

Replacing Blocked Files After Reboot On Windows

 Since IzPack 5.0

All Windows systems block executable files that are currently in use, as device drivers, EXE, DLL and even JAR files. Normally, a running application is to be shut down before such files can be overwritten, for instance by an IzPack installer. This might fit most of the use cases IzPack is used with.

There can be situations where the application cannot be shut down at the installation time, but has to be updated anyway. This is where this new feature comes into the game.

Microsoft offers a Windows API for handling such cases: [Setup API](#). From the view of the pure interface it has been quiet stable over the different Windows distributions. Although Microsoft recommends using it only for installing device drivers Setup API can be used for any kind of file to be installed. Several independent installers which are not based on Windows Installer make usage of it. For example, the native [NSIS](#) installer uses a `/rebootok` flag for copying files which might be potentially in use.

Here is how blockable file handling is integrated into IzPack:

Lets start with an example of install.xml:

```
<installation version="1.0">

  <info>
    ...
    <rebootaction>ask</rebootaction>
    ...
  </info>

  ...

  <native type="izpack" name="SetupAPI.dll">
    <os family="windows"/>
  </native>

  ...

  <packs>
    <pack name="Core files" required="yes">
      <description>The core files needed for
this test</description>
      <singlefile
        src="plain/my_exefile_v1.exe"
        target="{INSTALL_PATH}/my_exefile.exe"
        override="true" blockable="auto"/>
    </pack>
  </packs>

</installation>
```

Explanation:

The file `${INSTALL_PATH}/my_exe\file.exe` on the target system is assumed to be potentially running and therefore blocked at installation time, for instance as a system service. It might be not convenient to shut it down before installation. Instead, we intend to replace it after the next system reboot.

For this purpose we mark the according file as `blockable`. The value `auto` means that the OS recognizes automatically whether the file is blocked or not, instead of forcing marking it blocked (using `force`). If it is blocked it will be internally enqueued by Windows to be replaced after a system reboot. Otherwise the file is overwritten directly due to the `override=true`. If we had `override=false` and the file would be blocked during installation time copying of the file would be skipped as expected.

With `<rebootaction>ask</rebootaction>` we force IzPack to ask the user in case of using interactive installers whether the system should be rebooted immediately. The user has the chance to confirm or deny the reboot action. Anyway, if there are pending file operations the system has to be rebooted sooner or later to apply the changes that came with the installation.

The `blockable` attribute applies also on the packs elements `file` and `fileset`. For using it there must be included the `SetupAPI.dll` as can be seen above. This library is built-in into IzPack natively.

Note that if there no explicit constraint for the OS family = `windows` defined on the according parent element `file`, `singlefile` or `fileset` copying a file with a valid `blockable` value different to `none` there is a compiler warning given to the user:

"blockable" will implicitly apply only on Windows target systems

Anyway, it is possible to mark files as `blockable` even for multi-platform installations, for instance in case of `blockable jar` files. The `blockable` attribute will be simply ignored on non-Windows platforms, resulting in a "classic" behaviour for copying files.

For a complete specification of the according elements and attributes see the documentation of `<rebootaction>` in the [Header - <info>](#) and the `blockable` attribute in one of the elements [<file>](#), [<fileset>](#) or [<singlefile>](#)

.