

Link Shell Extension

Last Updated December 18 2022, Version 3.9.3.5

Privacy Statement The privacy statement can be found [here](#)

[Download](#)

[Documentation](#)

[Frequently asked questions \(FAQ\)](#)

[Blog](#)

[History](#)

[Donations](#)

Quick Start

 [French](#)

 [German](#)

The NTFS file system supports a facility known as **hard links** (referred to herein as **Hardlinks**). Hardlinks provide the ability to keep a single copy of a file yet have it appear in multiple folders (directories). They can be created with the POSIX command *ln* included in the Windows Resource Kit, the *fsutil* command utility included in Windows or my command line [ln.exe](#) utility. Thus, using standard Windows facilities Hardlinks can only be created at the command prompt, which can be tedious, especially when Hardlinks to multiple files are required or when one only makes occasional use of Hardlinks.

Support for Junctions in standard Microsoft software offerings is even more limited than that offered for hardlinks. Recently [Winfile.exe was github-reborn](#) by Microsoft, and I spent quite some time [to make it hardlink and symbolic link aware](#). For easy tasks you might try [Winfile](#)

Introduction

Link Shell Extension (LSE) provides for the creation of [Hardlinks](#), [Junctions](#), [Volume Mountpoints](#), and [Symbolic Links](#), (herein referred to collectively as Links) a folder [cloning process](#) that utilises Hardlinks or Symbolic Links and a copy process taking care of Junctions, Symbolic Links, and Hardlinks. LSE, as its name implies is implemented as a Shell extension and is accessed from Windows Explorer, or similar file/folder managers.

The extension allows the user to select one or many files or folders, then using the mouse, complete the creation of the required Links - Hardlinks, Junctions or Symbolic Links or in the case of folders to create Clones consisting of Hard or Symbolic Links. LSE is supported on all Windows versions that support NTFS, Windows 7/8/10. Hardlinks, Junctions and Symbolic Links are NOT supported on FAT file systems, and nor is the Cloning and Smart Copy process supported on FAT file systems.

Within this document the terms **action button** and **action (pop up) menu** are used to refer what are often referred to as the right mouse button and the pop up menu that is displayed when that mouse button is pressed (often referred to as the context menu). Recognising that people swap the usage of their mouse buttons, Microsoft refer to the *primary* and *secondary* mouse buttons. We prefer to refer the mouse buttons as the **Select** button and the **Action** button; and rather than terms such as Context Menu, Shell Menu, Right Mouse Menu we use the term **Action** menu.

The current user must have administrator privileges in order to install the software.

LSE is installed by executing the install program ([HardLinkShellExt_\\$\(platform\).exe](#)). Follow the instructions issued by the program, there are no mandatory inputs required during installation. It is possible to change the location into which LSE is installed, the default is

C:\Program Files\LinkShellExtension

SmartScreen pop-up

With Windows 10 the SmartScreen facility of Windows Defender might pop up and warn you:

Installation

Windows protected your PC

Windows Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

Please choose 'Install Anyway'. LSE is signed with a Standard Code Signing Certificate, but not with an [EV Code Signing certificate](#), which would prevent SmartScreen from interfering.

Explorer Restart

During installation Explorer.exe has to be restarted to make Link Shell Extension active. This means that all pending operations with explorer.exe are interrupted but with the interactive install you can decide to postpone the explorer.exe restart. A dialog box will give you the choices during installation.

Installation Arguments

Some command line switches can be passed during install, so that a silent install via .bat file can be achieved.

Silent install

/S pops up no prompts during install. If the /S switch is used, explorer.exe will be restarted after installation, to make Link Shell Extension active immediately.

Specify Language

/LANGUAGE passes the language in which LSE shows up. e.g.

HardLinkShellExt_\$(platform).exe /S /Language=English Currently *English, Chinese, Czech, French, German, Greek, Italian, Japanese, Korean, Polish, Portuguese, Russian, Slovak, Spanish, Swedish, Turkish and Ukrainian* are available as valid parameters for the /Language switch.

Specify Directory

When using the silent install a directory can also be specified with the /D switch e.g.

HardLinkShellExt_\$(platform).exe /S /Language=English /D=C:\Program Files\LSE

Uninstall

Link Shell Extension can also be uninstalled silently by issuing

\$LSEInstallDir/uninst-HardLinkShellExt_\$(platform).exe /S

If the /S switch is used during uninstall, explorer.exe will be restarted after uninstallation, to make Link Shell Extension inactive immediately.

No Check for VcRedist

On some Windows10 machines, there is no need to install the vc_redist, or worse: VcRedist can not be installed, but is somehow already on the system. To overcome this the /noredist switch can be passed via command line. e.g.:

HardLinkShellExt_\$(platform).exe /noredist

Link ShellExtension can also be installed via [chocolatey](#) by issuing

choco install linkshellextension

from a command prompt. Make sure you have [chocolatey installed](#). The current user must have administrator privileges in order to install the software via choco

LSE can also be installed via [winget](#) by issuing

winget install HermannSchinagl.LinkShellExtension

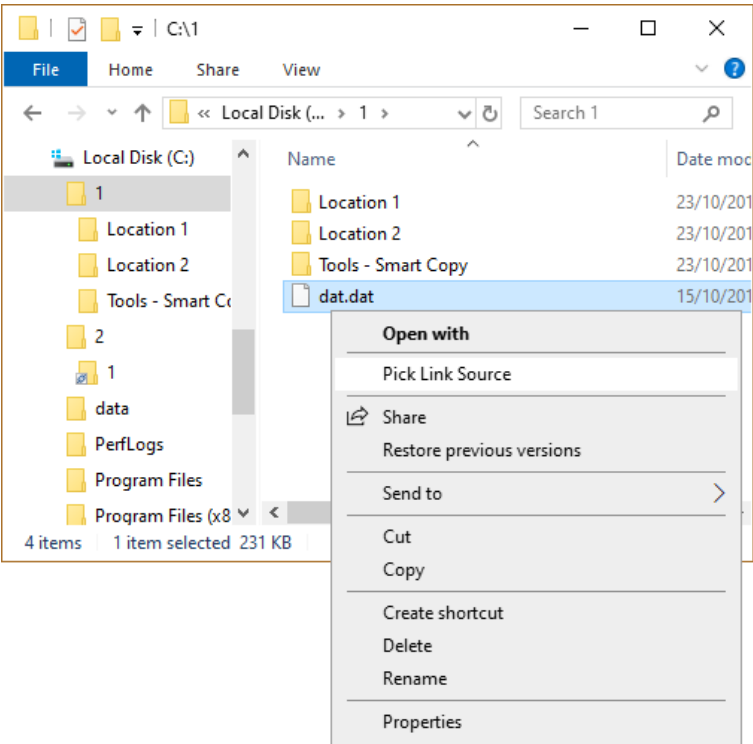
from a command prompt. Make sure you have [winget installed](#).

Chocolatey Installation

Winget Installation

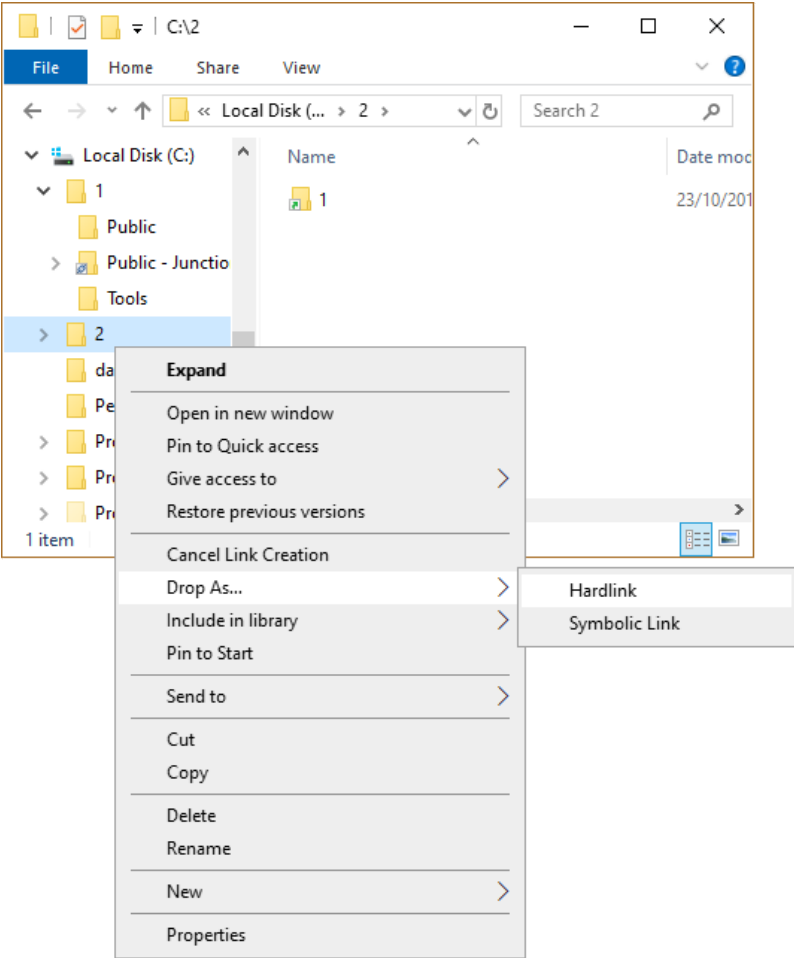
Using Link Shell Extension

Pick Link Source causes the selected files to be "stored" as the source for the Hardlinks that you want to create.



To create the Hardlinks a destination folder must be chosen, by clicking the mouse action button on the destination folder a

menu will pop up, which will include the entry - **Drop HardLink**



Choosing **Drop HardLink** will create the hardlinks in the selected destination folder.

Overlay Icons for Hardlinks

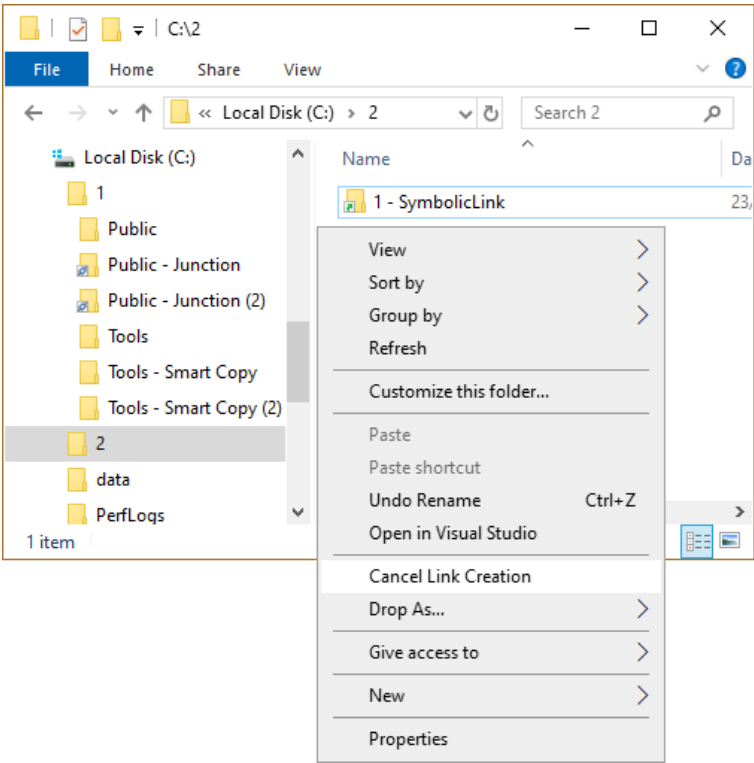
To help distinguish hardlinks folders from normal files, an **overlay icon** is implemented on hardlinks that shows a red arrow icon under the folder.



Overlay icons for Hardlinks can also be [customized](#).

**Cancel current
Pick Link
operation**

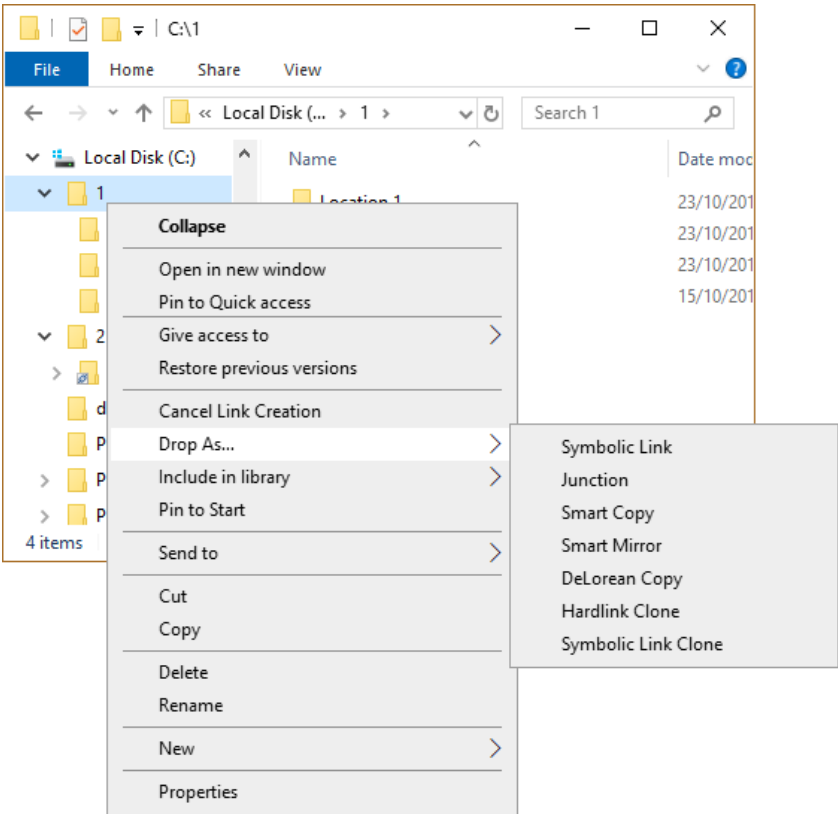
When doing an Action button click in the destination folder background, in addition to the Drop HardLink option there is the possibility to **Cancel Link Creation** entry.



Since LSE supports [Junction](#), [Clones](#) and [Symbolic Links](#), when one or more folders are selected as the Source, they can be dropped in several ways.

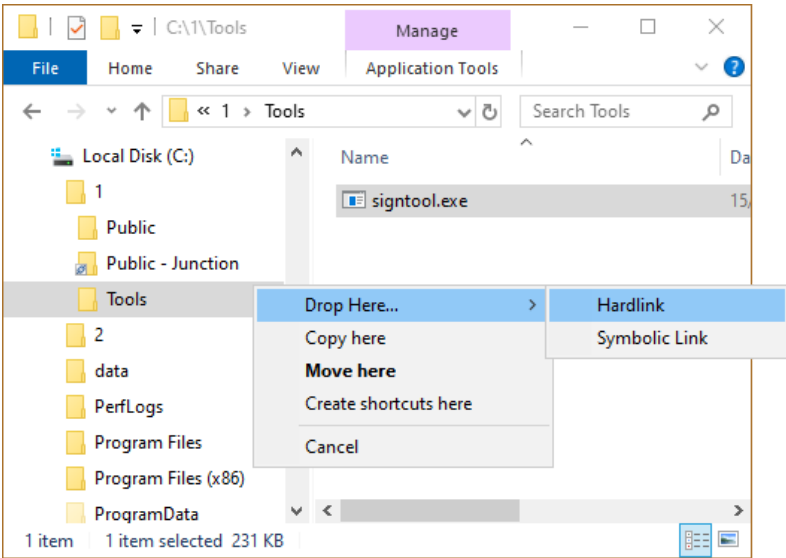
To avoid crowding the popup menu, a submenu is provided that contains the different types of Links applicable to folders.

Popup Submenu

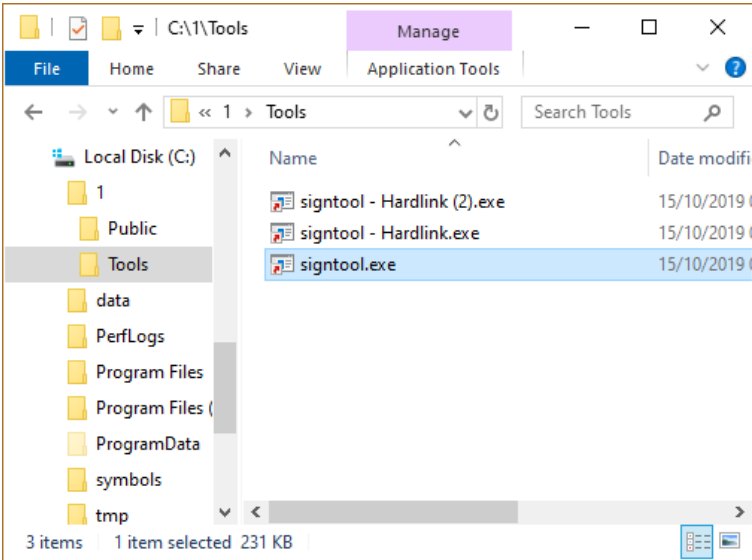


Drag and Drop Support

Creating Hardlinks via drag and drop is supported, after selecting one or more files you can drag them to the destination folder with the Action button held down; when it is released choose **HardLink Here** from the action menu to create the Hardlinks of the selected files in the destination folder.

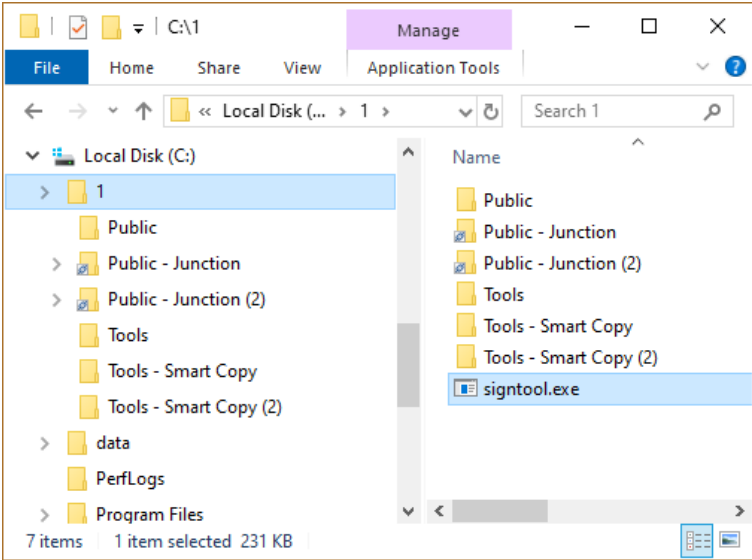


Files can be hard linked to the same folder as the source folder. Because two directory entries cannot have the same name, LSE uses '\$filename - Hardlink.\$ext' as the name of the the new link.

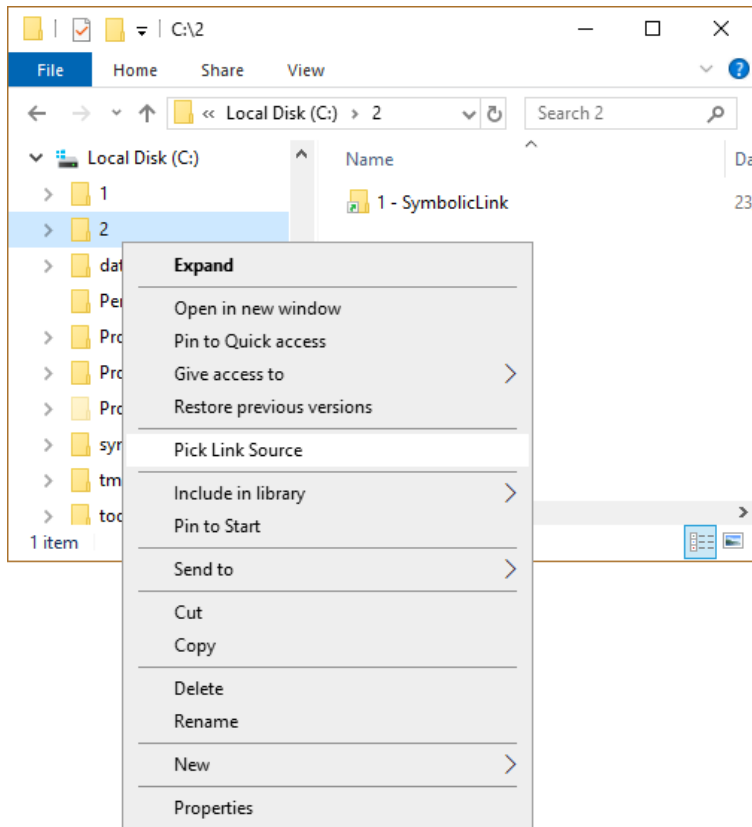


Auto Rename

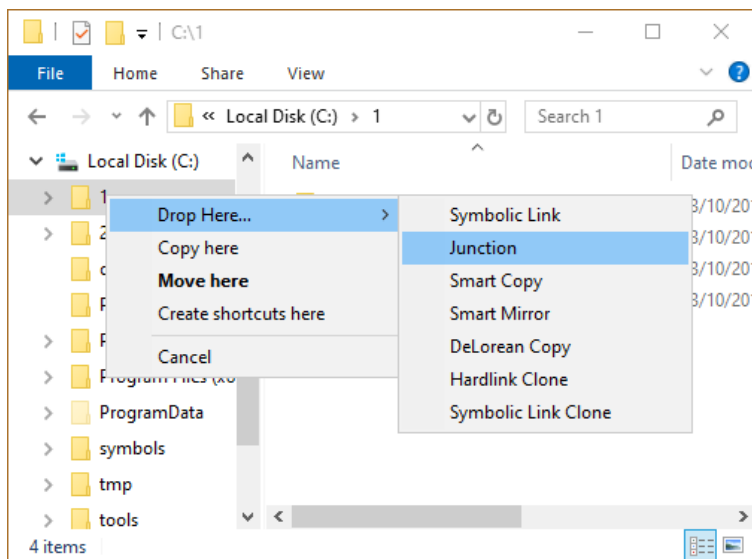
LSE uses the same hydraulics as explorer when it comes to multiple '\$filename - Hardlink': It uses numbers to enumerate the multiple hardlinks of one file in the same directory, e.g. *\$filename - Hardlink (2).\$ext*.
The Auto Rename mechanism is also used when Junctions, Hardlink Clones, Symbolic Links, Symbolic Link Clones, Mountpoints or Smart Copies are created in the same directory.



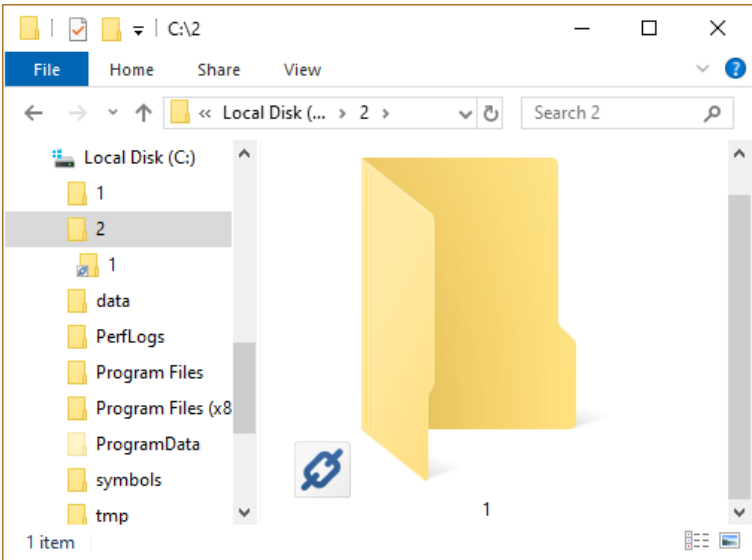
Junction Support [Junctions](#) provide for the creation of linkages among directories.



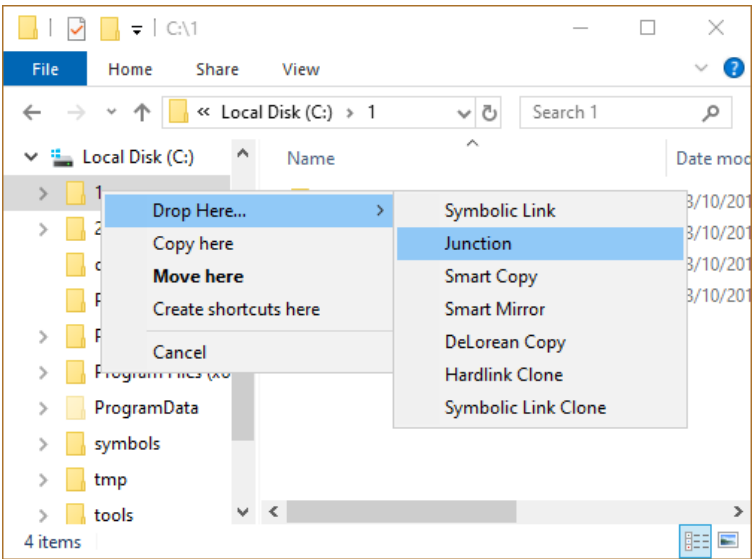
Junctions are created in the same way as Hardlinks, except that the Source Link is a folder rather than a file. Select a folder, click the right mouse button, choose **Pick Link Source** from the action menu, navigate to the destination folder, click the action button, open the submenu **Drop As ...** and select **Junction**:



Junctions are marked with a small piece of chain below the folder icon.



Junctions can also be created via Drag and Drop when the selected folders are dragged with the action button pressed to a destination folder; when the right mouse button is released, select the **Drop Here ...** submenu and then **Junction**.



Overlay Icons for Junctions

To help distinguish junction folders from normal folders, an **overlay icon** is implemented on junctions that shows a small three link chain icon under the folder.



Overlay icons for junctions can also be [customized](#).

Junctions can **span network drives** as long as the target is a mapped network drive. Unfortunately Junctions, which have a UNC Path as target, can be created with LSE, but even Windows 7/8/10 seems to contain a bug, which prevents it from dereferencing a UNC Path in a junction, even if LSE correctly sets up the reparse info for UNC junctions. When a UNC target junction is double clicked in explorer the error `ERROR_INVALID_REPARSE_DATA(4392)`, will show up and tell you that the info in the reparse point is illegal, even if it is not.

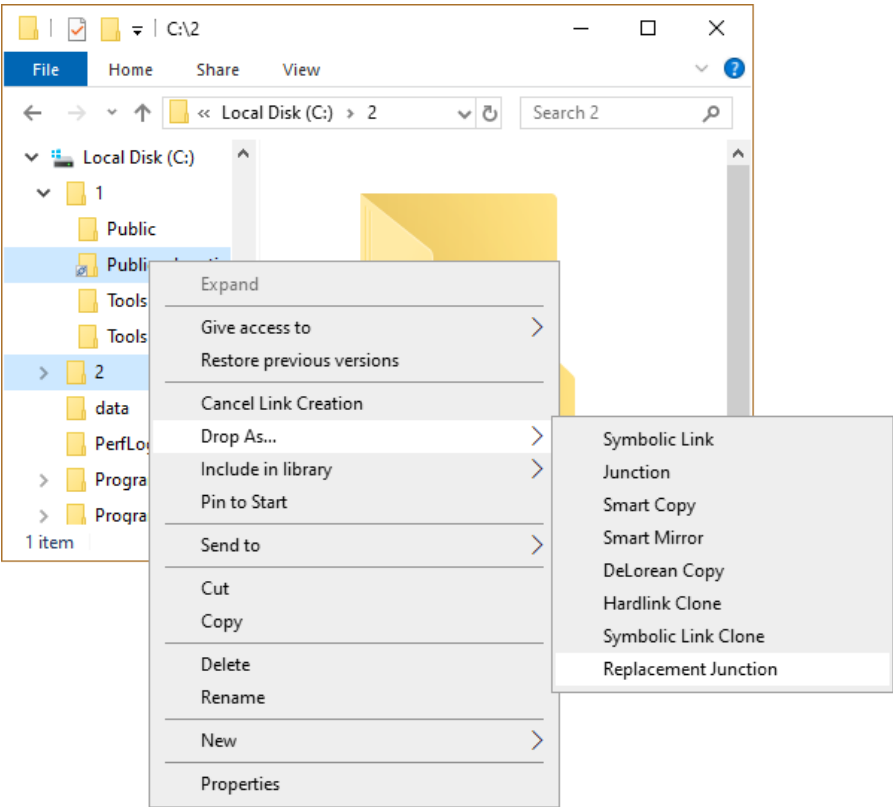
Elevation is needed in e.g. `c:\Program Files` for junctions beeing created in. This is why the [famous UAC](#) dialog must be acknowledged.

To be exact: Only the creation of directories needs elevation in such situations, but creating an empty directory is a vital part of creating a junction. The `DeviceIoControl()`, which does the real work in creating junctions would work without elevation.

Replacement Junction Symbolic Link Mountpoint

Link Shell Extension can change the target of an existing Junction, Symbolic Link or MountPoint either via Pick/Drop or Drag and Drop.

To use this feature simply select an existing directory as Link Source and drop it over an already existing Junction/Symbolic Link/Mountpoint. By selecting the 'Drop as ... Replacement Junction/Symbolic Link/MountPoint' from the action menu, the target of an already existing Junction/Symbolic Link/MountPoint is replaced by the newly picked target.

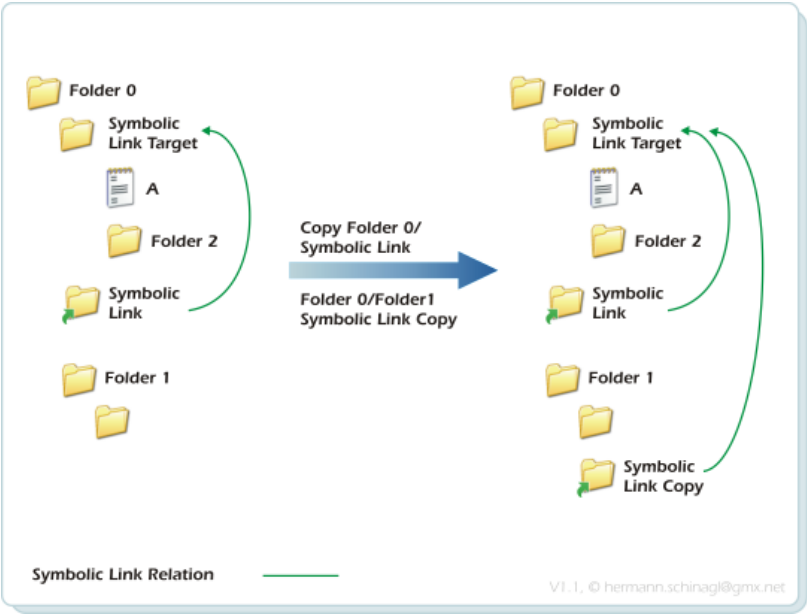


The same can be achieved via Drag and Drop for Symbolic Link Directories, Junctions and Mount Points, but not for Symbolic Link Files. Symbolic Link Files have to be repaired by the 'Pick Link Source', 'Drop as ... Replacement Symbolic Link' repair alternative.

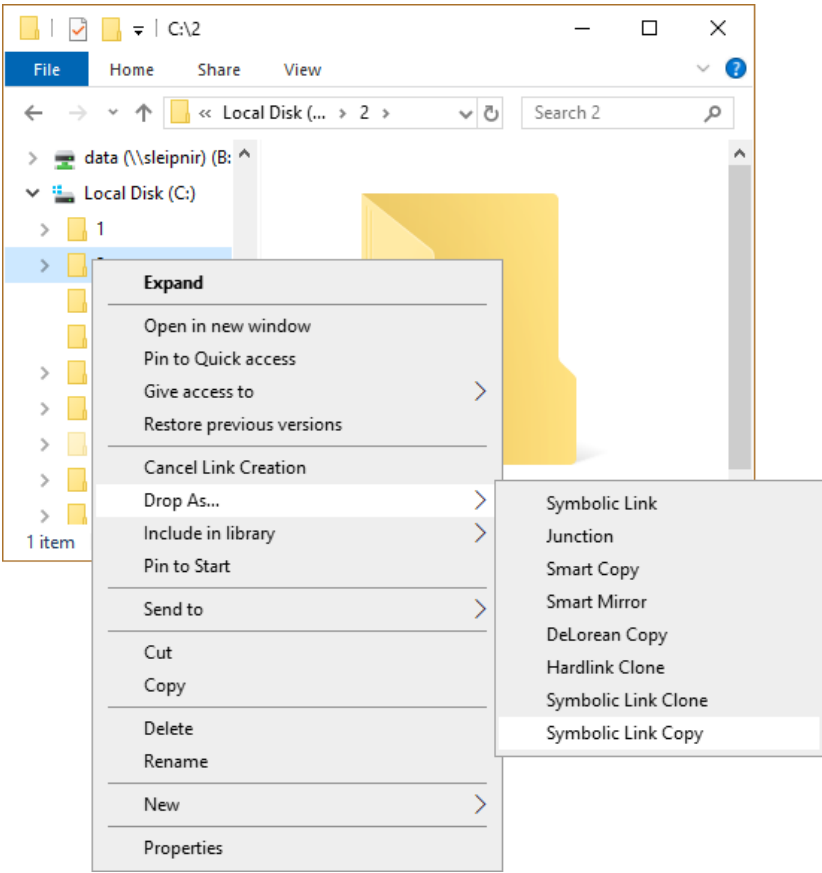
When the [backup mode](#) is selected the ACLs of the Junction/Symbolic Link/MountPoint are preserved.

**Copy Junction
Symbolic Link
Mountpoint**

Link Shell Extension can copy an existing Junction, Symbolic Link or MountPoint either via Pick/Drop or Drag and Drop.



To use this feature simply select an existing Junction, Symbolic Link or MountPoint as Link Source and drop it over an already existing directory. By selecting the 'Drop as ... Junction/Symbolic Link/MountPoint Copy' from the action menu, the Junction/Symbolic Link/MountPoint is copied to the target and the relation is adjusted



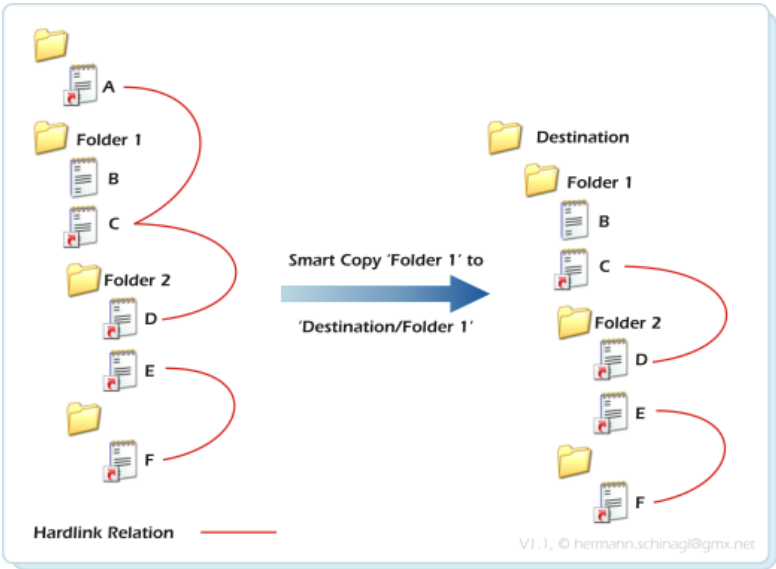
The same can be achieved via Drag and Drop for Symbolic Link Directories, Junctions and Mount Points.

When the [backup mode](#) is selected the ACLs of the Junction/Symbolic Link/MountPoint are preserved.

Smart Copy

Smart Copy creates a copy of the directory structure from the source location to the destination, but it preserves the inner hardlink structure and inner junction/symbolic link relations of the source, and recreates this inner hardlink structure and inner junction/symbolic link relation at the destination location:

With hardlinks it behaves as follows:



By closely looking at the above picture one can find three different types of files:

- Normal Files** The file B is a normal file. It gets copied as any other copy tool would do.
- Saturated Hardlinks** The files E and F are hardlinked together. In LSE terms they are called *Saturated Hardlinks*, because the reference count, which is here 2, matches the number of occurrences below 'Folder 1', which is here 2.

In General: A hardlink is called *Saturated* with respect to a folder *F*, if the number of occurrences below the folder *F* matches the reference count.

Saturated Hardlinks can be copied completely via Smart Copy.

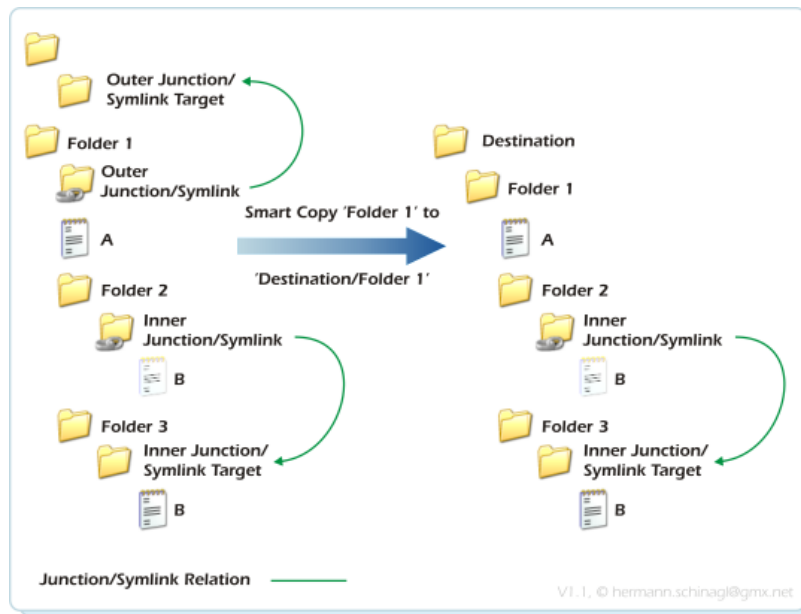
The File A, C, D are hardlinked together. In LSE terms they are called *Unsaturated Hardlinks*, because the reference count, which is here 3, does not match the number of occurrences below 'Folder 1', which is here 2. Only C and D are below Folder 1.

Unsaturated Hardlinks

In General: A hardlink is called *Unsaturated* with respect to a folder *F*, if the number of occurrences below the folder *F* is smaller than the reference count.

Unsaturated Hardlinks can only be partially copied by Smart Copy. In the above example C and D are hardlinked together in the destination location, but the hardlink to A is broken. This means that the reference count of C and D is 2 with the destination location.

With junctions or symbolic link directories [the default behaviour](#) during smartcopy is as follows:



By closely looking at the above picture one can find three different types of folders/junctions:

Normal Folders

The folder 'Folder 3' is a normal folder. It gets copied with its content as any other copy tool would do.

Inner Junctions Symlinks

The folder 'Inner Junction/Symlink' is targeted at 'Inner Junction/Symlink Target'. In LSE terms this kind of folder is called *Inner Junction/Symlink*, because its target points to a folder, which is below the common anchor 'Folder 1'.

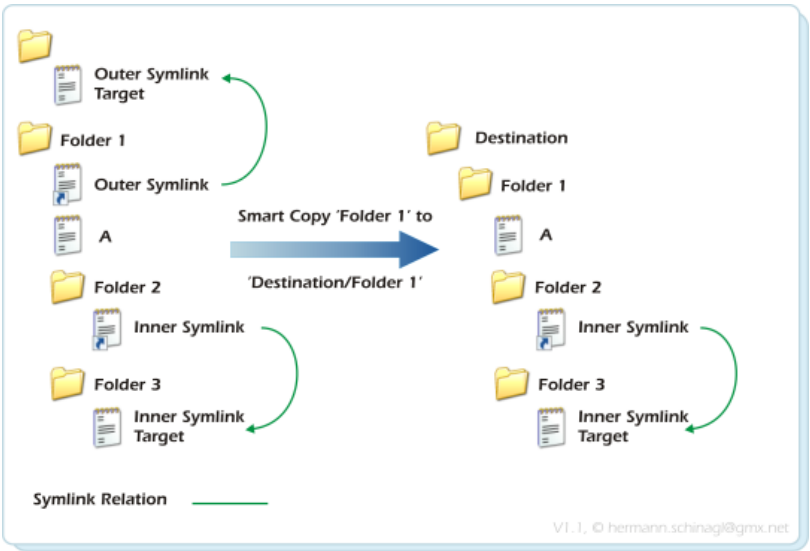
Inner Junctions/Symlink are restored properly via Smart Copy in the destination location.

Outer Junctions Symlinks

The folder 'Outer Junction/Symlink' is targeted at the folder 'Outer Junction/Symlink Target'. In LSE terms this kind of folder is called *Outer Junction/Symlink*, because its target points to a folder, which is in parallel and thus outside the anchor 'Folder 1'.

Outer Junctions/Symlink can be handled in three different ways. Please see the section on [Outer Junction/Symlink Handling](#).

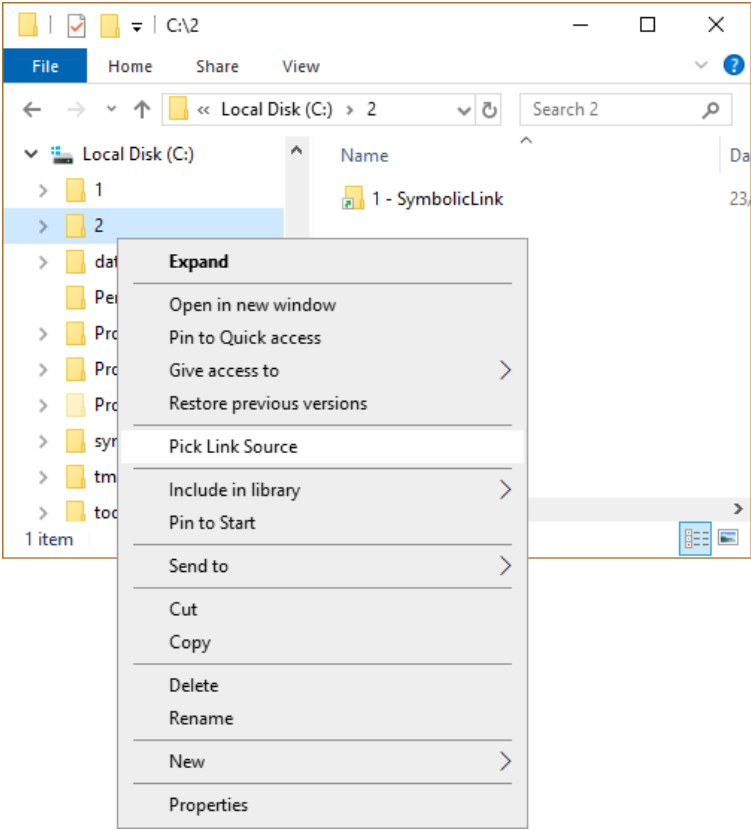
Windows 7/8/10 support Symbolic Links, which behave as follows during Smart Copy:



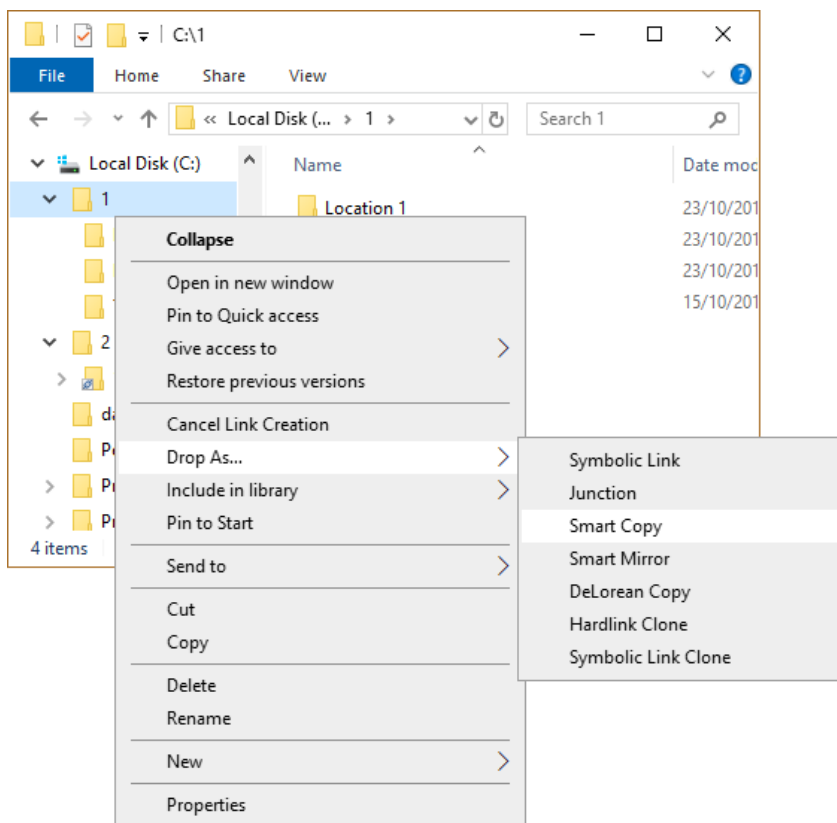
By closely looking at the above picture one can find three different types of files/symbolic links:

- Normal Files** The file A is a normal file. It gets copied as any other copy tool would do.
- Inner Symbolic Links** The symbolic link 'Inner Symlink' is targeted at 'Inner Symlink Target'. In LSE terms this kind of symbolic link is called *Inner Symlink*, because its target points to a file, which is below the common anchor 'Folder 1'. *Inner Symlink* are restored properly via Smart Copy at the destination location.
- Outer Symbolic Links** The symlink 'Outer Symlink' is targeted at the folder 'Outer Symlink Target'. In LSE terms this kind of symbolic link is called *Outer Symlink*, because its target points to a file, which is in parallel and thus outside the anchor 'Folder 1'.
Outer Symlink are handled by Smart Copy depending on the [Outer Junction/Symbolic Link](#) handling.

Smart Copies are created in the same way as Junctions, select a folder, click the Action button, choose **Pick Link Source** from the action menu...



...navigate to the destination folder, press the action button, open the **Drop As ...** submenu and select **Smart Copy**:



Smart Copy is a must if e.g.. the whole content of a hard disk, which has lots of hardlinks/junctions/symbolic links, should be copied to another hard disk. During the Smart Copy operation empty folders get copied too and the date/time stamps of folders/junctions/symbolic links are also restored at the corresponding destination locations.

Because Smart Copy creates inner hardlinks/junctions/symbolic links, this feature is only available on NTFS volumes.

If Smart Copy takes longer than 250msec a progressbar shows the status of the smart copy operation.

Smart Copy also processes all available alternative NTFS streams of a file.

If items are already available in the destination, Smart Copy only copies when the Files/Symbolic/Junctions/MountPoints are newer than the already existing items.

When restoring Symbolic links LSE forks its helper LSEUacHelper.exe to forwards this operation to it, because the creation of symbolic links needs elevation, and thus brings up the [famous UAC](#) dialog.

LSE **only** issues its helper LSEUacHelper.exe if a symlink is among/below the selected folders, so it saves you from one UAC prompt if you don't have symlinks among your selection.

Smart Copy by default creates [relative](#) symbolic links during the Smart Copy operation.

Command Line

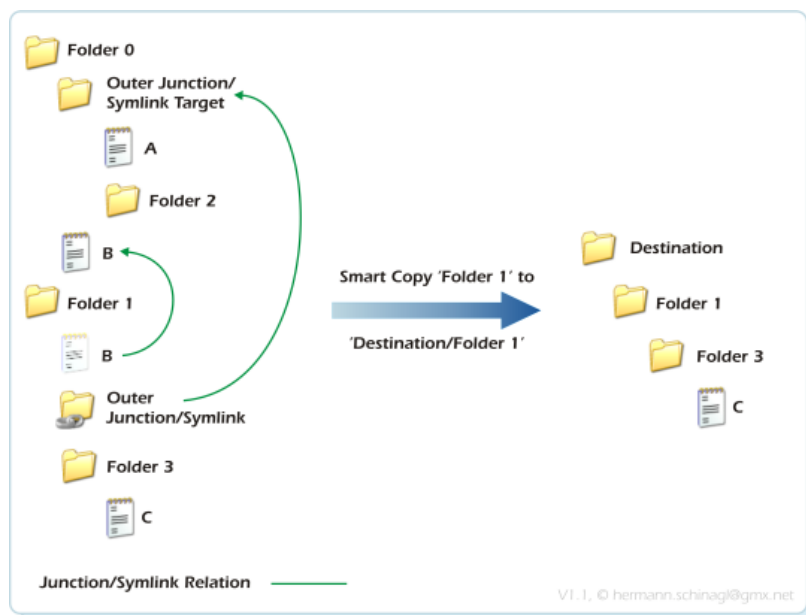
The Smart Copy functionality is also available via command line from [n.exe](#) via the --copy command line switch.

Crop/Unroll/Splice Outer Junctions/ Symbolic Links

During [SmartCopy](#), [Smart Mirror](#), [DeLorean Copy](#) and [Clone](#) so called [Outer Junctions/Symlink directories](#) may need processing. There are 3 different ways to deal with those Outer Junctions/Symlink directories:

Crop *Crop* breaks links to Outer Junctions/Symlink directories in the destination.

Crop also applies to Outer Symlink Files.



In the above example *Folder1* is copied to *Destination/Folder1*, but *Outer Junction/Symlink* is not available in the destination, because *Folder1/Outer Junction/Symlink* pointed to *Folder0/Outer Junction/Symlink Target*, which is not below *Folder1*.

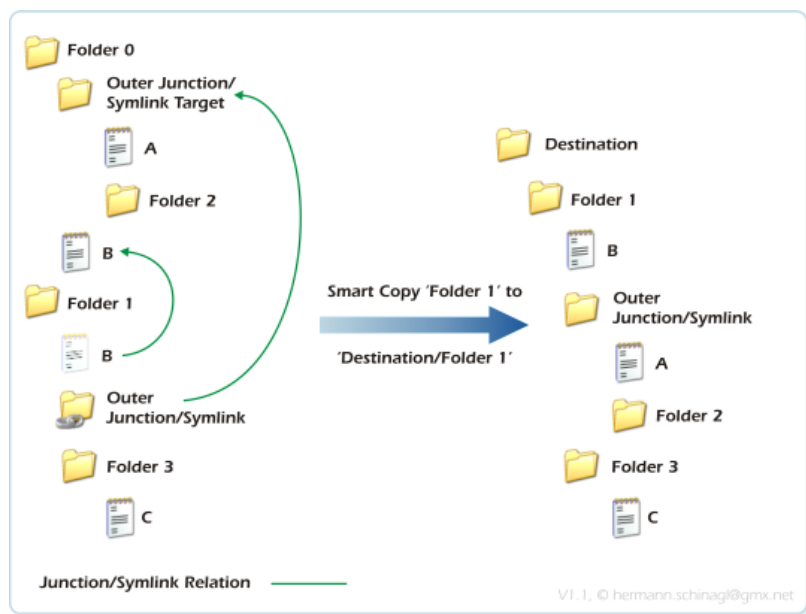
The objective behind cropping Outer Junctions/Symlink Directories is to get a pure copy during Smart Copy, Smart Mirror, Delorean Copy and Clone without connections to the source.

Enabling Crop for Outer Junction/Symbolic Links

Crop can be selected via the [configuration tool](#).

Unroll *Unroll* follows Outer Junctions/Symlink Directories and rebuilds the content of Outer Junctions/Symlink Directories inside the hierarchy at the destination location.

Unroll also applies to Outer Symlink Files, which means, that unroll causes the target of Outer Symlink Files to be copied to the destination location.



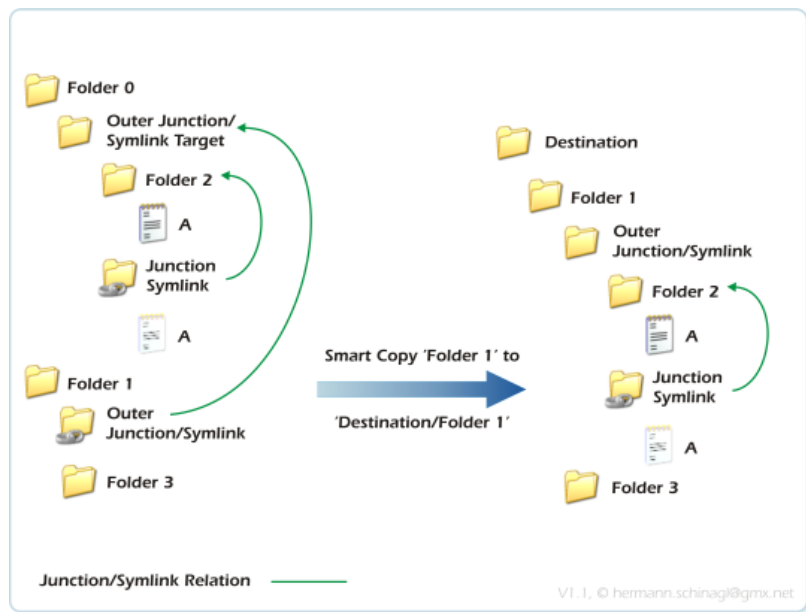
In the above example *Folder1* is copied to *Destination/Folder1*, and *Outer Junction/Symlink* and all the files/directories below *Outer Junction/Symlink Target* are copied to the folder *Outer Junction/Symlink* in the destination.

The objective behind unrolling Outer Junctions/Symlink Directories is to get everything with which the source is connected and rebuild it as separate copy in the destination. It resembles the 'hair of the elephant' pattern: Pull on a hair of an elephant, and get the whole elephant.

Unroll is the default behaviour for Smart Copy, Smart Mirror, Delorean Copy and Clone.

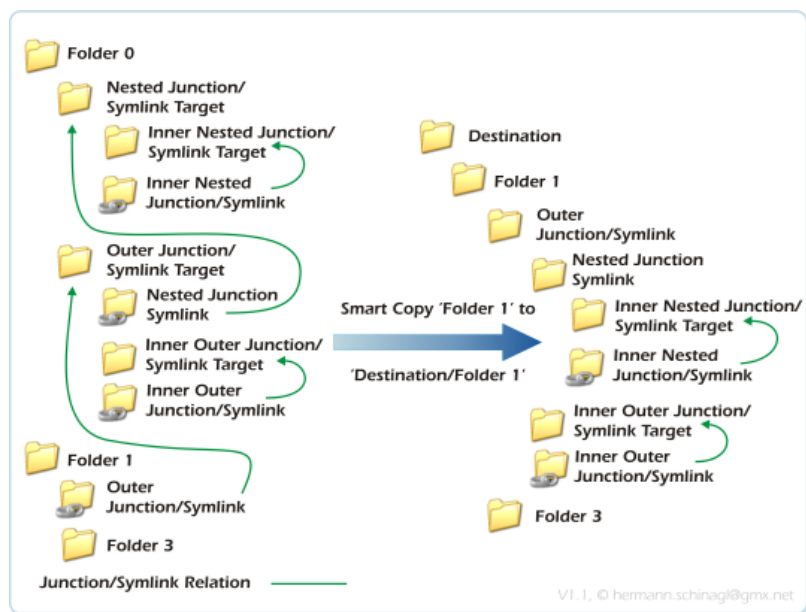
Advanced thoughts on Unrolling

The picture above was just the simplest case, because *Unroll* does much more when it encounters complex situations. Think of an outer junctions/symbolic links, which itself contains junctions/symbolic links, which are inner with respect to the first outer junction symbolic link:



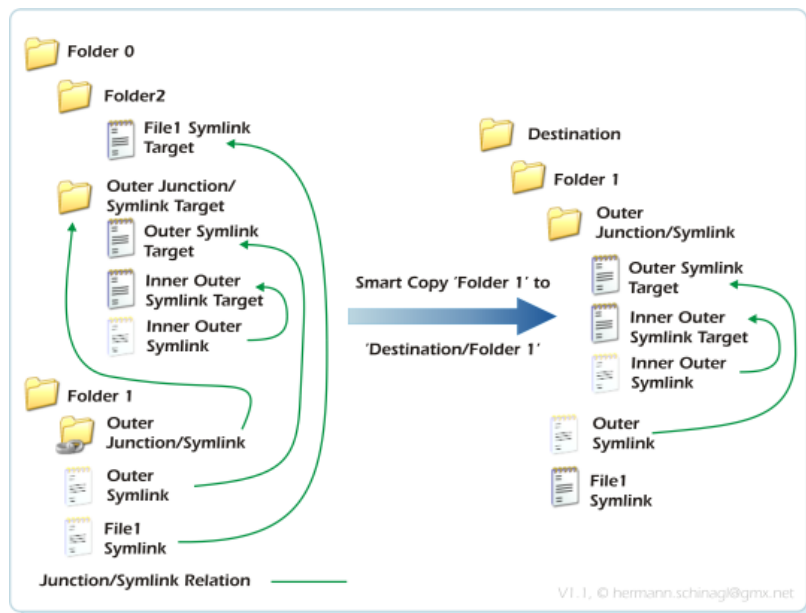
In the above example *Folder1* is copied to *Destination/Folder1*, and *Outer Junction/Symlink* and is unrolled as expected, but since *Junction/Symlink* is an inner junction with respect to *Outer Junction/Symlink Target*, the junction/symlink relation is restored in the destination.

This kind of **nesting** can be much more complex:



In the above example *Folder1* is copied to *Destination/Folder1*, and *Outer Junction/Symlink* and is unrolled as expected, but then it starts to get fascinating, because we have two levels of outer junctions/symlinks which all have respective inner junctions/symlinks, and which are restored properly. Once you digged yourself through the above picture, you got it. It is not simple I know, but it is necessary to properly unroll.

And complexity increases if **symbolic link files** are within unrolled outer junctions/symbolic links:

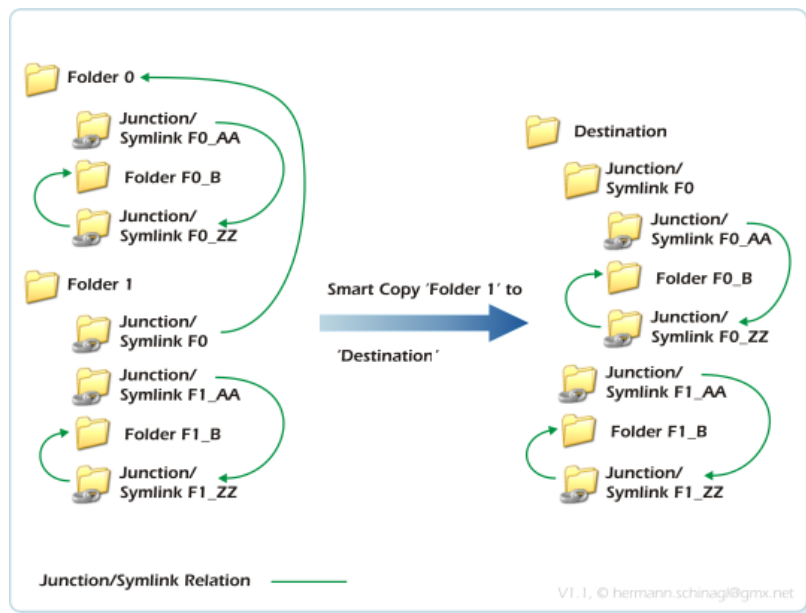


In the above example *Folder1* is copied to *Destination/Folder1*, and *Outer Junction/Symlink* and is unrolled as expected, but it contains *Inner Outer Symlink* which points to *Inner Outer Symlink Target* and this is an inner junction/symbolic link with respect to *Outer Junction/Symlink Target*

But worth mentioning is the Symbolic Link *Outer Symlink*, which would be a definitive outer symbolic link, but since its targets parent-directory *Outer Junction/Symlink Target* is unrolled, *Outer Symlink* becomes an inner symbolic link with respect to *Folder1*.

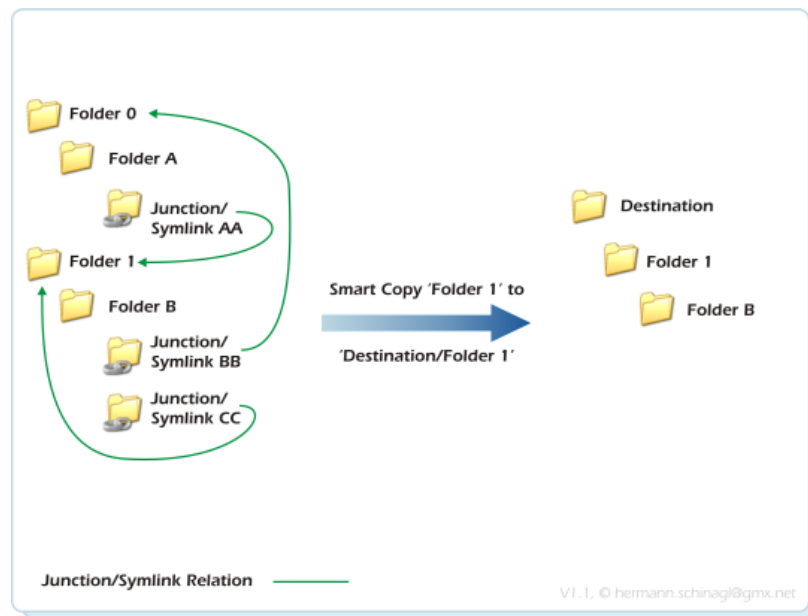
File1 Symlink is also an outer symbolic link, but its target parent-directory *Folder2* is not that lucky to get unrolled, so in the destination *File1 Symlink* is not a symbolic link any more, but a copy of the symbolic links' target.

Nested Reparse Points are also an interesting use case, which the algorithm has to tackle with:



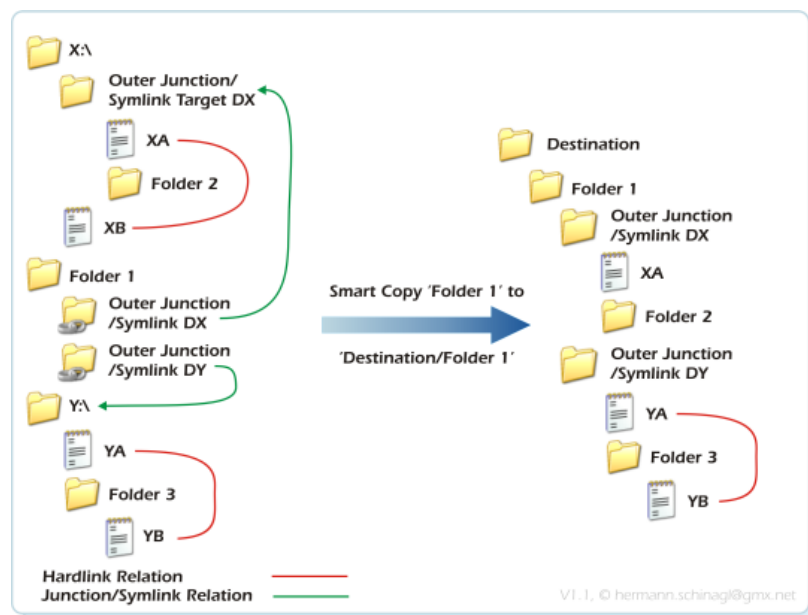
In the above example *Folder1* is copied to *Destination*, and *Junction/Symlink F0* and is unrolled as expected, but it contains inner nested reparse points. *Nested* means Reparse Points pointing to Reparse Points

The Unroll functionality also opens up the possibility to have **circular** Junction/Symbolic Link relations among a set of copied items:



In the above example *Folder1* is copied via the --unroll option to *Destination/Folder1*. Smart Copy/Smart Mirror and Delorean Copy operations can deal with the above shown circularities and break circularities by not following the affected Junction/Symbolic Link.

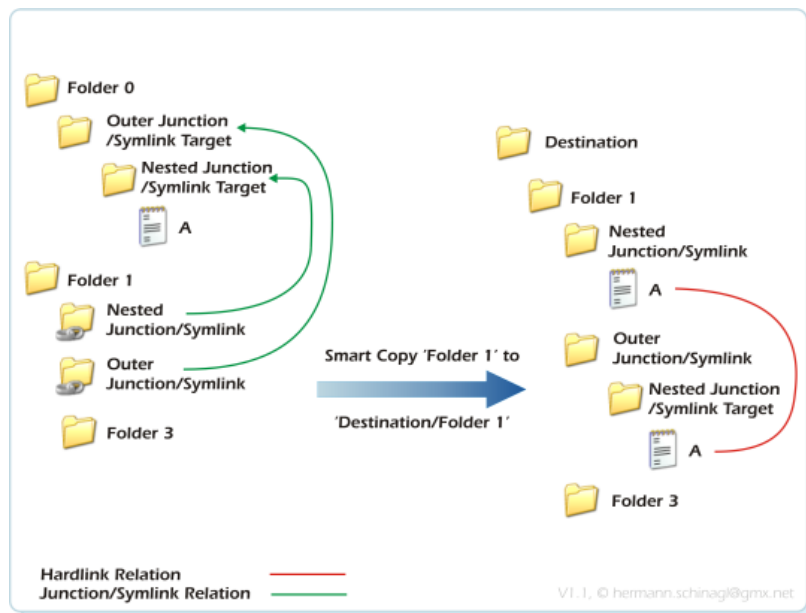
Junctions/Symbolic Links can also point to FAT drives or other NTFS drives requiring as a prerequisite unique Disk-IDs on all disks, which are chained together via Junctions/Symbolic links:



Hardlink siblings are found by matching the per NTFS volume unique file-id, but if more volumes are chained together it might happen that the same file-ids can be found on two different NTFS volumes. To address this all operations use the disk-id and the file-id to match hardlink siblings.

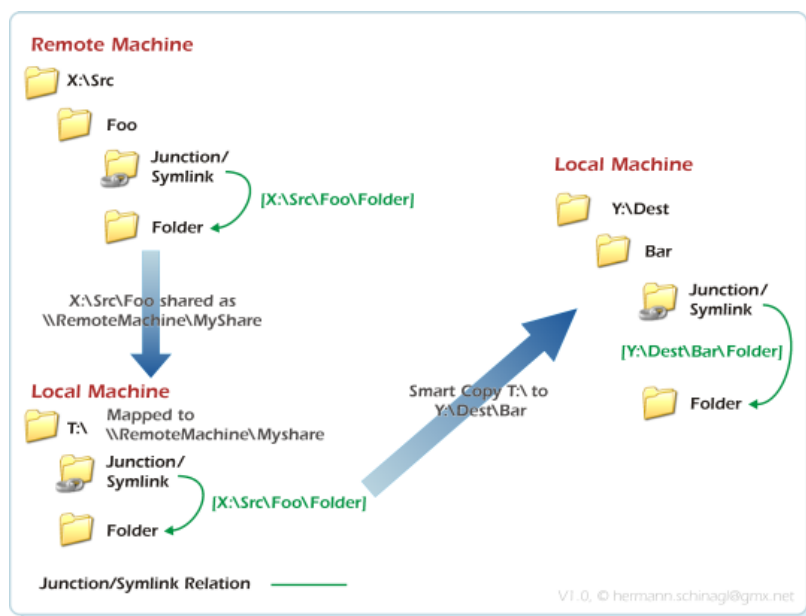
Furthermore it is not allowed to have the disk-id 0xffff-ffff, because the algorithms use this as internal indicator of a FAT drive.

The Unroll option also allows to point multiple junctions to the same target location, which causes the algorithms to traverse the same items many times:



At the first glance multiple traversal of items looks simple, but for files this means that multiple traversed files are the same in the destination and are hardlinked together. So don't be confused when you see hardlinks, which have never ever been there before.

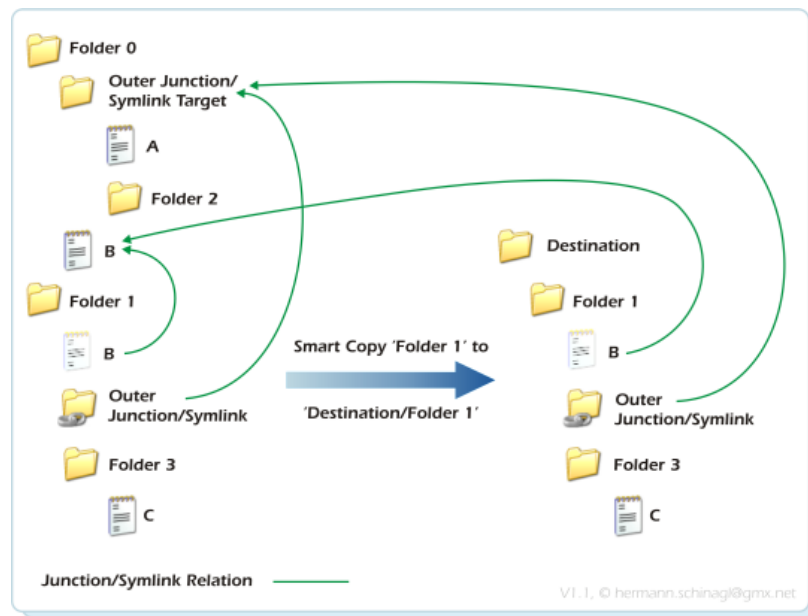
Copying Junction/Symbolic Links from mapped network drives also needs extra care, because junctions on the remote machine relate with path references only valid on the remote machine to each other:



The example in the above picture shows such a situation

- X:\Src\Foo\Junction/Symlink is a junction on a Remote Machine, which points to X:\Src\Foo\Folder. It is a valid Junction.
- X:\Src\Foo is shared via \\RemoteMachine\MyShare, which still is no problem.
- \\RemoteMachine\MyShare mapped to T:\ on a local machine is fine, but the Junction T:\Junction/Symlink, if asked for its target, will still return X:\Src\Foo\Folder even on the Local Machine. One could expect that the target is T:\Folder, but it is not for Junctions.
- When SmartCopy/Mirror/Delorean comes across such a situation, it translates the Junction correctly when it is copied onto the Local Machines drive Y:\, so that the final Junction target points correctly to Y:\Dest\Bar\Folder

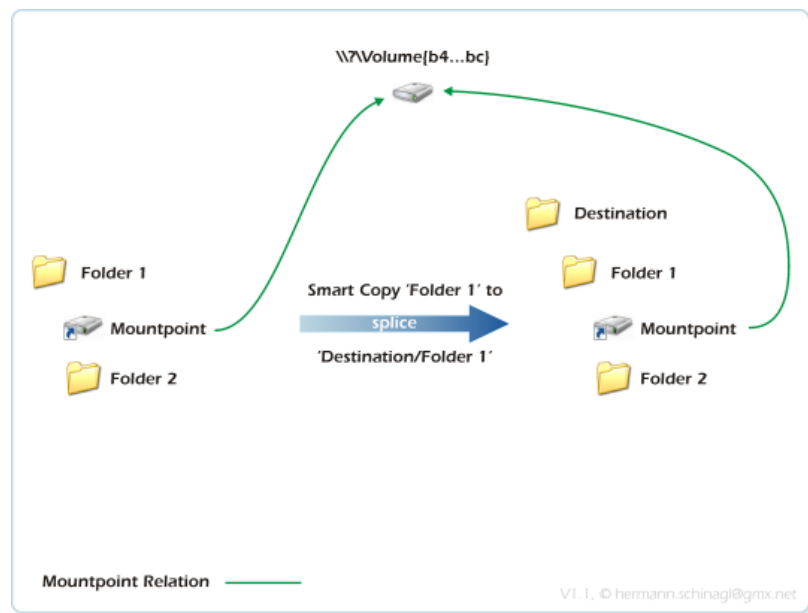
Splice *Splice* reconnects Outer Junctions/Symlink directories in the destination to their original targets.



In the above example *Folder1* is copied to *Destination/Folder1*, and *Outer Junction/Symlink* is available in the destination as junction, which points to the original location *Outer Junction/Symlink Target*.

The objective behind splicing Outer Junctions/Symlink Directories to its original location is to get a copy during smartcopy, but to reuse Outer Junctions/Symlink Directories source locations.

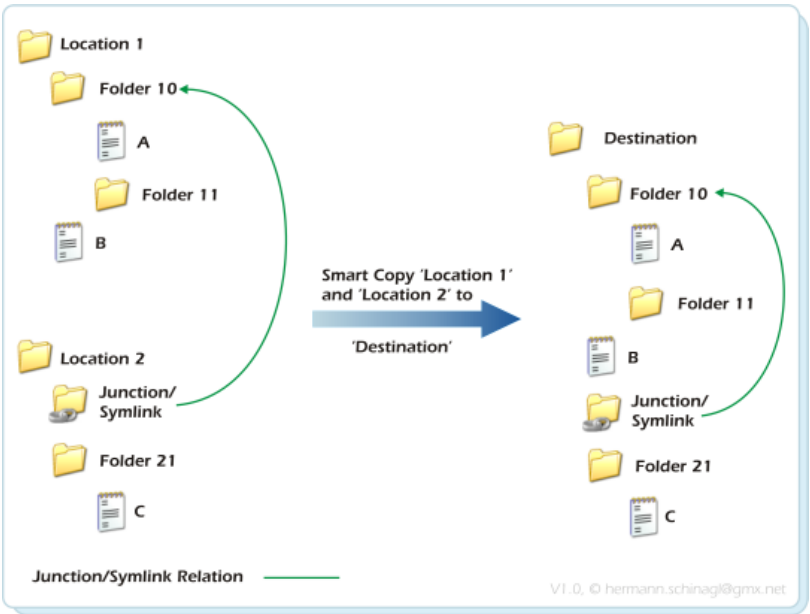
The *Splice* functionality is also useful, when mountpoints should be copied. Mountpoints are very similar to Junctions, but point to a path always starting with `\\?\VolumeGuid{}`.



Enabling Splice for Outer Junction/Symbolic Links

Splice can be selected via the [configuration tool](#).

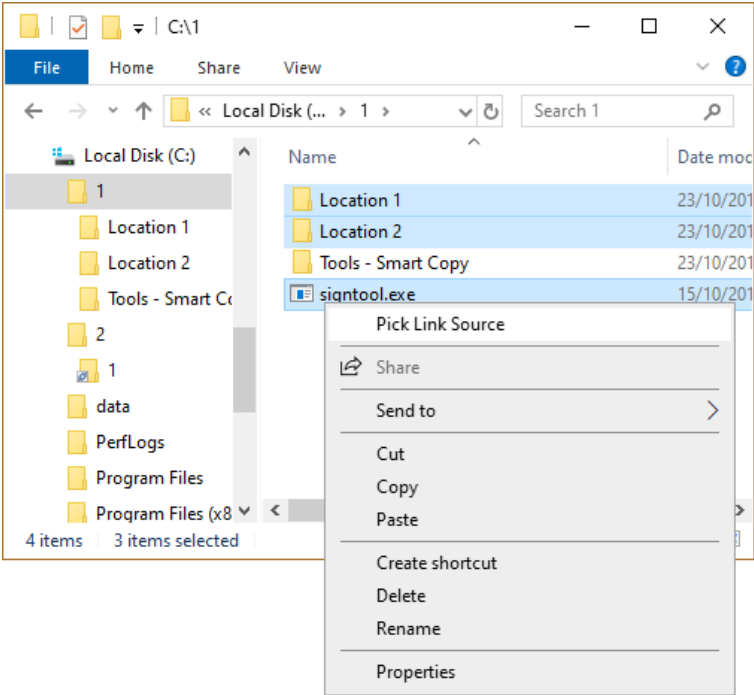
Multiple Source Multiple Source Locations can be specified for SmartCopy, Clone, and Delorean Copy. If there are junctions/symlinks between these source locations, they are handled as inner junctions/symbolic links, because all source locations are dealt like a common root.



In the above example *Location1* and *Location2* are copied to *Destination*. *Location2/Junction20* is treated as inner junctions to *Location1/Folder10* in the source, and that's why *Destination/Junction20* points to *Destination/Folder10* in the Destination.

The objective behind this is to treat all junctions/symlinks as inner junctions/symlins as long as they are in the set of source folders.

With LinkShell Extension this works as follows:

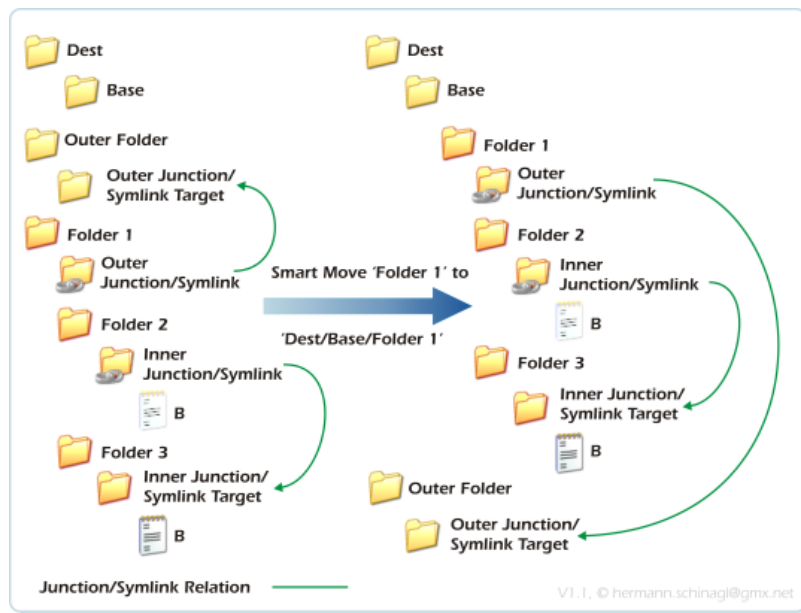


In the above example all content from *Location 1* and *Location 2* and pskill.exe are selected. Possible junctions/symbolic links in *Location 1* pointing to *Location 2* or vice versa are treated as inner junctions/symbolic links, because all selection is treated as a common root.

Smart Move

Smart Move enables folders with junctions and symbolic links beneath to be renamed. The junctions and symbolic links' targets are updated below that folder. Without Smart Move renaming of such folders would end in dead junctions and symbolic links.

With junctions or symbolic link directories it behaves as follows:



By closely looking at the above picture one can find three different types of folders/junctions:

Normal Folders

The folder 'Folder 3' is a normal folder. It gets moved with its content straight forward.

Inner Junctions/Symlinks

The folder 'Inner Junction/Symlink' is targeted at 'Inner Junction/Symlink Target'. In LSE terms this kind of folder is called *Inner Junction/Symlink*, because its target points to a folder, which is below the common anchor 'Folder 1'.

Inner Junctions/Symlink are updated properly via Smart Move in the destination location.

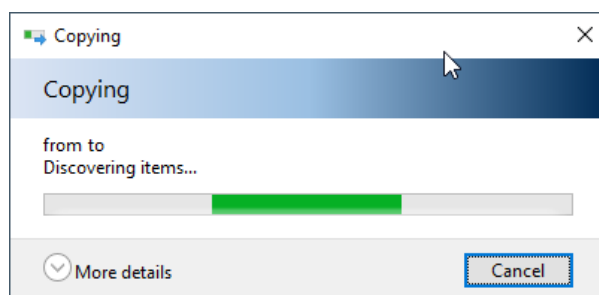
Outer Junctions/Symlinks

The folder 'Outer Junction/Symlink' is targeted at the folder 'Outer Junction/Symlink Target'. In LSE terms this kind of folder is called *Outer Junction/Symlink*, because its target points to a folder, which is in parallel and thus outside the anchor 'Folder 1'.

Outer Junctions/Symlinks are not touched by Smart Move and thus stay connected to their respective target. Please note that this is different to Smart Copy, which has [3 different ways](#) to deal with Outer Junctions/Symbolic Links.

The Smart Move functionality is integrated into Explorer seamlessly, so that you don't have to do anything special. Simply drag a folder in explorer to its destination location, or e.g. press F2 in Explorer to rename a directory and LSE will intercept this operation under the hood, takes care of junctions or symbolic links, and will update them.

Intercepting move and rename operation means, that LSE takes over control before rename/move, and recursively searches the selected folder for junctions or symbolic links. But searching large amounts of files and folders takes time, so LSE will show a progress bar when searching takes longer than 250msec.



If symbolic links have to be updated LSE calls its [UAC](#) helper LSEUacHelper.exe.

If backup mode is enabled the UAC helper LSEUacHelper.exe is called anyway, because LSE needs to enumerate files in locations, where it might have no permissions.

Smart Move creates [relative](#) symbolic links during the Smart Move operation.

If a LocalizedResourceName is given via desktop.ini and the very folder is readonly, SmartMove will not work. This seems to be a bug in Shell Explorer ever since. ICopyHook::CopyCallback() is not called at all in this situation.

Enabling/Disabling Smart Move

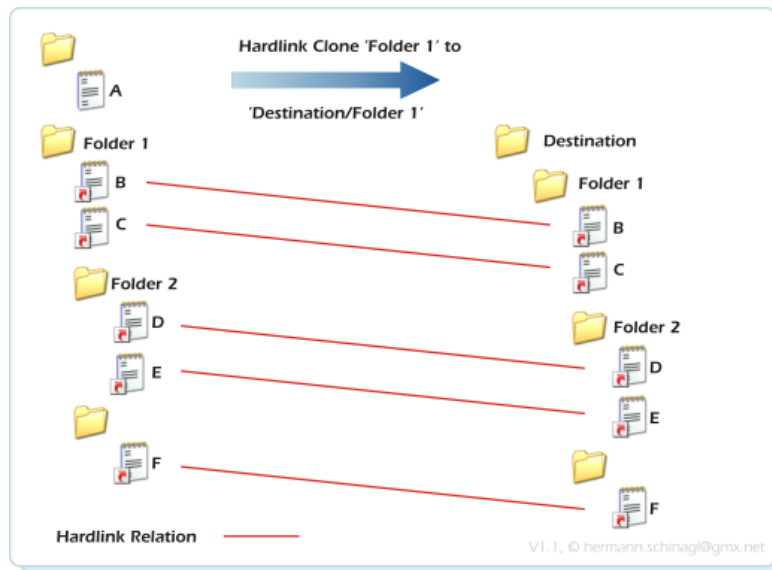
Smart Move can be switched on/off via the [configuration tool](#)

Command Line

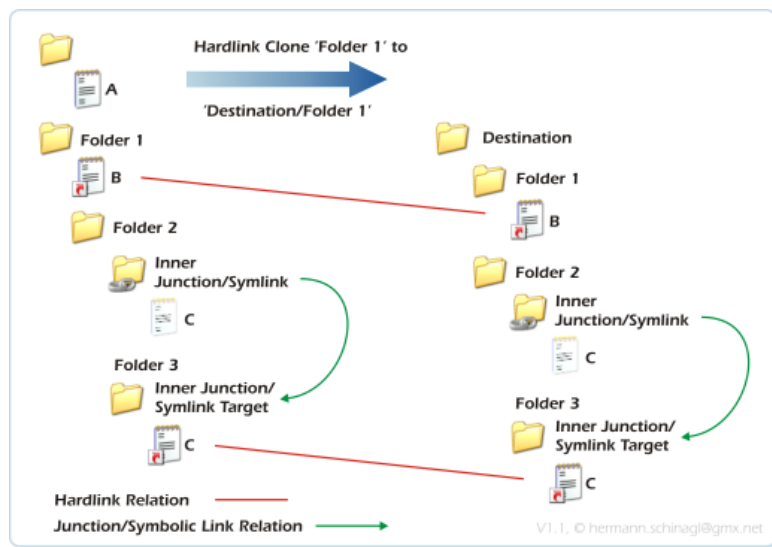
The Smart Move functionality is also available via command line from [ln.exe](#) via the --move command line switch.

Clone

Clones are copies of a folder tree from a source location recreated at the destination location, however the files within the new folder tree are Hardlinks or Symbolic Links to the respective files in the source folder tree.

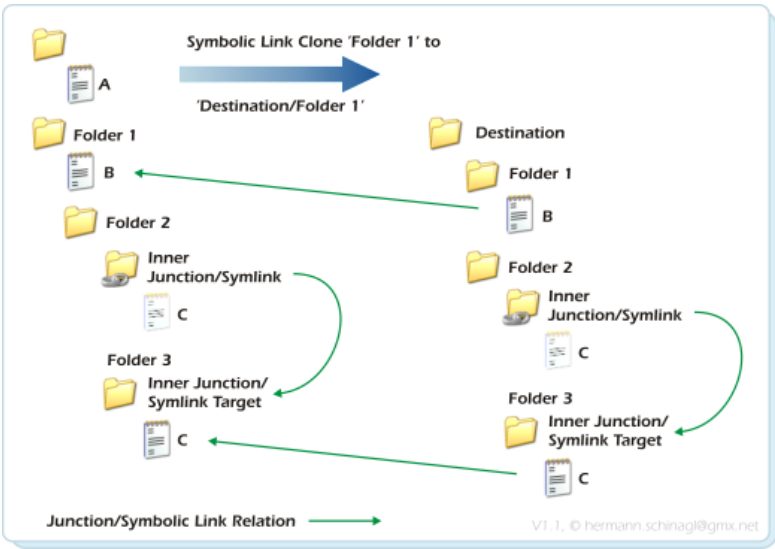


A folder tree might also contain Junctions or Symbolic Links. The clone process recreates [inner junction/symbolic](#) links at the destination location very similar as Smart Copy does.

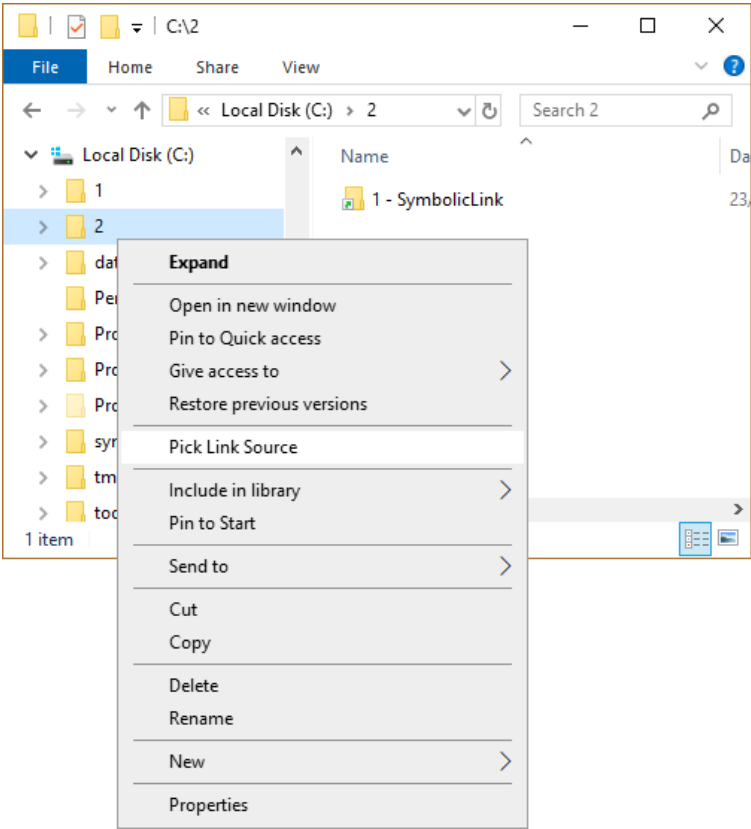


Outer Junctions/Symbolic links are recreated with respect to the specified [Outer Junction/Symbolic Link handling](#). e.g.

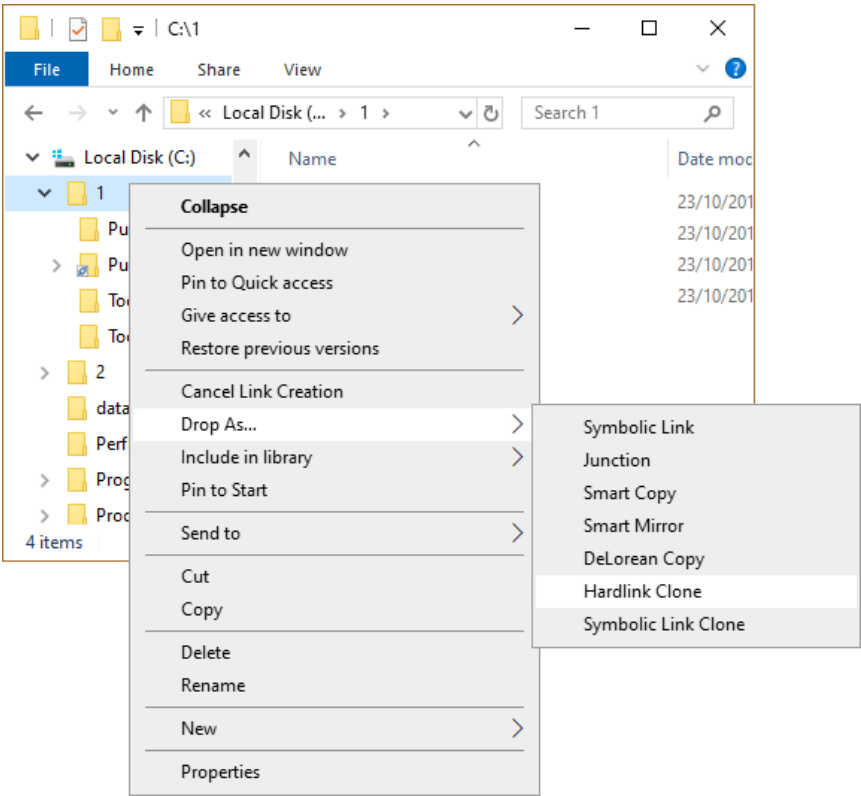
With Windows 7/8/10 this cloning process is also available with Symbolic Links instead of Hardlinks.



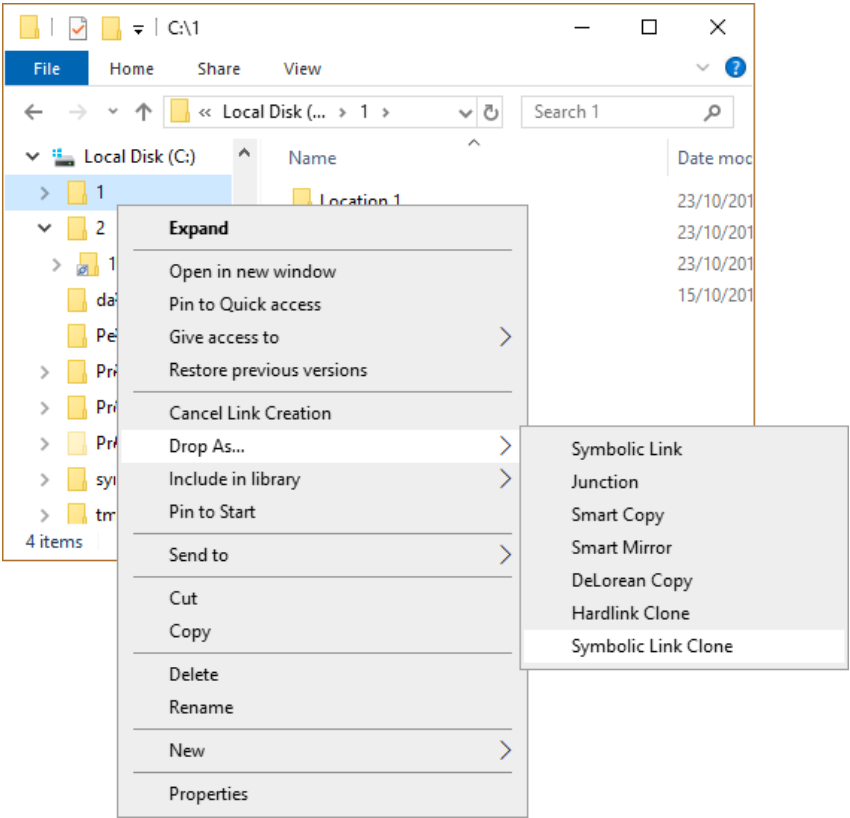
Clones are created in the same way as e.g Junctions. Select a folder, click the Action button, choose **Pick Link Source** from the action menu...



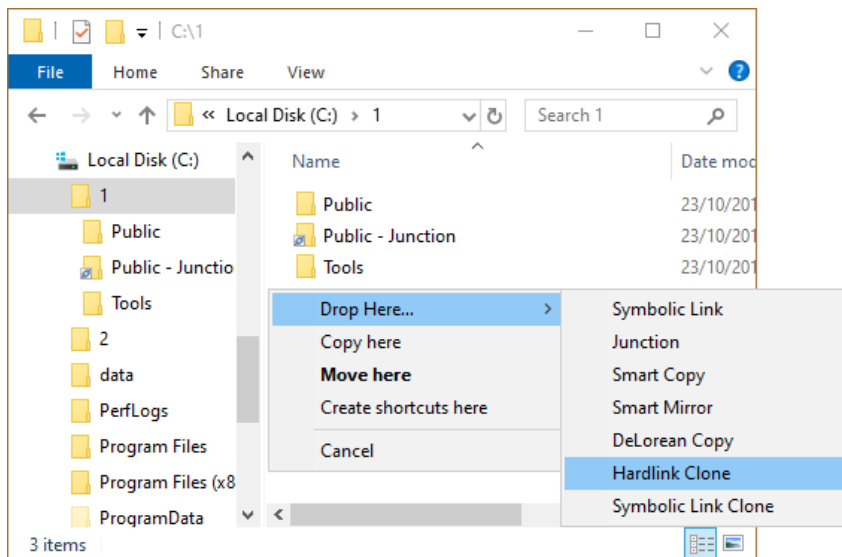
...navigate to the destination folder, press the action button, open the **Drop As ...** submenu and select **HardLink Clone**:



... choose Symbolic Link Clone to create clones of existing tree structures.



HardLink and Symbolic Link Clones can also be created via Drag and Drop. Select a folder and drag with the action button depressed to a destination folder. When the action button is released open the **Drop Here...** submenu and select **HardLink Clone** or with Windows 7/8/10 **Symbolic Link Clone**:



HardLink or Symbolic Link Clones are useful if you need to replicate a folder tree at a different location. The disk space required is minimal because the new structure consists entirely of NTFS directory entries with no real amount of actual data storage.

If both files and folders are selected as Source Links and dropped as a **HardLink Clone** then the selected files are dropped as Hardlinks alongside the HardLink Clones.

Because Clones use Hardlinks or Symbolic Links, they are only available within an NTFS volume. Hardlink Cloning can not replicate the folder structure from one disk volume to a different volume, because Hardlinks are limited to operation on a single volume. Symbolic Link Clones can be used to create volume spanning Clones.

When creating Clones, LSE forks its helper LSEUacHelper.exe to forwards this operation to it, if the folder tree contains [symbolic links](#), because the creation of symbolic links needs elevation, and thus brings up the [famous UAC](#) dialog. LSE **only** issues its helper LSEUacHelper.exe if a symlink is among/below the selected folders, so it saves you from one UAC prompt if you don't have symlinks among your selection.

Limitations

Hardlink creation needs write access to the source hardlink sibling. This means Hardlink Clone will fail, and just generates an empty directory structure, if write access for the source files is not available. This is caused in the way Windows implements hardlinks, because the security information for a file and thus for all hardlink siblings is shared among all hardlink siblings.

Creating hardlink clones in such cases can be enabled, by granting write access for the source files to the user who will create a hardlink clone, or in a single user use-case to everyone.

Command Line

The Clone functionality via Hardlinks or Symbolic Links is also available via command line from [ln.exe](#) via the `--recursive` command line switch.

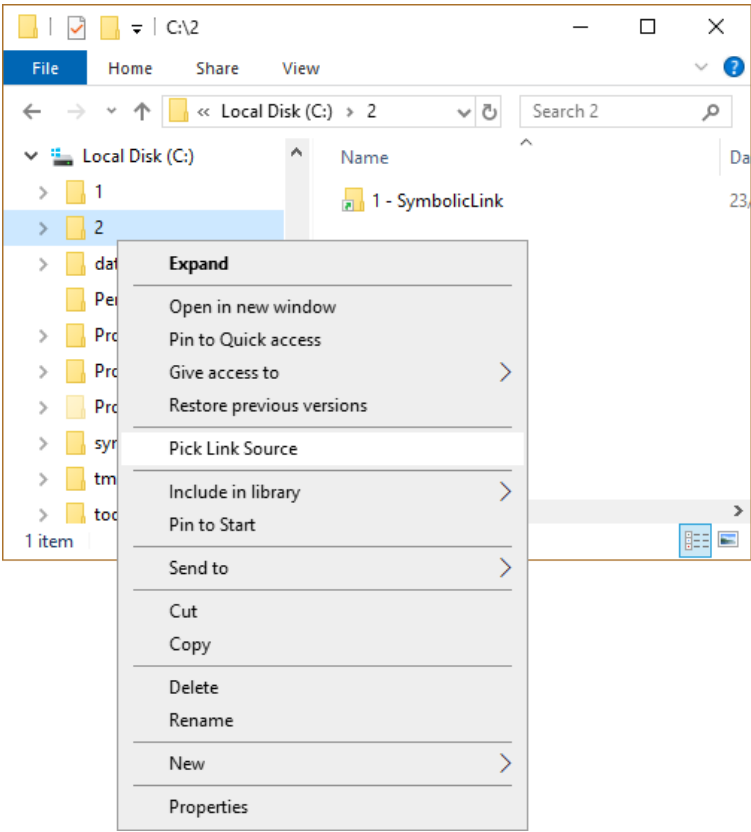
Smart Mirror

Smart Mirror is very similar to [Smart Copy](#) and not only copies but *synchronises* the folder *Source* to *Destination*:

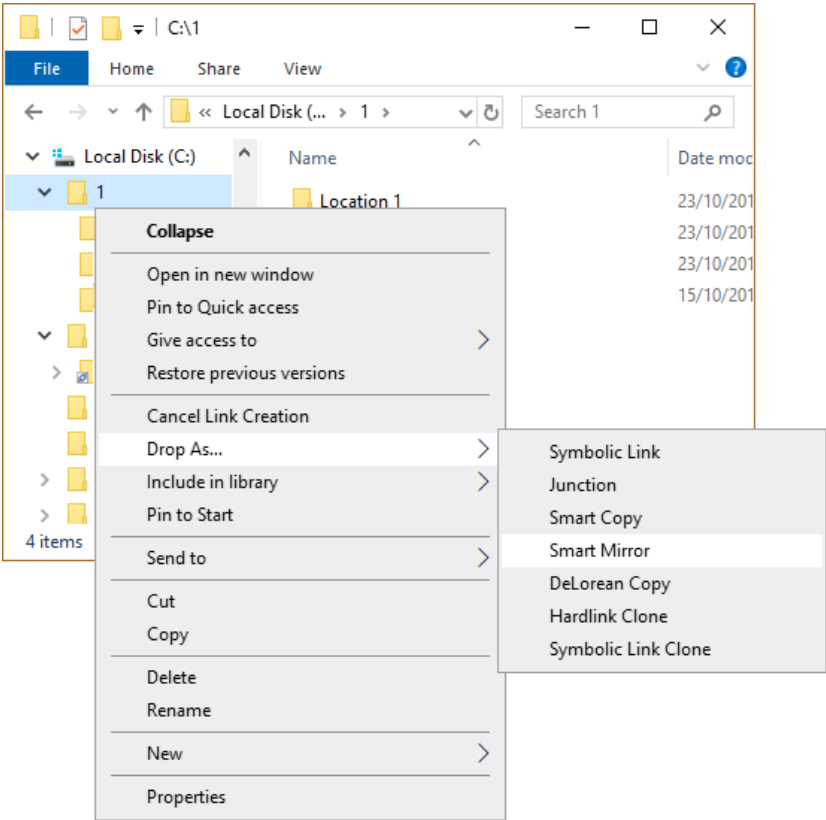
- Smart Mirror copies if the timestamp of items in the *Destination* is different from the *Source*.
- Delete files not anymore available in *Source* from *Destination*.

Smart Mirror takes care of Hardlink Relations, Restores Inner Junctions or Inner Symbolic links or when issued [unrolls or splices](#) Outer Junctions or Outer Symbolic Links.

Smart Mirror is created in the same way as e.g Junctions. Select a folder, click the Action button, choose **Pick Link Source** from the action menu...

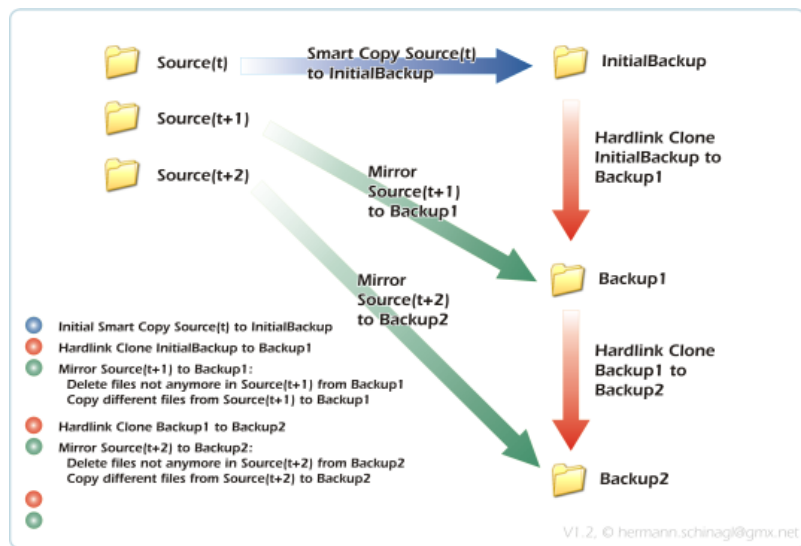


...navigate to the destination folder, press the action button, open the **Drop As ...** submenu and select **Smart Mirror**:



Smart Mirror is a little bit different with respect to [Auto Rename](#), because it expects a already existing folder in the destination location, which has the same name as the source folder, so that it can do the mirror.

DeLorean Copy DeLorean Copy is a way of creating incremental backups by using a combination of hardlink clone and Smart Copy.
The following picture gives an overview what DeLorean Copy is about



In general a DeLorean Copy has 3 principals: Source(t), InitialBackup and Backup(n).

Phase 1: Initial SmartCopy

The folder *Source(t)* is initially copied to *InitialBackup*. This is shown by the blue arrow.

Changes happen

During this phase the files under source change, and *Source(t)* becomes *Source(t+1)*.

Phase 2: Hardlink Clone

The folder *InitialBackup* is Hardlink Cloned to *Backup1*, which ties *InitialBackup* and *Backup1*. This is shown by the red arrow.

Mirrors the folder *Source* to *Backup1*. This is shown by the green arrow:

Phase 3: Mirror

- Keeps unchanged files as hardlinks to InitialBackup.
- Deletes files not anymore in *Source(t+1)* from Backup1.
- Copies different files from *Source(t+1)* to Backup1.

With completion of this first round *Backup1* contains the first lean and mean copy of *Source* only consisting of either hardlinks to *InitialBackup*, or of copied files if there was the need to copy them over from *Source(t+1)* because they were newer under *Source(t+1)*.

The point is that all files in *Backup1* are transparently accessible, but really little space is used, because not all files in the *Source(t+1)* changed, so that there was only the need to effectively copy over a few files from *Source(t+1)* to *Backup1*. This can be repeated on and on. The second round would be using *Source*, *Backup1* and *Backup2* for DeLorean Copy:

Changes happen

During this phase the files under source change, and *Source(t+1)* becomes *Source(t+2)*.

Phase 2: Hardlink Clone

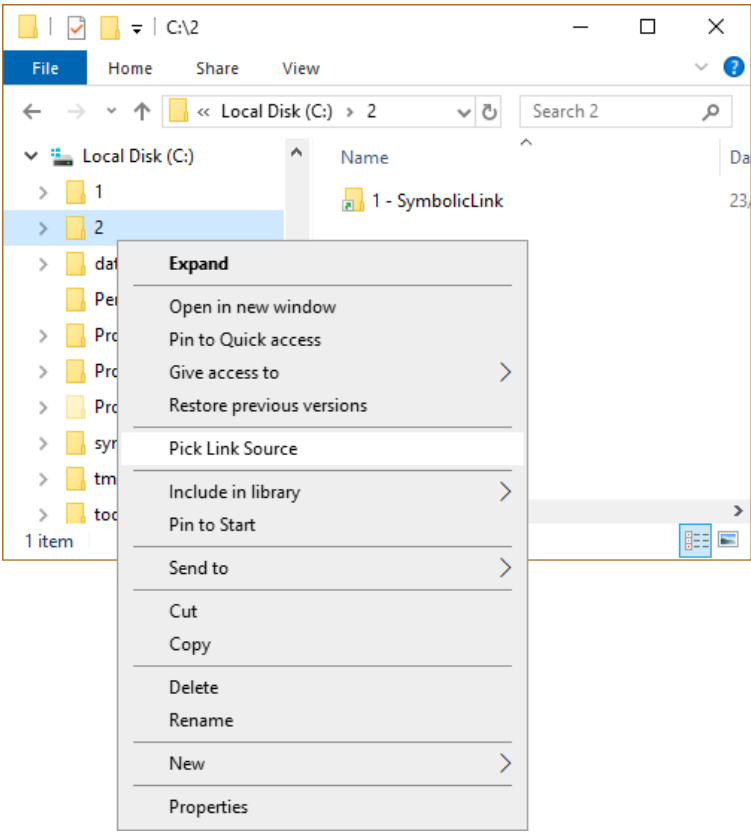
The folder *Backup1* is Hardlink Cloned to *Backup2*, which ties *Backup1* and *Backup2*. This is shown by the red arrow.

Mirrors the folder *Source(t+2)* to *Backup2*. This is shown by the green arrow:

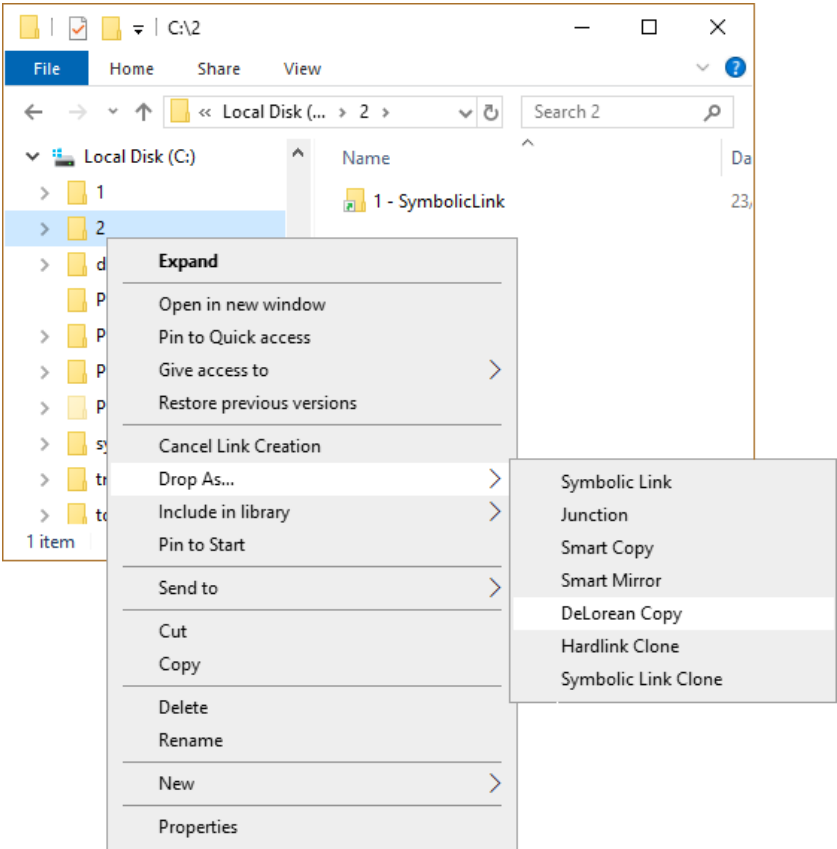
Phase 3: Mirror

- Keeps unchanged files as hardlinks to Backup1.
- Deletes files not anymore in *Source(t+2)* from Backup2.
- Copie newer files from *Source(t+2)* to Backup2.

DeLorean Copies are created in the same way as e.g Junctions. Select a folder, click the Action button, choose **Pick Link Source** from the action menu...



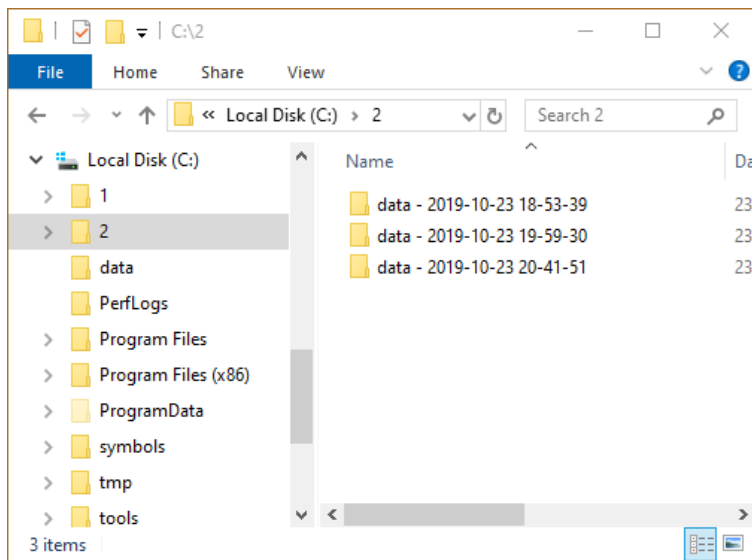
...navigate to the destination folder, press the action button, open the **Drop As...** submenu and select **DeLorean Copy**:



If a DeLorean Copy was dropped the first time onto a directory, the operations, which are described under phase 1 in the above descriptions, namely a Smart Copy, is performed. Link Shell Extension automatically generated the folder name for the destination by appending a timestamp.

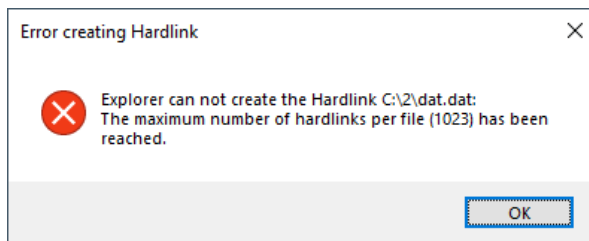
Any successive drop of a directory onto the a destination directory initates Phase 2 and Phase 3 from the above description, namley it does the Hardlink Clone from the former backup onto the current backup and furthermore does mirror the source onto the current backup.

A directory holding many copies may look like this.



Limitations

It is little known, but NTFS has a limit to create a maximum of 1023 hardlinks to one file. For DeLorean Copy this means that it will display an error message if this limit is exceeded, because exceeding this limit means loss of data among the most recent backup sets:



The reason for exceeding this limit could either be, that there have been more than 1023 backup sets but no hardlinks within the *Source*, or there are hardlinks within the *Source* and less than 1023 backup sets.

The DeLorean Copy submenu will **not** appear if **more than one** folders are selected as source.

DeLorean Copy is long path safe which means it can handle more than 256 characters in pathes. This is important, because placing a copy with quite long path, but still below 256 characters path length, to destination locations might result in combined path length greater than 256 character. In such a situation no data loss must happen, which DeLorean copy guaranties with beeing 'long path safe'. Please make sure that Explorer can not show you the result of such a copy, but the files are there. Alternative explorers like [SpeedCommander](#) can handle this.

Command Line

DeLorean Copy functionality is also available via [ln.exe](#)

Backup Mode

Backup mode enables LSE to also copy ACLs and Encrypted Files from all directories, even the ones without access for the current users.

- LSE always forks its helper process LSEUacHelper.exe, and thus raises the [UAC prompt](#) for elevation and password verification.

It applies to SmartCopy, SmartMirror, SmartClone, DeLorean Copy, SmartMove and Replacement Junction/Symbolic Link/Mountpoint.

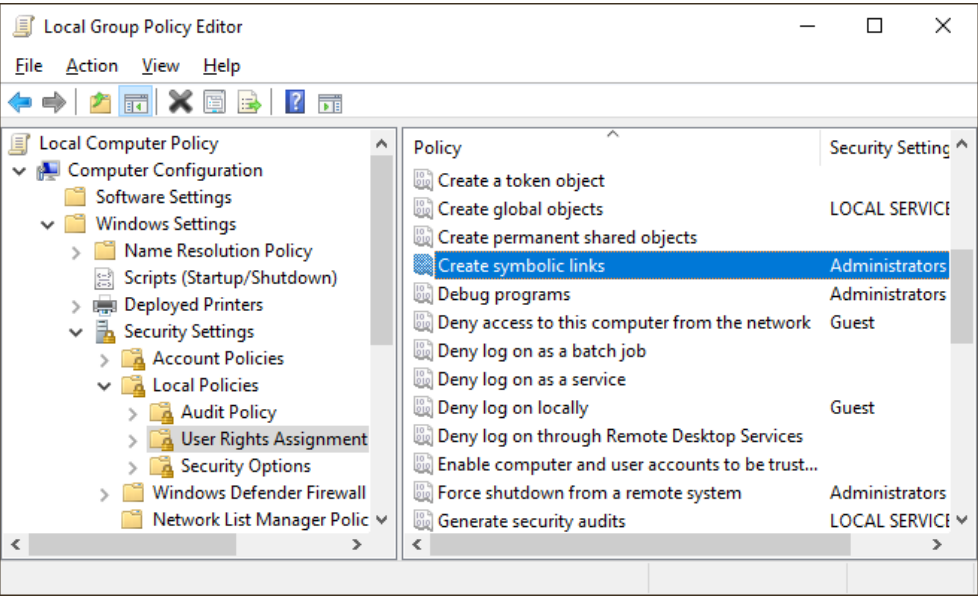
LSE.exe thus copies

- Alternative Streams on files and folders
- EA Records on files and folders (rarely used)
- Reparse Info
- File Attributes
- Timestamps: Creation Time, Last Access Time, Last Write Time
- Sparse Files and Alternative Sparse Streams
- Encrypted files
- ACLs

Backup Mode is disabled by default, and can be turned on via the [LSE configuration](#) tool.

To perform backup operations a user must hold the SE_BACKUP_NAME and SE_RESTORE_NAME privileges. An out of the box Windows configuration assigns the Backup-Operator and Administrator group these privileges, but additionally the above privileges can be assigned individually to certain users or groups.

Assigning privileges can be accomplished
By **gpedit.msc**



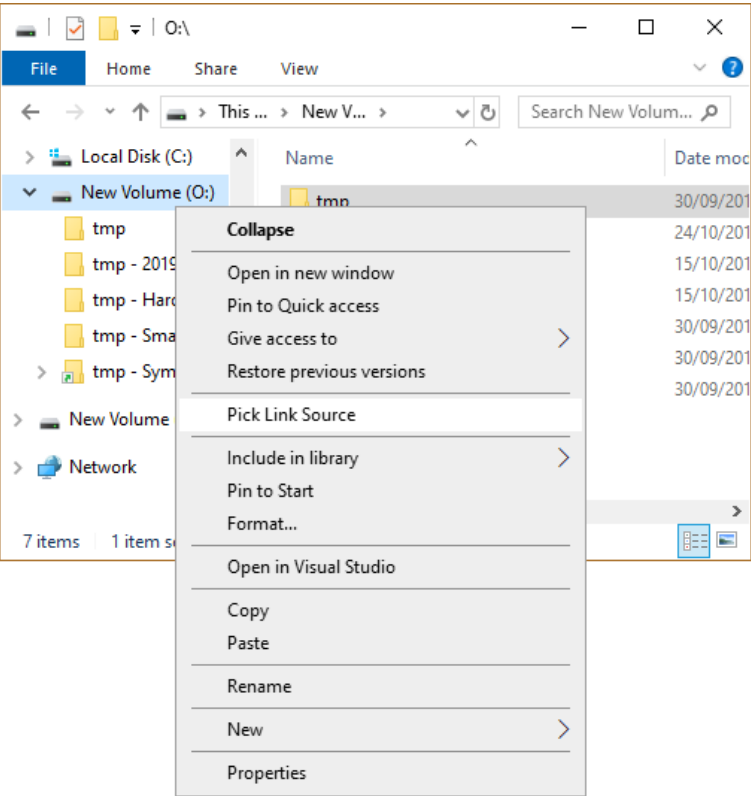
and navigating to
"Computer Configuration" ->"Windows settings" -> "Security Settings" -> "Local Policies" -> "User Rights Assignments" -
> "Backup files and directories"
"Computer Configuration" ->"Windows settings" -> "Security Settings" -> "Local Policies" -> "User Rights Assignments" -
> "Restore files and directories"

Without gpedit.msc

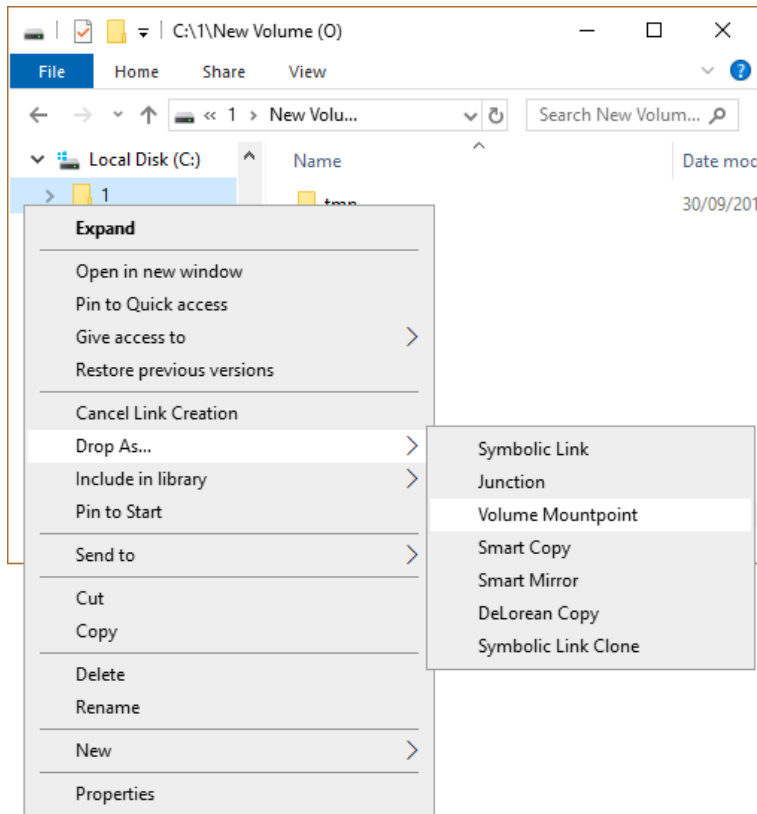
Download [PolsEdit](#) and add users or groups, who should be able to run backups, to the SE_BACKUP_NAME and SE_RESTORE_NAME privileges.

Volume Mount Point Support

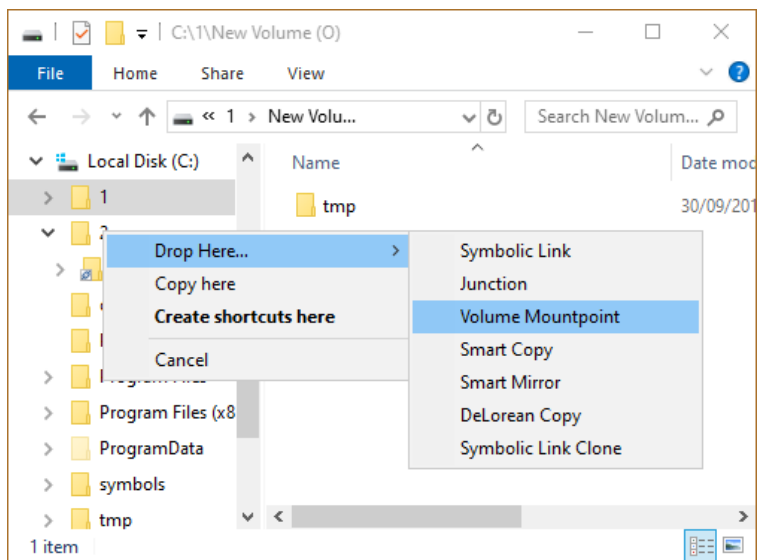
Volume Mountpoints provides functionality to map complete local volumes onto arbitrary disk locations.



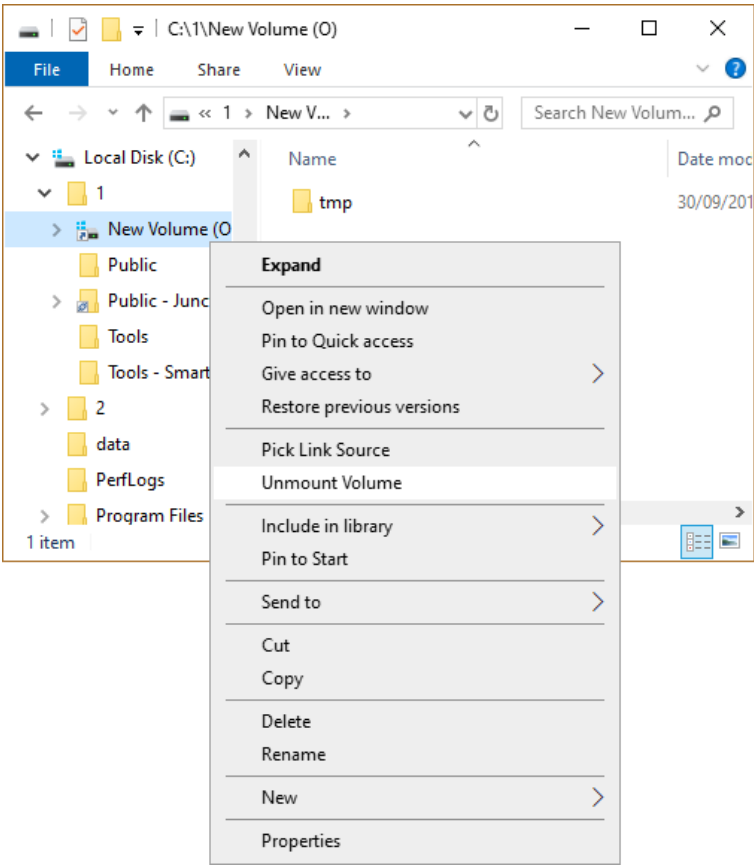
Volume Mountpoints are created in the same way as Hardlinks, except that the Source Link is a volume rather than a file. Select a local volume, click the right mouse button, choose **Pick Link Source** from the action menu, navigate to the destination folder, click the action button, open the submenu **Drop As ...** and select **Volume Mountpoint**:



Volume Mountpoints can also be created via Drag and Drop when the selected local volume is dragged with the action button pressed to a destination folder; when the right mouse button is released, select the **Drop Here ...** submenu and then **Volume Mountpoint**.



Mount Points can be deleted by using the Unmount Volume command from Explorer as usual.



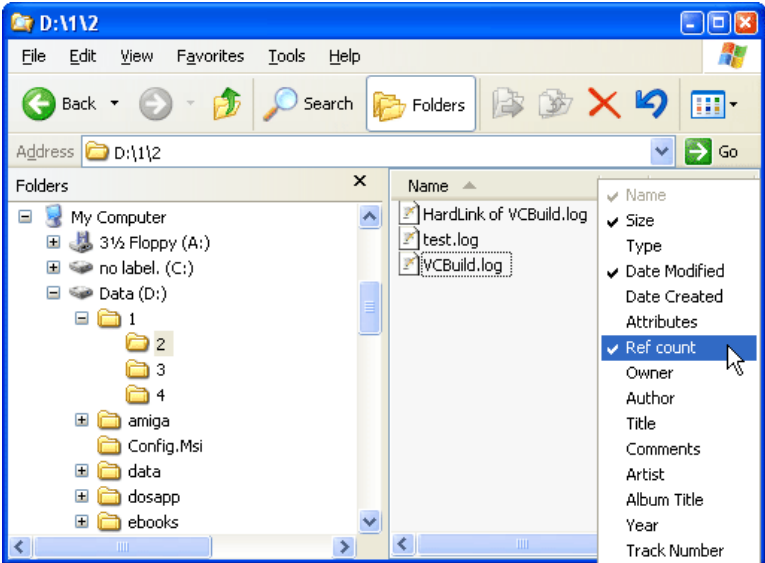
To show the origin of a Volume Mountpoint, the [reference column](#) of a Volume MountPoint shows the volume which is mounted onto the selected path.

Make sure that only local volumes can be mounted but not mapped network drives.

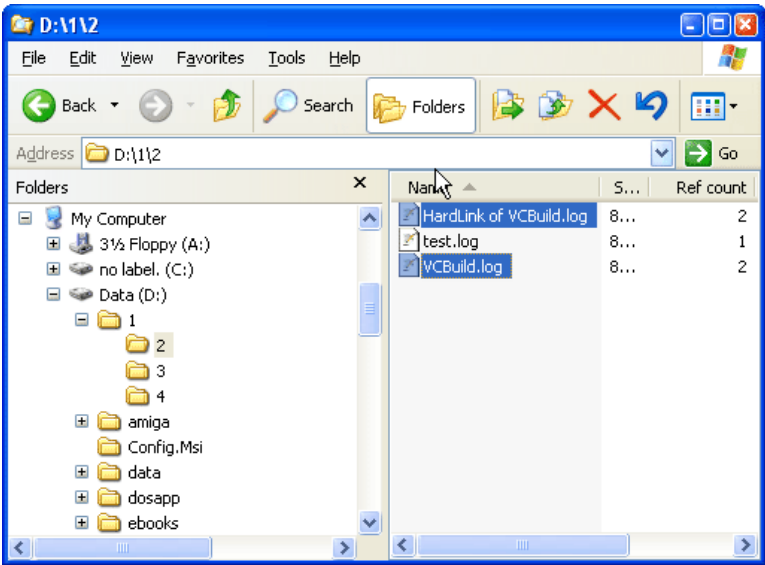
The creation and deletion of Volume Mountpoints is bound to successful elevation, which means that the [famous UAC](#) dialog must be acknowledged.

Reference Count As described in the [backgrounders section](#) NTFS maintains a reference count for each data stream object how many NTFS directory entries refer to that objects.

To show the reference counts, a column can be enabled in Explorers right pane by action clicking the Titles row of the details view.



After enabling the reference column the reference count is shown for each file.

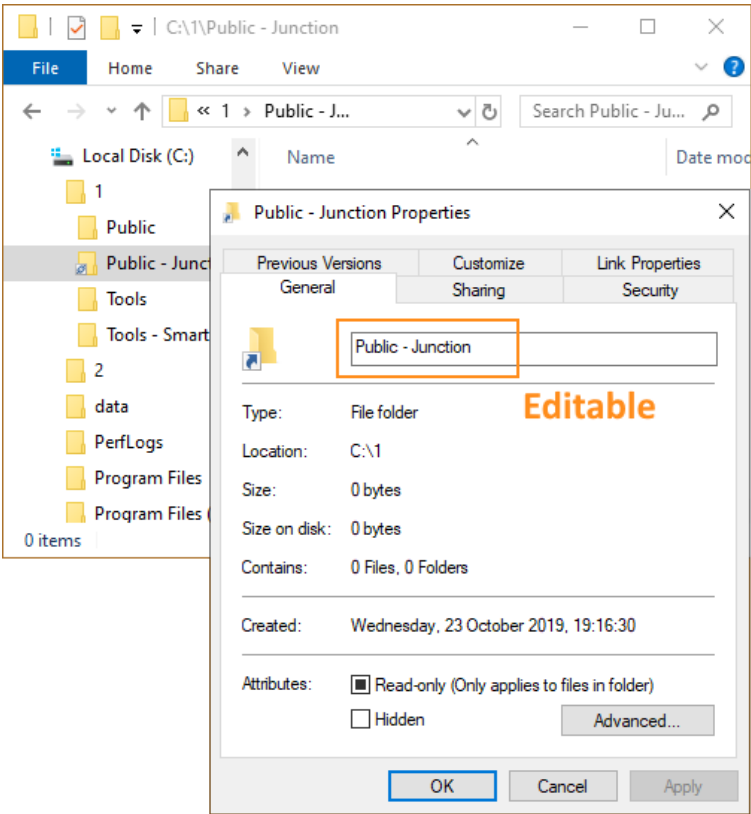


Windows7/8/10: The column, which shows the reference count and the origin of the junction is not available, because the way Windows7/8/10 handles user defined columns has been completely revamped by Microsoft and all applications working with so called ColumnHandlers will not work.

Link Shell Extension also supports so called Explorer Property Sheets, which means that if a file or directory property in explorer is opened, Link Shell Extension adds its own tab to show the properties of a hardlink, junction, volume mountpoint or symbolic link.

This additional tab only shows up in the file or directory properties, if the file or directory is a hardlink, junction, volume mountpoint or symbolic link, otherwise this tab is not available.

Link Properties



Explore

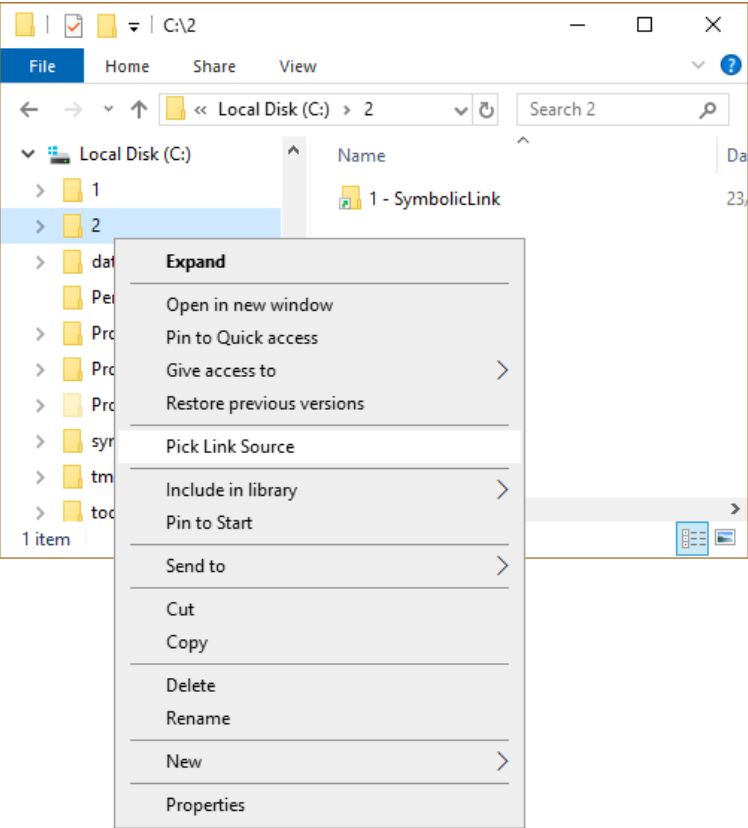
For junctions, volume mountpoints or symbolic links this dialog also shows a 'Explore Target' button, which opens an explorer in the specified directory.

Edit

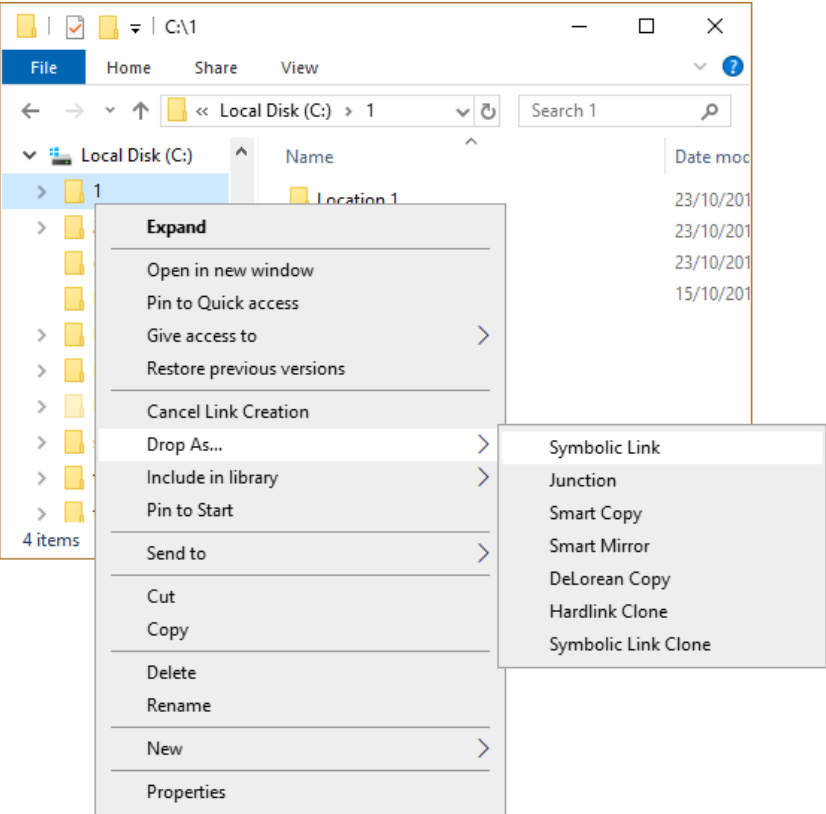
For Junctions, Symbolic Links or Mountpoints the target field can be edited, and after either pressing the Apply button or leaving the Link Property dialog with ok, the changes are applied to the Junction, Symbolic Link or Mountpoint.

If the [Backup Mode](#) is enabled, the ACL of the edited Junction, Symbolic Link or Mountpoint is preserved.

Creating a **Symbolic Link** is essentially the same as the other Link creation processes. Action click on the selected file(s) and select Pick Link Source(s) from the action menu.



When the destination folder is action clicked the menu contains a **Drop As ...** submenu, to create a Symbolic Link select SymbolicLink from the submenu. Unlike Hardlinks Symbolic Links can span storage volumes.



If both files and folders are selected as the Source Links and dropped as a **Symbolic Link Clone** then the selected files are dropped as Symbolic Links alongside newly created *Symbolic Link Clone* folders.

Symbolic Links can also be created between directories.

Relative versus absolute symbolic link target pathnames

The target of a symbolic link can either be

- a fully qualified path starting at the root of a drive, e.g `e:\data\cpp\myfile.txt`
- or can be specified relativeley, e.g `..\..\data\cpp\myfile.txt`

LSE by default tries to create **relative** target path names for symbolic links as long as this is possible, e.g the file and its target are on the same logical drive. Having relative symbolic link targets is much smarter especially when the target of links is in the same directory. If a symbolic link and its target are on different drives, LSE uses absolute pathnames.

The [configuration](#) tool can switch Link Shell Extension in either relative or absolute mode.

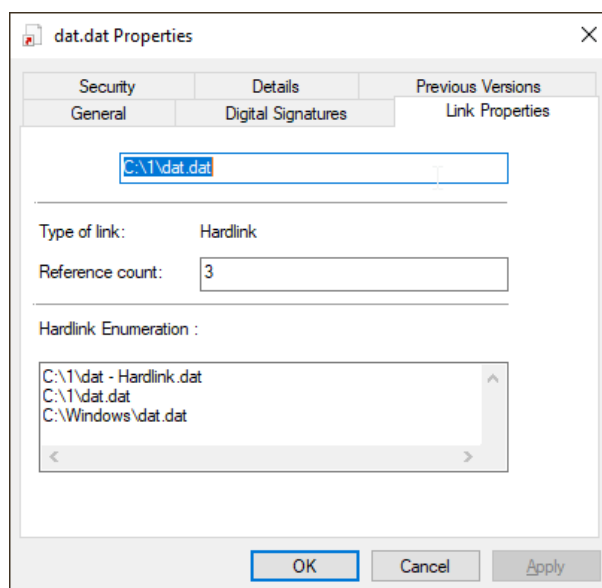
Overlay Icons for Symbolic Links

To help distinguish Symbolic Links from normal files/directories, an **overlay icon** is implemented on symbolic links that shows a light green arrow icon under the folder.



Overlay icons for Symbolic Links can also be [customized](#).

Simply select a hardlinked file and select [Properties](#) from the action menu:

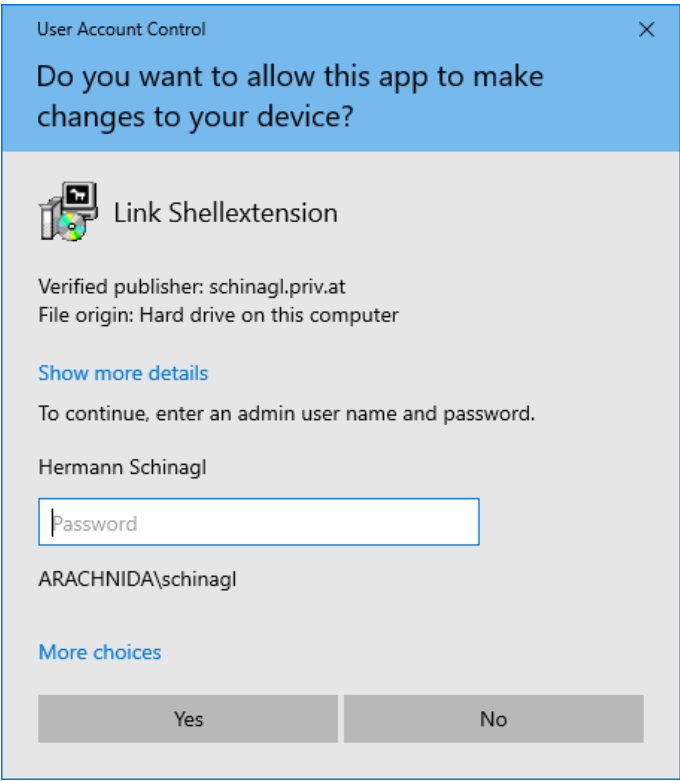


Enumeration of Hardlinks

The Hardlink Enumeration functionality is also available via command line from [ln.exe](#) via the `--enum` or the `--list` command line switch.

UAC

Due to UAC some API calls need elevation to administrative level, and this elevation must be acknowledged via the below shown dialog box. So if you see the below box, and the program asking for elevation is LSEUacHelper.exe, it is Link Shell Extensions contribution to UAC and you must acknowledge it to get Symbolic Links created.

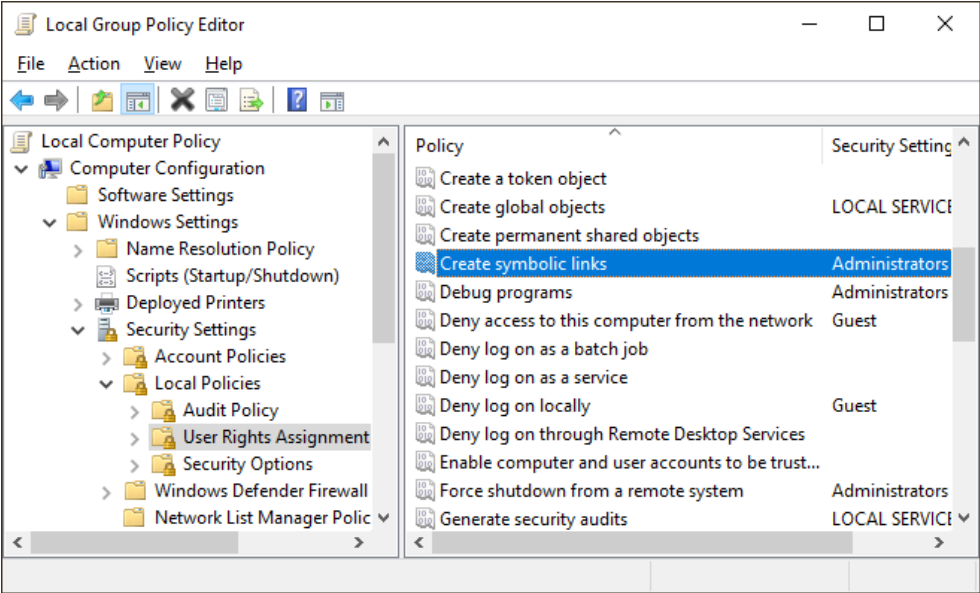


One way to come around the UAC prompt for the creation of Symbolic Links is to globally allow users to create Symbolic Links by changing the policy.

With gpedit.msc

Start gpedit.msc from the command line and grant/revoke specific users the permissions: Under "Computer Configuration" -> "Windows settings" -> "Security Settings" -> "Local Policies" -> "User Rights Assignments" -> "Create Symbolic Links".

Change Symbolic Link Privilege



Without gpedit.msc

Download [PolsEdit](#) and add users to SE_CREATE_SYMBOLIC_LINK_NAME. Please make sure you log off and log on after changing policies, so that the changes become active.

Linkshell Extension can deal with the above granting of privileges, and if the Symblic Link Privilege is available avoids the UAC prompt.

Developer mode in Windows10

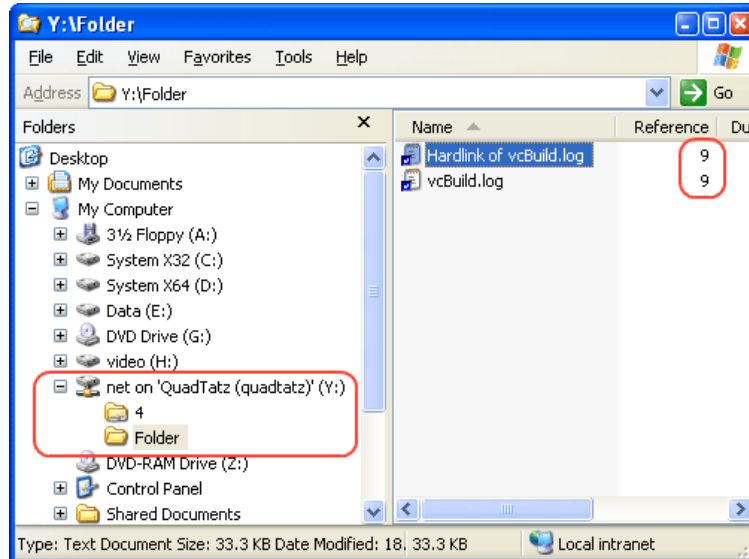
Another way to come around the UAC prompt would be to [enable the developer mode](#), which is available since Windows10/14972.

It is little known, but the SMB networking protocol supports operations to create remote Hardlinks, Junctions and Symbolic Links within SMB mapped network NTFS drives.

This feature is used by Link Shell Extension, to enable the creation of so called Remote Hardlinks, Remote Junctions, or Remote Symbolic Links. e.g.

- Map a network share
- Pick a file from that share
- Drop the file as Hardlink within the same share

A Hardlink has been created, which can be easily verified



Remote Capabilities

Furthermore SMB1.0 also reports the reference count for Hardlinks and the junction origin for Junctions, which enables Link Shell Extension to show the properties dialog for remote files. Currently the reference count of a hardlink is reported via SMB1.0 in 90% correctly, so please be aware of this restriction.

LSE supports both, mapped network drives and UNC paths.

Mapped but not available network drives can in general be the reason for sloppy explorer startup performance. Delays of a few seconds can be experienced if explorer has to check all drive mappings, especially the ones which are not available. This gets worse, if LSE also checks the status of all drives.

To workaround this caveat the Remote Capabilities of Link Shell Extension can be switched on/off via the [configuration](#) tool.

Remote Hardlinks and the SMB version

There are different SMB Versions implemented in different Windows versions which means different behaviour for hardlinks:

SMB1.0: Windows XP, Windows2000 ...

SMB2.0: Windows Vista ...

SMB2.1: Windows7/8, Windows Server 2008 R2 ...

SMB3.0: Windows10, Windows Server 2012 R2 ...

All of those versions support the creation of hardlinks remotely, but since SMB2.0 one can not find out if a file on a remote drive is a hardlink or not.

This means, that e.g. if you connect with your Windows XP machine to a SMB2.1 drive, which is provided by a Windows7/8/10 machine, you will not be able to see overlay icons for hardlinked files, but you will be able to create them remotely.

LSE supports removable media, which have been formatted with NTFS, to create all kind of features it does for fixed drives too. The only limitation is that it intentionally won't work on removable media if they are mounted to drive A: or B:. The reason is that A: or B: are commonly used for floppy drives.

Removeable Media

With removable media formatted to NTFS there is the slight chance that LSE reports 'Access denied' problems, when creating hardlinks or junctions. This is due to file object permissions on the removable NTFS drive, which have been created with a different computer on that removable media, thus causing this 'Access denied' messages. The solution here is to change the permission on that removable media as Administrator.

Explorer supports pathnames up to 256 characters, thus limiting all applications to that length for pathnames.

Very long Path

On the other hand NTFS supports pathnames with up to 32767 characters, so one might have already experienced pathnames, which are longer than 256 characters. To deal with that, LSE can handle *Very Long Path* up to 32767 characters with all operations.

Subst Handling

With the subst.exe command one can create driveletters, which point to a certain path on a NTFS volume. This means that two different driveletters in the end might resolve to locations on the same NTFS volume. Link Shell Extension checks these situations when it comes to allow the creation of hardlinks, and as a consequence allows the creation of hardlinks among different logical drives if they resolve to the same NTFS volume.

With Windows Server 2012 Microsoft introduced the [ReFS](#) filesystem, which is the designated successor to NTFS. But the first implementation of ReFS can do nice things, but lacks a few important features from NTFS like the Hardlink support. Hardlinks are available with ReFS3.5

Link Shell Extension supports ReFS so that one can create Symbolic Links, Mountpoints, Junctions on ReFS volumes. With ReFS version smaller than 3.5 it will throw an error message if a hardlink is about to be created on a ReFS volume. The impact on Link Shell Extension in detail is:

ReFs Support

- ReFS drives as the destination of [Delorean Copies](#) will for sure fail, at least when creating the second backup in a Delorean set.
- [SmartCopy/SmartMirror](#) will fail as destination for [hardlinks](#) within the source.
- SmartMove will work.

Since ReFS is expected to support the full set of NTFS functionality in a later release, Link Shell Extension has no checks implemented to tackle with the above constraints.

There are a lot of filesystems out by third party vendors nowadays, which support hardlinks, symlinks... In order to provide the LSE functionality on that drives, the supported filesystems can be configured: Add your favourite filesystem name in a comma separated list to

HKEY_LOCAL_MACHINE\SOFTWARE\LinkShellExtension\ThirdPartyFileSystems

Changes to the known filesystems from the above registry key take effect after an explorer.exe restart.

Third Party FileSystems

If you don't know the name of the filesystem you might determine this by issuing [ln.exe](#) from a command prompt.

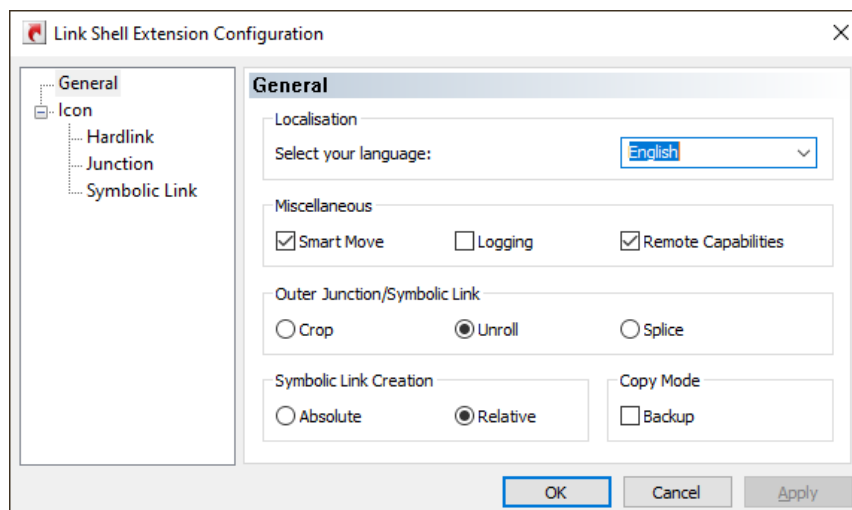
In --probe fs x:

By default *Btrfs* is configured as a known third party filesystem.

Configuring your favourite filesystem to be recognized by LSE is on your own risk. Basically LSE would do all operations to the configured filesystems, which it does to NTFS. So make sure your filesystem supports the same primitives as NTFS does otherwise certain operations will fail.

Configuration

Link Shell Extension can be tweaked/configured to fit the different personal taste in some aspects. To ease this, Link Shell Extension has a companion called LSEConfig, which changes Link Shell Extension behaviour via a User Interface. Once started, LSEConfig will throw the [famous UAC](#) UAC dialog, because Link Shell Extensions settings are changed in the Windows registry.



Localisation

Link Shell Extension's UI and commands are available in a few languages. You can choose from

- English(default)
- Chinese
- Czech
- French
- German
- Greek
- Italian
- Japanese
- Korean
- Polish
- Portuguese Brazilian
- Russian
- Slovak
- Spanish
- Swedish
- Turkish
- Ukrainian

Changing the UI Language of Link Shell Extension will need a restart of the explorer once Apply or Ok is pressed.

Smart Move

It might be useful to totally switch off [Smart Move](#), if there are folders with really much folders. This can be achieved by ticking the *Smart Move* checkbox.

Logging

All output of a LSE operation like SmartCopy, SmartMirror, or Delorean Copy is logged to the file %TEMP%\LinkShellExtension.log

Remote Capabilities

It might be useful to totally switch off [Remote Capabilities](#), if there are lots of 'dead network drives' around. This can be achieved by ticking the *Remote Capabilities* checkbox.

Outer Junction/Symbolic Link Handling

Decide whether [Outer Junctions should be handled](#) as *Crop*, be *Unrolled*, which is the default, or *Spliced*.

Symbolic Link Creation

By selecting either *relative* or *absolute* Link Shell Extension will create the [target of Symbolic Links](#) respective.

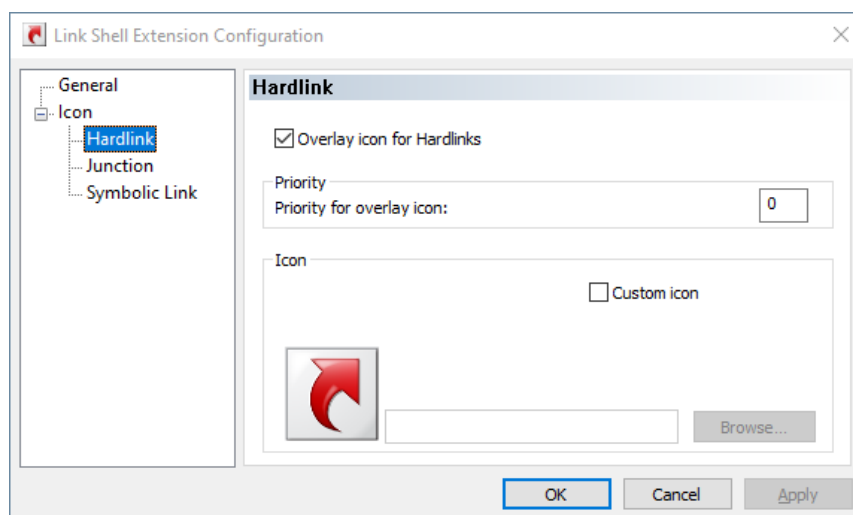
Copy Mode

By checking the *Backup* Link Shell Extension will run in [Backup Mode](#).

Custom Overlay Icons

Link Shell Extension has built in overlay icons for junctions, hardlinks and symbolic links. Since icons are subject to individual taste, the icon used by Link Shell Extension can be specified.

Changing any settings related to overlay icons will ask for a restart of explorer when Apply or Ok is pressed.



Overlay Icon

Sometimes it might be useful to totally disable certain overlay icons from Link Shell Extension, which can be achieved by ticking the checkbox for *overlay icons*.

Priority

Only one overlay icon can be shown with an icon, but many overlay handlers might apply to provide the overlay icon. To sort this out each overlay handler can specify a priority to explorer and explorer shows the overlay icon with highest

priority.

High priority means lower number, with 0 as the highest priority

Custom Icon

By ticking the checkbox for *custom icon* the *Browse...* button gets enabled, and an icon can be selected. Keep in mind that custom icons are specific to each user.

General

Windows 7/8/10 are a little bit special, because overlay icons for 256x256 must not be in the lower left corner of the icon, and must not be already smaller to perfectly 'overlay' an icon. 256x256 overlay icons must fill up the complete available icon, and also must not be resized.

Or in other words Windows 7/8/10 takes any 256x256 icon and resizes it to 92x92, moves it to the left lower corner and overlays.

For all other resolutions smaller than 256x256, Windows 7/8/10 you have to prepare an overlay icon in the lower left corner.

For my investigations the icon editor of choice capable of dealing with Windows 7/8/10 icons was [RealWorld Icon Editor](#)

Apply Changes

When you press OK or Apply on LSEConfigs dialog, settings will be taken over.

If changes were made to Link Shell Extensions language settings or settings related to overlay icons, you will be asked to confirm a restart of explorer.exe, so that your changes become effective. Restarting explorer.exe means, that e.g. any copy operation or other pending operation within explorer.exe is interrupted.

Hardlinks are a feature common to many Unix based systems, but are not directly available with Windows. It is a feature, which must be supported by the file system of the operating system.

So what are Hardlinks? It is common to think of a file as being an association between a *file name* and a *data object*. Using Windows Explorer, the file system can be readily browsed, showing a 1:1 relationship between the *file name* and the *data object*, but this 1:1 relationship does not hold for all file systems.

Some file systems, including UFS, XFS, and NTFS have a N:1 relationship between *file name* and the *data object*, hence there can be more than one directory entry for a file.

So, how does one create multiple entries for the same data object? In Unix there is a command line utility *ln*, which is used to create link entries for existing files, hence there are many file names, or so called Hardlinks, for the one data object.

For each HardLink created, the file system increments a reference count stored with the *data object*, i.e. it stores how many *file names* refer to the data object, this counter is maintained (by the file system) within the data object itself. When a file name referencing a *data object* is deleted, the *data object's* reference count is decremented by one. The *data object* itself **only** gets deleted when the reference count is decremented to zero.

The reference count is the only way of determining whether there are multiple *file name* references to a *data object*, and it only informs of their number NOT there whereabouts.

Junctions are wormholes in the tree structure of a directed graph. By browsing a Junction a maybe far distant location in the file system is made available. Modifying, Creating, Renaming and Deleting files within a junction tree structure operates at the junction target, i.e. if you delete a file in a Junction it is deleted at the original location.

Symbolic Links are to files what Junctions are to folders in that they are both transparent and Symbolic. Transparency means that an application can access them just as they would any other file, Symbolism means that the data objects can reside on any available volume, i.e. they are not limited to a single volume like Hardlinks. Symbolic Links differ from Shortcuts in that they offer a transparent pathway to the desired data object, with a shortcut (.lnk), something has to read and interpret the content of the shortcut file and then open the file that it references (i.e. it is a two step process). When an application uses a symlink it gains immediate access to the data object referenced by the symlink (i.e. it is a one step process).

Backgrounders

Limitations

- Supported platforms are NT4/W2K/WXP/W2K3/W2K3R2/W2K8/W2K8R2/W2K12/W2K12R2/WXP64/Vista/Vista/Windows 7/8/10 in 32bit, 64bit or Itanium.
- Hardlinks can only be made on NTFS volumes, under the supported platforms.
- Hardlinks can only be made within one NTFS volumes, and can not span across NTFS volumes.
- Junctions can not be created on NTFS volumes with NT4.
- The *Pick Link Source* and *Drop ...* choices are only visible, if it's possible to create Hardlinks/Junctions/Symbolic Links. E.G.: If you select a file on a FAT drive and press the action button, you won't see the *Pick Link Source* in the action menu, because FAT file systems, don't support Hardlinks/Junctions/Symbolic Links. This also happens, if you select source files on a network drive, or select a file as destination, etc.
- There is an OS limit of creating more than 1023 hardlinks per file. This is less known, but it is there.
- [ReFs](#) does not support hardlinks.

Frequently Asked Questions

- **Q: On Windows 7/8/10, the Save As... box shows symlinks with the white "shortcut" overlay, instead of the green symlink overlay.**

A: This happens if the processes shown during installation of Link Shell Extension were not closed. If you really run into this rare situation, a reboot will help.

- **Q: However the value of the reference count is not updated when hardlinks are deleted. That is, when I add new hardlinks the value increases properly, but when I delete hardlinks, the value does not change. Is that a bug? Or there is a way of refreshing the Windows Explorer?**

A: Once a file is deleted in Explorer it is moved into Recycle Bin, but not really deleted. If you press Shift-Del for deleting a file instead of just pressing Del, the file really gets deleted and the reference count is decremented.

- **Q: I could'nt make a successful hardlink for image or vector files - I mean, I was able to *make* the hardlink copy, but when I modified one file it didnt affect the other. I'm wondering do you know why this might be - could it be my otherwise quite normal computer (!) or could it be something to do with the hard link process ?**

A: You were able to make hardlinks successfully, but when you open a hardlinked file for **edit**, it depends on the editor associated to the file if the file either gets

- opened, changed, the original deleted, and the new one saved (==> link broken)
- opened, changed, and saved back (==> link alive)

- **Q: When I deleted a source directory, its junction point is left behind in a non-operational state. Is there a way to prevent this? That is, for example, is it possible to automatically delete the junction points if the associated source is deleted? Or, is it possible to have a program prune such orphaned junctions afterwards?**

A: No sorry, Junctions are a one way relation, and if the targets disappears the junction points to an orphaned destination.

If you have [SmartMove](#) enabled, at least [inner junctions/symbolic links](#) are adapted

- **Q: When I delete a symbolic link, which points to a zipped folder by pressing DEL, later on when I want to empty the recycle bin, explorer denies by showing me error message 0x80071128. What's wrong?**

A: Unfortunatley this is a bug in Explorer, and it only happens to symbolic links pointing to .zip files. The workaround is to move it manually out of recycle bin rename it, and then delete it once more.

- **Q: I have created a symbolic link to an .exe and when I double click on it, I get the following error message: *The specified path does not exist. Check the path and try again.***

A: Unfortunatley this is a bug in Explorer, and I don't have a clue how to come around this in explorer. If you start the symlink to an .exe from a command prompt it works fine, and even third party explorers like [SpeedCommander](#) can do this, but explorer seems to have a limitation Does anybody know the registry hack to enable this in explorer.exe? Drop me a line.

- **Q: I double click on a symlink in explorer, which e.g. points to an .xls, and the explorer asks me to choose a program to open it.**

A: With KB3039066 Microsoft changed the behaviour of symlinks. Uninstall it and it will work again. See also [Symbolic Link Type Changed](#)

- **Q: The overlay icons do not show up**

A: The number of different icon overlay handlers that the system can support is limited by the amount of space available for icon overlays in the system image list. There are currently 11 slots allotted with Windows 10 for icon overlays, some of which are reserved by the system.

All is controlled by the alphabetical order of OverlayHandlers under

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ShellIconOverlayIdentifiers

If the OverlayHandlers for LinkShellExtension somehow slipped to a slot greater 15 under 32bit Windows or greater 11 with Windows 10, the LSE Overlay Icons won't show up.

To manually boost the priority of LSE OverlayIcons, open the above registry location with regedit and prepend a blank

```
IconOverlayHardLink --> 'IconOverlayHardLink
IconOverlayJunction --> 'IconOverlayJunction
IconOverlaySymbolicLink --> 'IconOverlaySymbolicLink
```

and either restart the explorer or log-off and log-on again. The point here is to change the alphabetical order by prepending one blank. You might end up in a race to the top by adding more and more blanks, since other OverlayHandlers such as [DropBox](#) have already added a few blanks.

- **Q: I'm trying to store Dropbox files only on removable storage instead of the internal 128gb of storage. My DropBox already contains lots of files. How do I accomplish redirecting the dropbox folder to the removable**

storage?

A:

- Copy the whole dropbox folder from the internal storage under c:\users\[username]\dropbox to e.g x:\data\dropbox
- Rename the dropbox folder c:\users\[username]\dropbox to e.g c:\users\[username]\dropbox_org
- Pick Link Source x:\data\dropbox
- In c:\users\[username] use *Drop as* and select *Symbolic Link* or *Junction*
- If everything went fine finally delete c:\users\[username]\dropbox_org
- **Q: When I create Symbolic links they appear in Explorer as 0 bytes. I cannot remember if this is expected or not?**

A: The resulting Symbolic Links show as 0 bytes in explorer.exe, that's expected.

- **Q: I only changed the attribute of a file, even the timestamp and content are the same, but --delorean copies the file instead of linking to the old backupsets and just changing the attributes in the current backup set.**

A: The files in the backup sets of --delorean are hardlinked if they are the same. NTFS provides *one set* of timestamps and attributes for *all* hardlink-siblings of a file, so if a file needs different attributes in a backup, it has to be copied.

- **Q: Broken Junctions (with non-existent targets) have their overlay icons displayed, but right clicking on them in Windows Explorer of Windows7 x64 SP1 and choosing "Properties" does not bring up the "Link Properties" tab. Consequently the information where the broken Junction is trying to point is not accessible and cannot be corrected manually.**

A: Unfortunatley this is an explorer problem, and LSE has no chance to intercept. Broken symlinks/juncitons can be easily repaired by [the replacement junction/symbolic link](#) feature

- **Q: When I right click a symbolic link and go to properties, and click on Link properties, if the path contained inside is invalid, will give you the message "The name '(invalid path goes here)' specified in the Target box is not valid. Make sure the path and file name are correc.**

A: Unfortunatley this is an explorer problem, and LSE has no chance to intercept. Broken symlinks/juncitons can be easily repaired by [the replacement junction/symbolic link](#) feature**History**

Version 4.0.0.0

in progress

- Complete support for UNC path in any Smart Move/Copy/Clone & Delorean Operation.
- Lots of little fixes/improvements, I always wanted to do, but never had the time to.

Version 3.9.3.5

January 16,
2021

- In LSE now maintains its [winget package](#).
- With SE_CREATE_SYMBOLICLINK assigned to a user, UAC dialog was raised anytime after second attempt.
- German Translation of Documentation.
- Pressing F1 in LSEConfig crashed it. Fixed.
- During Hardlink/Symbolic Link Clone pressing Cancel didn't have an impact.
- Progress bar prediction has been improved.
- Fixed crash when username was > 20 chars as non-admin during symbolic link drop.
- Link ShellExtension now [maintains](#) its [chocolatey package](#).
- Link ShellExtension is public on [gitlab](#).

November 8, Version 3.9.3.0
2019

- For non Admin users under Windows 10, the explorer did not restart after install
- During custom icon change one was asked to close lots of dependant apps, which is unnecessary. Reboot Explorer.exe is enough.
- Lots of strings very wrong. Introduced with 3.9.2.8
- Removed 'Delete Junction' from main menu.
- Improved menu hydraulics, e.g. once can not create Smart Copies of Symbolic Link Files.
- Adapted check for Redistributables to VS2017 and Windows 10.
- On non NTFS drives, files could not be selected as Link Source. Introduced with 3921
- Unmount volume did not work, when elevation was needed.
- New Windows 10 screenshots for docu and docu streamlining.
- Signed with a [standard code signing certificate](#)
- During Replace Junction/Symbolic Link/Mountpoint the original attributes got lost
- The Link Properties Tab showed relative symbolic link dirs as absolute
- The Link Properties Tab did not show up on dangling Junctions
- Creation of file symbolic links didn't work in protected folders like 'Program Files' with [Developer Mode](#) on.
- Replacement Junction/Symbolic Link didn't work on dangling Junction/Symbolic Link.
- UChelper mixed up relative and absolute link creation. Introduced with 3920
- Added Ukranian localization, Finished localisation for missing texts

- Cross Drive Drops didn't work. Introduced with 3921
- Properties Dialog could cause a crash on changing junction targets
- Autorename had a flaw when having more than 9 files with same name in a directory
- Backup mode was broken. Introduced with 3921
- Introduce [Copy Symbolic Link/Junction/Mountpoint](#)
- Changes of non-admin users to e.g. symlink relativ/absolute had no effect. Fixed the misconception. What a mess
- LSE is on gitlab.com. For now only private, but hope to change this soon.
- SmartMirror in LSE had problem with nested repare points
- The symbolic replacement mechanism was broken for the elevated use-case. Caused by 3.8.7.2
- Improved the progress estimation for the progressbar and introduced the [Windows7 progress dialog](#)

Version 3.9.0.2

December 28, 2018

- Improved the progress estimation for the progressbar
- Fixed crash in LSEConfig when it replaced the texts upon startup
- Removed vcredist-vs2005 check from installer
- [Internal] but important change from VS2005 (sic) to VS2017. Basically everything compiled smoothly except for the heap, thus...
- [Internal] Removed the Rockall fast heap. This was necessary, but also a big performance gain. Memory allocation is 2 times faster, and memory deletion is 10 times faster. Memory allocation is crucial for the core of ln.exe and LSE.
- [Internal] Dropped Itanium configuration, since VS2017 does not support it anymore, and I am sure there is no Itanium hardware out in the wild anymore.

Version 3.8.7.2

December 2, 2018

- The overlay icons didn't show up under 32bit applications.
- With Windows 10 LSE shows flat icons for symlinks, hardlinks and junctions. Thx to Yaroslav for the icons.
- The overlay icon for Junctions didn't show up at all with Windows 7.
- Symbolic Link creation is now possible unelevated for [Windows 10/14972](#) when in developer mode.
- The property dialog for hardlinks, when used on substituted drives, showed wrong path for the siblings.
- [Third party filesystems](#) can be configured.

Version 3.8.6.8

April 19, 2016

- Relative symbolic links on absolute symbolic links to a different drive, were not created properly.
- Added Greek localization. Thanks to George Malamas!
- SmartMove didn't work when Backup Mode was enabled.
- Runs again with Windows 2000 (Flaw introduced with 3.8.6.6).
- The Vcredist check is back by default, but can be skipped by passing [/noredist](#) during install via commandline.
- Relative symbolic links on UNC path were recreated with wrong target during SmartCopy.
- NTFS filesystem fragmentation decreased and thus copying should be faster
- Cancel works when pressed during copying of (large) files.
- Creation of hardlinks in UAC protected folders caused an error message, even if the hardlink was created.

Version 3.8.6.6

January 10, 2016

- Files with a size beeing a multiple of 16777216 were copied properly, but in the logfile generated an error message even if there was none.
- The progress bar didn't increase properly during Smartmirror if the files were the same.
- In a rare situation only the first symbolic link of a smart-copy was linked incorrectly.
- The [Auto Rename](#) functionality didn't work in rare situations. Fixed for XP and Windows7/8/10
- Adaptions for Windows 10 Tech Preview 9841/9926.
- Fixed a crash, when files were dropped from Bandzip to the desktop.
- Added localization for Korean. Thanks to Mireado from Korea!
- Fixed a crash in W10 when closing the properties dialog on junctions/symlinks.
- On Windows10 systems which have been upgraded from Windows7 the enumeration of hardlinks siblings via the property page takes very long.
- Prerequisites install not needed with Windows 10, thus do not check for prerequisites under Windows 10.
- Prerequisites still checked by installer with Windows10.

June 3, 2014 Version 3.8.5.1

- The [Backup Mode](#) arrived to LSE.
- LSE produced logfiles with LF but not CR/LF.
- Junctions/Symbolic Links/Mountpoints ACLs are preserved when the target is changed either via [Replacement](#) functions or editing from the Link Properties.
- Japanese translation for [LSEConfig](#).
- The target of mountpoints can be changed from [Link Properties](#).
- LSE now can handle [Mountpoints](#) during SmartXXX/Delorean operations.
- WindowsXP: Symbolic links across drives didn't work.
- WindowsXP: Symbolic links to Volume GUIDs didn't work.
- WindowsXP: Symbolic links between [very long path](#) didn't work.
- WindowsXP: Lots of tweaks here and there.

- UNC path as Link Source for SmartCopy/Mirror/Delorean/SymbolicLinkClone now work.

Version 3.7.5.9

December 29, 2013

- With Delorean Copy when elevated via LSEUacHelper.exe in rare situations files pointed to by symlinks could get deleted in the source. Ugly. Sorry!
- A changed [file attribute](#) didn't cause the file to be treated as changed during all SmartXXX/Delorean functions.
- Symbolic Link Clones always created absolute symlinks regardless of the LSE Settings
- The LastWriteTime, CreationTime and LastAccessTime for files/folders/junctions/symlinks is restored during [SmartMirror](#) or [DeloreanCopy](#).

Version 3.7.5.1

August 23, 2013

- Dead Junctions to a different drive could lead to not detecting hardlinks during all operations. Very Nasty, but no dataloss caused.

Version 3.7.5.0

August 4th, 2013

- For Junctions or Symbolic Links the Target field can be edited in the [Properties Dialog](#).
- LSE elevates the creation of hardlinks in system protected directories, e.g.: %systemroot%.
- LSE now offers all its function also in the *Library* folder.
- If [enabled](#) LSE summarizes the output of operations in a log file.
- Fixed a crash related to UNC path and Overlay Icons.
- The progress bar showed wrong/incomplete filename-path combinations during operations on large files.
- Symbolic links to mapped network drives can be created now via LSE.
- [Replacement](#) functions can be used to repair broken junctions/symbolic links/mountpoints.
- During SmartCopy/SmartMirror/DeloreanCopy the type (absolute/relative) of symbolic link relation is kept in the destination.
- During SmartCopy/SmartMirror/DeloreanCopy the Compression Attribute is copied too.
- The check for prerequisites during install is more accurate aka takes mfc80.dll into account.
- During un-install it is also checked if the hardlinkshelltext.dll is held by some process.
- Fixed a problem during SmartMirror when a directory changed into a file or vice versa from one mirror to the next and had exactly the same name.
- The OS Version detection during installation went wrong on certain machines causing symmlinke.exe missing from the installation.
- The enumeration of hardlink siblings didn't work under XP, when the root dir of a drive had to be traversed.
- Tested with Windows8, and thus updated the documentation.
- The x64 version now contains also a 32bit version in one unified install.
- Deinstallation left over a few registry keys.
- During installation not all processes were detected, which blocked the installation.
- Enabled Link Shell Extension on [ReFs](#) volumes.
- Added a Swedish localisation. Thanks to Mikael Grönholm.
- Added a Turkish localisation. Thanks to Memet.
- Added a Czech localisation. Thanks to Ashus
- Added a Slovak localisation. Thanks to RobertS
- LSEConfig has been localised.
- The handling of the compression bit during copying/mirroring/deloreaning for files and directories was broken.
- Dragging from or Dropping to zipped folders caused an explorer crash.
- LSEConfig localized to French.
- Columnprovider shows now shrinked path for junctions if the path is longer than 32characters.
- On some machines LSEConfig always showed up with French. Introduced with 3.749

June 24th, 2012

Version 3.7.2.0

- When working on mapped network drives via SMB or CFIS, as many NAS boxes do, LSE uses a more traditional enumeration mode and this will copy files (which it did not in any case).
- [Multiple locations](#) can be selected and the location are treated as a common root with respect to hardlinks/junctions/symbolic links.
- Nested junctions and symbolic links (aka junctions on junction on junctions ...) are now properly restored in any situation.
- Smartmove had problems with relative symbolic links in rare situations.
- Italian translation updated.
- Support for symbolic links under Window XP.
- Can handle [subst.exe created](#) driveletters.
- Overlay Icons for symbolic links under Window XP are available now.
- Fixed a few bugs related to WindowsXP and symbolic links handling.
- Elevation to LSEUacHelper.exe now happens only if [UAC](#) is on, or the elevation is really necessary.
- Fixed a problem with the creation of absolute symbolic links to directories.
- The installer came up with Chinese as default installation language.
- LSEConfig has an About Box, which shows the version of Link Shellextension.
- Replace Symbolic Link failed when not elevated.
- Replace Symbolic Link always created absolute symbolic links regardless of the settings when not elevated.
- Drop Symbolic Link sometimes did not create absolute links when needed in certain situations.

- The installer now shows the language of the installed OS as default.
- The installer provides more info in Control Panel/Program and Features.
- The target of the Symbolic Links or Junctions can be edited in the [properties dialog](#).
- Non administrators could not create symbolic links.
- With XP and the symlink driver installed the Delete Junction menu didn't show up.
- The status of the [privilege for Symbolic Link Creation](#) is checked, so that UAC can be avoided
- Fixed deployment problems for the Win32bit version.

Version 3.6.5.3

March 9th
2012

- Speed improvements during SmartCopy/SmartMirror/HardlinkClone and Delorean Copy.
- Introduced new Heap Manager Rockall for x64 and x86 builds to gain performance.
- Russian translation updated.
- Installation notifies about already running processes, which would make LSE installation fail, because they have loaded dlls from LSE.
- Fixed issue with Windows 8 installation.
- Symbolic Link Icon Overlays installation in registry was wrong, causing problems with the green arrow for symlinks.

Version 3.6.0.4

April 17th
2011

- With all Smart* functionality, Outer Junctions/Symbolic Links can now be [unrolled or spliced](#).
- Added [Smart Mirror](#).
- Speed improvements to Smart Copy, Smart Move and Delorean Copy.
- The [configuration tool](#) does not restart explorer for minor changes to the settings.
- Symlinks creation resulted in absolute symlinks even if creation of symlinks was specified as relative, if their common ancestor was a root dir.

Version 3.5.0.1

November
21st 2010

- Introduced [DeLorean Copy](#), which is a way of creating incremental copies using hardlinks.
- Fixed creating Symbolic Links from Symbolic Link files created Symbolic Link directories.

Version 3.4.0.2

October 3rd
2010

- LSE now by default creates [relative](#) target path names when creating symbolic links.
- Symbolic Links now have an [overlay icon](#).
- Added a [configuration](#) tool for LSE options
- Added a [priority level](#) for the individual overlay icons.
- Added an [option](#) to switch off overlay icons totally for each type of overlay.
- The Pick Link Source context menu now also shows up on FAT drives, if the item potentially is the source for a LSE operation.
- Overlay icons can be [disabled](#).
- Documented how the [install directory](#) can be specified when using silent (un)install.
- The menu hydraulics have been reworked, so that it is decided early to only show menu for choices, which are really possible.
- LSE and symlink are now linked with ASLR.
- Transparency glitches in the Junction overlay icon have been fixed.
- Hardlink Clone and Symboliclink Clone have been extended so that [Inner Junctions and inner Symbolic Links](#) are properly handled.
- LSE now supports also the replacement of [Mountpoints and Symbolic Links](#).
- LSE shows a dialog box whether explorer should be [restarted](#) or not during non silent installation.
- Under Windows Vista and Windows 7 the *Delete Junction* menu does not show up anymore.

Version 3.3.5.8

July 19th
2010

- LSE now deals with Symboliclinks during [Smart Copy](#).
- LSE supports [Smart Move](#) functionality, which updates inner junctions/symlinks in case of moving/renaming directories
- Added localization for Brazilian Portuguese. Thanks to Marcio R. for the translation.
- Fixed flaws of the [automatically renaming](#) feature with respect to directories under W7.
- Overriding custom overlay icons under HKCU was flawed.
- Hardlink Clone now restores the attributes of cloned folders.
- Smart Move progress bar showed a wrong caption text.
- Added Polish localisation, Thanks to Arthur from Poland.
- Fixed a crash during undeleting files from Recycle Bin.
- Mountpoints could not be properly created under Windows XP.
- gFlags was not properly read from the HKCU registry, causing *Smart Move Disable* and *Remote capabilities Disable* to malfunction.
- Fixed the problem of Hardlink Clone stopping unsolicitedly after about 500msec.

February
21st 2010

Version 3.2.2.4

- Added localization for Chinese and Russian. Thanks to Zuo Weiming and Ivan(b0s) for the translations.
- Longer lasting operations, like [Smart Copy](#), [Symbolic Link Clone](#), [Hardlink Clone](#), or [Enumerate Siblings](#) show a progress bar.
- The [Properties](#) dialog of an item offers an 'Explore Target' button for Junctions, Mountpoints and Symbolic Links.
- Added localization for Japanese. Thanks to Taka from Japan!

- Under Windows7 [auto rename](#) behaves in the same way as Windows7/8/10 does for '- Copy'.

Version 3.1.6.0

September
28th 2009

- With W2K the junction creation was broken.
- Fixed a handle leak caused by enumerating hardlink siblings under non Vista/W2K8
- Under Vista & W7 one can create junctions everywhere without elevation, but not in e.g c:\Program Files. LSE is now aware of this and asks elevation for junction creation when necessary.
- With drives mapped via a Remote Desktop session, the whole explorer & remote desktop session hang, when this drive was expanded in explorer, but only under W2K3 as terminal server.
- Support for Windows7
- [Hardlink Sibling Enumeration](#) now also works for XP, W2K NT4, but due to OS constraints not that fast as with Windows7.
- Under W2K it turned out, that CreateHardlink() from kernel32.dll with long pathnames (e.g. \\?) was broken.
- Fixed a memory leak in serving as COM server.

Version 3.0.0.1

October 4th
2008

- There is a new [Smart Copy](#) feature, which enables LSE to copy whole folder structures and preserve the inner hardlink and junction structure.
- [Very Long Pathname](#) support added for Smart Copy and Hardlink Clone.
- Junctions can now be created targeting also Junctions.

Version 2.9.5.3

June 21st
2008

- Hardlinks can be [enumerated](#) under Vista & Windows7.
- Fixed a handle leak for HKCU\Software\LinkShellExtension.
- Removeable media support didn't work, when remote capabilities were switched off.

Version 2.9.0.3

May 1st
2008

- Already existing Junctions can be [replaced](#) by dragging a directory over it
- Naming has been streamlined more towards 'Link Shell Extension'
- If the maximum number 1023 of hardlinks for a file is exceeded, an error message is displayed. This applies for hardlinks and hardlink clones.
- The Vista & Windows7 junction overlay icon in 256x256 is in proper size.
- [Custom icons](#) can be specified for Junction and Hardlink overlays
- Version for Itanium available
- Pick/Drop does not interfere with the creation of Hardlinks, Junctions or Symbolic Links via Drag and Drop. It is now possible to pick a link, then drag another file via right mouse click to some location, drop it there and afterwards drop the first, picked file
- The property dialog of a Volume Mountpoint now displays the logical drive letter of the mounted drive instead of the odd volume name.
- Ongoing work towards localisation to East Asian languages.
- LSE now also works on [removable NTFS media](#), which are not A: or B:
- The [location in the registry](#) to specify the LSE language has changed, since the old place under HKCR was not Vista & Windows7 compatible at all.
- Default values for language settings and overlay icons settings are copied over automatically to freshly logged on users profile
- [Volume Mountpoint](#) support for Vista & Windows7.
- Introduced [silent install](#) capabilities.
- Symbolic Links now can be created even if the filename contains UTF-16(Asian)characters.
- LSE now also works for non Administrators under Vista & Windows7 (after they acknowledged the elevation dialog with the admin password for sure).
- Symbolic Links for files or directories can now be created across volumes.
- LSE now can also create Hardlinks from Shortcuts, which didn't work for ages.
- The print name (the name some can see to the right of a junction, after issuing 'dir' in a command prompt) for Junctions under Vista & Windows7 is now correct.
- Lots of usability fixes
- During Installation under Vista64 Explorer automatically gets restarted.
- The setup contains a check if the VS2005 SP1 Redistributable Package is installed.
- The setup contains a check if the LSE version for the proper platform is going to be installed.
- Vista & Windows7 compliant [overlay](#) icons for Hardlinks.

Version 2.8.0.6

January 20th
2008

- Hardlinks show up with a small overlay icon. This icon will change for Vista compliancy, but at least it is here now.
- Junctions have a Vista compliant overlay icon in many resolutions
- Support of creation and deletion of [Volume Mountpoints](#). Unfortunately this does not work under Vista
- LSE now prevents the creation of 'loops', when setting up a junction or hardlink clone
- Hardlink 'cross drive drops' are now not possible anymore
- Some bug fixes for NT4
- Lots of usability fixes

October 16th Version 2.7.1.0
2007

- First fixes for the x64 world. Maybe more to come. Fixing x64 is top priority since I have Q6600 myself now...

Version 2.7.0.1

March 25th
2007

- Fixed a nasty bug, which caused HardlinkShellExt to slow down explorer, when it was started. Also fixed the problem that it accessed drive A:, when an explorer started.
- PropertySheet on file and directory properties will show various info with W2K/XP
- *Delete Junction* is back, because sometimes, especially when deleting junctions, which point to directories with big amount of data, the Copyhook Handler does not act as expected. Until this phenomenon is solved, *Delete Junction* is back.

Version 2.6.0

January 12th
2007

- Link Shell Extension is now robust with respect to deleting junctions. Delete commands issued from explorer unlink junctions, but do not delete its content.
- Due to Junction-awareness of explorer the *Delete Junction* is gone from the context menu
- Support for Windows Vista & Windows7. Link Shell Extension is now capable of creating Symbolic Links, but also has a few restrictions related to the Reference column

Version 2.5.1 released

December
27th 2006

- Added localisation for Italian and Spanish to commands and messages. Thanks to Nicola Guidotto and Diego Segobia for the translations.

Version 2.4.0 released

December
6th 2006

- Added localisation for French and German to commands and messages

Version 2.3.0 released

November
26th 2006

- Minor fixes in the installer/deinstaller
- Introduced the hydraulics of multiple auto rename. If you drop links/junctions in the same directory, now it behaves exactly as explorer and puts numbers on the [multiple instantiations](#) of a file.

Version 2.2.2 released

June 16th
2006

- BugFix. Link Shell Extension is now also able to create hardlinks via Drag and Drop in root dirs of a drive.

Version 2.2.1 released

May 29th
2006

- Updated documentation after review of [Philip Daniels](#)
- Made a big step forward to being Vista compliant
- Fixed path length limitations in several places
- Junctions can span over local NTFS volumes

Version 2.1 released

March 14th
2006

- Added overlay icons for junctions, so that junction visually pop into your eye.

Version 2.0 released

February
27th 2006

- Revamped the internal structure of ShellExt.
- Introduced creation of Hardlink Clones.
- Introduced submenu in the context menu, when more than one entries would be added to context menu to show the many dropping choices
- Support for SymbolicLinks with 'Vista'.
- Support for SymbolicLink Clones with 'Vista'.
- Fixed crashes when dragging files, and using 'HardLink here'.
- Fixed problem when showing up wrong menu, when having folders disabled in the left explorer pane.
- Junctions display their origin in the reference column.
- A *Pick Link* operation can be cancelled now.
- The installer restarts explorer.exe to properly add/remove the shell extension
- Added an entry to Start Menu/Programs
- Support for WindowsXP64.

Version 1.7 released

November
26th 2005

- Added the *Delete Junction* context menu, when right mouse button is pressed on a junction.
- Fixed a handle leak in CreateJunction.

Version 1.6 released

January 23rd
2002

- Added a Columnhandler, so that the reference count of a hardlinked file is shown in explorer. This feature only works with W2K/WXP.
- Deployment revamped so that the doc is now in .html.

October 27th
2001

Version 1.5 released

- Revamped internal string handling to Unicode.
- Added junction support. Junctions are a feature of NTFS5, which allows to hardlink two directories.
- Added a directory background handler. This means, that after picking a hardlink it is possible to press the right mouse button on the right explorer pane and drop the hardlinks/junctions/symbolic-links.

Version 1.201 released

March 23rd
2001

- Fixed occurrence of 'Hardlink Here' if shortcuts are selected.

Version 1.20 released

March 23rd
2001

- Added Drag and drop support

Version 1.10 released

March 20th
2001

- Fixed the problem, that the help text was not displayed properly
- Changed the installer to the lean and mean nullsoft installer.
- Fixed the problem, that read-only files can not be hardlinked
- Fixed the problem, that hardlinks in the root dir didn't work
- Tested on W2K and HardlinkShellExt is W2K compliant

May 8th
1999

Version 1.00 released

Status

The 3.9.3.x version is a stable version for the [supported platforms](#).

I wish to thank those who have contributed significantly to the development of Link Shell Extension. Those include:

Acknowledgements

[Felix Kasza](#) for the [hardlink basics with NT4](#).
 Nullsoft for the great lean and mean [nsis installer](#)
[Jean-Pierre Bergamin](#) for the [drag and drop support samples](#).
[Travis Illig](#) suggested to add the overlay icons for junctions, which he uses in his [Junction Shell Extension](#).
[Mark Russinovich](#) for tips on [junction](#)
[Philip Daniels](#) for a technical writers documentation review
[Daniel Thibault](#) for the French localisation, and for a dozen of bug reports and feature requests.
[Masatoshi Kimura](#) for the symbolic link driver for WindowsXP.
 Gerard Durand for the French translation of the documentation

Open Issues

- With Vista & Windows7 the column handler in explorer, providing the reference count, does not work, since Microsoft deprecated the interfaces with respect to this functionality.

License

- This program is provided as is. See [license.txt](#) from this distribution for legal issues.
- Link Shellextension uses [tre](#) as the regular expression machine. See the [tre license](#).

Contact /
Donations

Bug reports, or feature requests send to [Hermann Schinagl](#).
 LSE is and will be freeware, but if LSE was really helpful for you and saved lots of your time please think of donations either via PayPal

Donate

or by flattring me

or by sending me a gift certificate from



or by donating bitcoins:

bc1q4hvevwrnnwt7jg8vws0v8xajywhffl4gwca5av



Link Shellextension also has its page on [Facebook](#), where you can find announcements for new releases, and you can discuss feature requests



Link Shellextension broadcasts its release notes via [RSS](#).



This version contains the 64bit version of Link Shell Extension, but also contains a 32bit version, which is installed in parallel to the 64bit version, to satisfy third party filemanagers/explorers like total commander:

All Windows 64

[Link Shell Extension \(3.76Mb\)](#)

All necessary runtime dlls are already installed on your system, but if not grab it from [here for 64bit](#) and [here for 32bit](#)

Simply download and install

[Link Shell Extension \(3.56Mb\)](#)

All Windows 32

All necessary runtime dlls are already installed on your system, but if not grab it from [here](#).

Download

Link ShellExtension can also be installed via [chocolatey](#) by issuing

Chocolatey Installation

choco install linkshellextension

LSE can also be installed via [winget](#) by issuing

winget install HermannSchinagl.LinkShellExtension

Winget

Installation from a command prompt. Make sure you have [winget installed](#).

Legacy Download

The Itanium version is not supported anymore, but the last VS2005 based version 3.8.7.2 is kept for legacy. Please make sure that the necessary runtime .dlls are installed on your system. This prerequisites package can be downloaded from Microsoft:

[vcredist_IA64.exe for VS2005 SP1, version 6195/June 2011 \(6.3 Mb\)](#)

All Windows Itanium

Afterwards install the

[Link Shell Extension \(3.76Mb\)](#)

The version for Windows NT4 will be no more actively developed on, and its functionality is frozen with **Windows NT4** version [Link Shell Extension 3.2.0.0 \(1.13Mb\)](#), which basically has all the important features.

The version for Windows 2000 and for Windows XP will be no more actively developed on, and its
Windows 2000 functionality is frozen with [Link Shell Extension 3.8.7.2 \(32 bit\)](#) and [Link Shell Extension 3.8.7.2 \(64 bit\)](#).
Windows XP

The driver to enable even WindowsXP with symbolic link functionality is provided courtesy of Masatoshi Kimura. You can download the driver from his homepage or from my site acting as a mirror.

**Symbolic Link
Driver
for Windows
XP**

[Symbolic Drivers for WindowsXP 64 \(86kb\)](#).[\[Original Location\]](#)
[Symbolic Drivers for WindowsXP 64 \(86kb\)](#).[\[Mirror schinagl.priv.at\]](#)

[Symbolic Drivers for WindowsXP \(86kb\)](#).[\[Original Location\]](#)
[Symbolic Drivers for WindowsXP \(86kb\)](#).[\[Mirror schinagl.priv.at\]](#)

[Sources for Drivers \(23kb\)](#).[\[Original Location\]](#)
[Sources for Drivers \(23\)](#).[\[Mirror schinagl.priv.at\]](#)