*.*"
```

 **Note:** The second line in both examples must end on either *.* or *. This specifies all files. The files are then limited by the preceding INCLUDE or EXCLUDE lines. Setting INCLUDE or EXCLUDE to "" disables their action.

Regular Expression Matches

The filter strings can also use a subset of full regular expression pattern matching that is useful for more sophisticated matching criteria. This is a feature available for knowledgeable power users. It is only active when :OPT REGEX=T is earlier in the editlist.

The following table shows the regular expression metacharacters that are allowed and how they work.

Metacharacter	What it does
.	Matches any single character.
[]	Indicates a character class. Matches any character inside the brackets (for example, [abc] matches "a", "b", and "c").
^	If this metacharacter occurs at the start of a character class, it negates the character class. A negated character class matches any character except those inside the brackets (for example, [^abc] matches all characters except "a", "b", and "c"). If ^ is at the beginning of the regular expression, it matches the beginning of the input (for example, ^[abc] will only match input that begins with "a", "b", or "c").
-	In a character class, indicates a range of characters (for example, [0-9] matches any of the digits "0" through "9").
?	Indicates that the preceding expression is optional: it matches once or not at all (for example, [0-9][0-9]? matches "2" and "12").
+	Indicates that the preceding expression matches one or more times (for example, [0-9]+ matches "1", "13", "666", and so on).
*	Indicates that the preceding expression matches zero or more times
??, +?, *?	Non-greedy versions of ?, +, and *. These match as little as possible, unlike the greedy versions which match as much as possible. Example: given the input "<abc><def>", <.*?> matches "<abc>" while <.*> matches "<abc><def>".
\$	At the end of a regular expression, this character matches the end of the input. Example: [0-9]\$ matches a digit at the end of the input.
	Alternation operator: separates two expressions, exactly one of which matches (for example, "T the" matches "The" or "the").
!	Negation operator: the expression following ! does not match the input. Example: a!b matches "a" not followed by "b".
\	Escape character: interpret the next character literally (for example, [0-9]+ matches one or more digits, but [0-9]\+ matches a digit followed by a plus character). Also used for abbreviations (such as \a for any alphanumeric character; see following entries for specifics).
\a	Any alphanumeric character: ([a-zA-Z0-9])
\b	White space (blank): ([\t])
\c	Any alphabetic character: ([a-zA-Z])
\d	Any decimal digit: ([0-9])
\h	Any hexadecimal digit: ([0-9a-fA-F])
\n	Newline: (\r \r?\n)
\q	A quoted string: (^"[^"]*"')(^'[^']*')
\w	A simple word: ([a-zA-Z]+)
\z	An integer: ([0-9]+)

Regular expression example using several meta characters.

```
^[a-h].*\c.*$
```

This expression matches all files that start with the letter 'a', or 'b', or 'c', or 'd', or 'e', or 'f', or 'g', or 'h' with 0 or more following characters and any file extension that starts with 'c'.

Example Version 1 (Old Style) Editlist

```

"This is a comment block that is totally ignored"

D:\CDPRO          "source for the files is set as subdir cdpro on drive d:"
                  "the files from d:\cdpro will go in the root of the CD-R"
CD1.ICO           "first file"
CD2.ICO
CDD521.SYS
HEXED.EXE
HEXED.PIF
CD_PROBE.EXE     "last file"
\ASPI4DOS\       "change the subdirectory on the source and CD-R"
UPGRADE.EXE     "first file from subdirectory \cdpro\aspi4dos\"
UPDATE.DOC
ASPI4DOS.SYS
ASPIDISK.SYS    "last file"
C:\              "source for the files is changed to the root of drive c"
\REC\ACCTS\     "the files from c:\rec\accts directory will be put subdir"
                  "\rec\accts on CD-R"
1991.DOC        "first file"
1992.DOC
1993.DOC        "last file"

```

Example Version 2 Editlist

```
; This is a comment line that is totally ignored

"D:\CDPRO\"      ; source for the files is set as subdir cdpro on drive d:
                  ; the files from d:\cdpro will go in the root of the CD-R
"\\"             ; CD Path starts in root
"CD1.ICO"        ; first file
"CD2.ICO"
"CDD521.SYS"
"HEXED.EXE"
"HEXED.PIF"
"CD_PROBE.EXE"   ; last file
"\ASPI4DOS\"     ; change the subdirectory on the source and CD-R
"G:\NEW PROGRAMS\PROGRAM UPGRADE.EXE"
                  ; file override
"\\SERVER\SYS\NEW PROGRAMS\OTHER PROGRAM.EXE"
                  ; file override with UNC
"G:\NEW PROGRAMS\*.DAT"
                  ; file overrides with wild cards
"*.DOC"          ; all .DOC files included
"ASPI4DOS.SYS"
"ASPIDISK.SYS"   ; last file
"C:\"            ; source for the files is changed to the root of drive c
"\DATA\"         ; change cd and base path to \DATA\ and get all files
from that subdirectory
"\REC\ACCTS\"    ; the files from c:\rec\accts directory will be put subdir
                  ; \rec\accts on CD-R
"MYDATA.TXT"     "G:\MYFILES\DATA\DATA.TXT" ; File DATA.TXT is renamed to
MYDATA.TXT
"1991.DOC"       ; first file"
"1992.DOC"
"1993.DOC"       ; last file"
```

XML Editlist

Starting in eIS version 8.0 a totally new format editlist is possible using XML. This improves consistency since the image orders are already in XML format.

The XML editlist DTD is:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Martin Nohr (Rimage)
-->
<!-- Rimage Corp 2007 Martin Nohr
      Initial version 8/9/07
      8/20/07 Nohr - Added Options Element and eliminated duplicate values for the
attributes in Options. They used to be in both SrcDst and DstGroup.
      8/23/07 Nohr - Added VolumeGroup option and also reorganized the options.
Filters was split out so they won't be reset if any other option is used.

      TODO: Should we add a tree flatten option? The old editlist had one, but it
is unknown if anybody ever used it.
      Should Mac|PC|both be allowed at the file level? It was in the old
editlist, but it mostly made the editlist bigger. It might not be useful.
-->
<!-- Note that Options may be placed anywhere in a list of BasePathGroup's and
SrcDst's. The options affect all following entries until changed by another
Options line. -->
<!ELEMENT EditList (BasePathGroup | SrcDst | Options | Filters)+>
<!ATTLIST EditList
      Version CDATA "1.0"
>
<!-- This element allows a list of source and destination pairs. Each one is a
fully qualified file specifier, i.e. path and name.
      If Dst ends in a \ then it is a folder and all files will be placed there in
the image. This allows wildcards in Src.
      If Dst is a full filepath then the file is placed at that location, no
wildcards are allowed, but files may be renamed.
      Examples: Src="c:\files\test.txt Dst="\abc\test.txt"
      Src="c:\files\*.txt Dst=""
-->
<!ELEMENT SrcDst EMPTY>
<!ATTLIST SrcDst
      Src CDATA #REQUIRED
      Dst CDATA #REQUIRED
>
<!ELEMENT BasePathGroup (DstGroup+)>
<!-- BasePath may be either drive letter based or UNC, but it must be a complete
path, examples are:
      c:\abc
      \\machine\driveC\abc
-->
<!ATTLIST BasePathGroup
      BasePath CDATA #REQUIRED
>
<!ELEMENT DstGroup (File*)>
<!-- This element is a list of files and where they go on the media.-->
<!ATTLIST DstGroup
      DstPath CDATA #REQUIRED
>
<!-- File may be either a complete path or just a file name.
      If a full pathname (ie path and filename), then it becomes the absolute path
to the source file, the destination is CDPATH\namepart, where namepart is the
filename only part of the source path.
      If a filename, then the source is BasePath\CDPATH\name and destination is
CDPATH\name.
      DOS Style wildcards are allowed in the filename. Example: *.jpg.
      If no File elements are specified then *.* will be used. This may be
combined with regular expressions to match files.
-->
<!ELEMENT File EMPTY>
<!ATTLIST File
      Name CDATA #REQUIRED
>
```

```

<!-- The HybridPart specifies which partition the files will be placed on for
PC/Mac hybrid type discs.
PicsPlaySlide : false means it isn't a slide or movie, true means it is.
both means it is a slide/movie but also put it on the disc as a file.
PicsPlayVideo: true for video, else false, RotateVideo can optionally rotate
the video
ExpandFolders causes the folder tree to be copied to the destination. If
false only the current level is used.
All folders are used, the Filters are ignored for folder names, but are used
for files in subfolders.
For SrcDst the ExpandFolders option only works when Dst ends in a \. It will
then expand the tree from the Src level. The tree will be copied to the
destination.
Setting VolumeGroup to start will begin a new grouping of files. Grouping
files attempts to keep them together during spanning. Use "end" to finish a
group.
There is no limit to the number of groups. The first group will be process
first, the second next etc. After each group is finished (end) a new spanning
volume will be started even
if the current volume is not full.
-->
<!ELEMENT Options EMPTY>
<!ATTLIST Options
  HybridPart (PC | Mac | both) #IMPLIED
  PicsPlaySlide (true | false | both) #IMPLIED
  PicsPlayVideo (true | false) #IMPLIED
  RotateVideo (0 | 90 | 180 | 270) #IMPLIED
  ExpandFolders (true | false) #IMPLIED
  VolumeGroup (start | end) #IMPLIED
>
<!-- The Filters attributes are used in combination with wildcards in the
filename. They provide more precise selections of files.
RegExInclude is a regular expression to include files. RegExExclude is a
regular expression to exclude files.
Note that RegEx filters only affect files, not folders.
The modified time format is CCYY-MM-DD HH:MM:SS
ModTimeBefore will only pass files modified before and including the
specified time/date.
ModTimeAfter will only pass files modified after and including the
specified time/date.
-->
<!ELEMENT Filters EMPTY>
<!ATTLIST Filters
  RegExInclude CDATA ""
  RegExExclude CDATA ""
  ModTimeBefore CDATA #IMPLIED
  ModTimeAfter CDATA #IMPLIED
>

```

The ModTimeBefore and ModTimeAfter are date and time selectors that can be used to limit file matches. The format is CCYY-MM-DD HH:MM:SS.

A sample SrcDst type is shown below. Note the Options element used to set the ExpandFolders attribute. This is a new feature that was not available with the old editlist format. This option affects SrcDst lines that follow it. As many Options elements as required may be used. The first SrcDst entry in this example uses the default values since the Options element is missing. The ExpandFolders attribute can significantly reduce the editlist size. This is a good thing since going to XML significantly increases the editlist size. See the next example.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE EditList SYSTEM "C:\SourceCode\VS8_8.0\XML\EditList_1.0.dtd">
<EditList Version="1.0">
  <SrcDst Src="\NEWDELL\TestSets\dir1\*.*" Dst="\dir1\"/>
  <Options ExpandFolders="true"/>
  <SrcDst Src="\NEWDELL\TestSets\dir2\*.txt" Dst="\dir2\"/>
</EditList>

```

This example shows how simple it can be to copy a tree into the image file system. Note that if *.* is not used, for example *.jpg, then only files matching that spec will be copied from all the folders referenced.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE EditList SYSTEM "C:\SourceCode\VS8_8.0\XML\EditList_1.0.dtd">
<EditList Version="1.0">
  <Options ExpandFolders="true"/>
  <SrcDst Src="c:\pictures\*.*" Dst="\"/>
</EditList>
```

A sample BasePathGroup example is shown here. This one includes jpg, tiff, and bmp files. Pay attention to the regular expressions. They are not DOS wildcards, so *.jpg does not do what you might think, in fact it is an illegal regular expression!

Like the old editlist the first DstPath will get files from c:\files\images and put them in \images while the second DstPath will get them from c:\files\images\morefiles and put them in \images\morefiles.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE EditList SYSTEM "C:\SourceCode\VS8_8.0\XML\EditList_1.0.dtd">
<EditList Version="1.0">
  <Filters RegExInclude=".+\.(jpg|tiff|bmp)"/>
  <BasePathGroup BasePath="C:\files">
    <DstGroup DstPath="\images">
      <File Name="*.*"/>
    </DstGroup>
    <DstGroup DstPath="\images\morefiles">
      <File Name="*.*"/>
    </DstGroup>
  </BasePathGroup>
</EditList>
```

Another example illustrating that the File Elements are optional. If no File elements are found then the filespec *.* will be used. This allows a simple form as shown in the following example. Note that this example also shows the ExpandFolders attribute that will recreate the tree on the media much like using the ParentFolder option in the Image Order. This example would place all of the files in c:\files on the media root folder. All the folders from c:\files and lower would also be created on the media.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE EditList SYSTEM "C:\SourceCode\VS8_8.0\XML\EditList_1.0.dtd">
<EditList Version="1.0">
  <Options ExpandFolders="true"/>
  <BasePathGroup BasePath="C:\files">
    <DstGroup DstPath="\"/>
  </BasePathGroup>
</EditList>
```

The grouping options for spanning work like this.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE EditList SYSTEM "C:\SourceCode\VS8_8.0\XML\EditList_1.0.dtd">
<EditList Version="1.0">
  <Options ExpandFolders="true"/>
  <Options VolumeGroup="start"/>
  <SrcDst Src="\\NEWDELL\TestSets\dir1\*.*" Dst="\dir1"/>
  <Options VolumeGroup="end"/>
  <SrcDst Src="\\NEWDELL\TestSets\dir2\*.*" Dst="\dir2"/>
</EditList>
```