Adding or unpacking a single file to a target directory

The `<file>` element

The `<file>` tag is a nested element to the <pack> element and specifies a single file to be added to that pack.

You can use `<fileset>` to add multiple files.

Attributes

Attribute	Description	Required	Values (Default)
`src`	the file location (relative path). The src name may contain previously defined static variables (see `<variables>`).	yes	
`targetdir`	the destination directory, could be something like `$INSTALL_PATH/subdirX`	yes	
~~os~~	~~Limit installation of this particular file only to the given target OS type.~~	~~no~~	~~"unix" | "windows" | "mac"~~
`override`	Whether to overwrite existing files. • `true` if the file should always overwrite an existing file with the same name. • `false` if the file should be skipped if the file already exists in the `targetdir`. • `asktrue` if the user should be interactively asked what to do and supply default value of "true" for non-interactive use. • `askfalse` if the user should be interactively asked what to do and supply default value of "false" for non-interactive use. • `update` if the new file is only installed if it's modification time is newer than the modification time of the already existing file. Note that this is not a reliable mechanism for updates - you cannot detect whether a file was altered after installation this way.	• no	"true" | "false" | "asktrue" | "askfalse" | "update" ("update")
overrideRenameTo	Globmapper to rename a conflicting file to. This works similar to the <globmapper> in File Name Mappers, whereby the mapper's from attribute is set to the empty string and the to attribute exactly to the value given here. Example: `overrideRenameTo=".bak"` will rename the target file by appending the suffix .bak instead of overwriting it. The override attribute must be set "true" to activate this feature. Since IzPack 5.0	no	String - valid globmapper target expression

blockable	For Windows only, ignored on non-Windows systems: Defines whether and how blocked target files on Windows should be handled. This might result in pending file operations which require a system reboot. Pending file operations are introduced during the installation in two phases: 1. The `blockable` attribute marks files pending to be replaced after a system reboot if they are blocked. Blocked files are recognized during physically installing them. 2. How and whether the system should be really rebooted directly from the installer in case there are such pending file operations at the end of the installation can be controlled by the `<rebootaction>` tag nested to `<info>`. Notes: Using `blockable` does not necessarily force you to limit such files on Windows systems. For multi-platform installations there is a compiler warning shown that `blockable` will be ignored on non-Windows systems. The native library `WinSetupAPI` must be explicitly included using this feature. Possible values: • **`none`** No recognition of blocked target files will be done at all, this is the default behavior of previous IzPack versions. • **`auto`** Automatic recognition of a blocked target file by the operating system, resulting in leaving a pending file operation to be finished after system reboot. Using `auto` this applies only for files that are really blocked, the other files are copied normally, which can result in mixed, old and new target files at the end of the installation, unless the system won't be really rebooted. • **`force`** Forces target file to be always assumed a blocked, resulting in leaving a pending file operation to be finished after system reboot. Using `force` this applies for each file, regardless whether it is really blocked during installation. This makes sense if you don't want to mix files old and new files at the end of the installation to not disturbing a running process, but having the complete set of target files installed after system reboot. Since IzPack 5.0	no	`"none"` | `"auto"` | `"force"` ("none")
`unpack`	if true and the file is an archive then its contents will be unpacked and added as individual files. Note: Only archives with ZIP compression are supported. This includes jar, war, zip, etc...	no	"true" | "false" ("false")
`condition`	Limit installation of this particular file to the given condition, which must be true during the file installation.	no	String - a valid condition ID
casesensitive	Whether to treat the file name case-sensitive.	no	"true" | "false" ("true")

defaultExcludes	Whether to use global default excludes.
	Implicit default exclude patterns are typically:
	`*/~{}`
	`*/##`
	`*/.#`
	`*/%%`
	`*/._`
	`**/CVS`
	`*/CVS/*`
	`**/.cvsignore`
	`**/SCCS`
	`*/SCCS/*`
	`**/vssver.scc`
	`**/.svn`
	`*/.svn/*`
	`**/.DS_Store`
	Since IzPack 5.0
		no	"true" | "false" ("true")

followSymLinks	Whether to follow symbolic links on target systems which support them. Since IzPack 5.0	no	"true" | "false" ("true")

Nested Elements

The following nested elements can be used in the `<file>` tag:

<os>

Limit the installation of this file to conditions depending on the target OS, see OS Restrictions.

<additionaldata>

This tag can also be specified in order to pass additional data related to a file tag for customizing.

Attribute	Description
key	key to identify the data
value	value which can be used by a custom action

;<additionaldata> is an element which may provide additional information as key-value pairs to certain custom actions. The particular key-value pairs will depend on the particular custom action.

Currently, there are two built-in custom actions consuming such data, `ChmodCompilerListener` and `ChmodInstallerListener`, where relevant keys are

- *`permission.dir`*,
- *`permission.file`*

with integer values interpreted as permissions like in the Unix chmod:

If value begins with "`0`" -> octal number,

otherwise is is a decimal number representing some permission.

These permissions are applied to the appropriate files either during the compilation of the package or while installing them later, depending on whether the consumer implements a CompilerListener or InstallerListener.