

Header - <info>

General information about the installation - <info>

The <info> element is used to specify some general information for the installer. It contains the following elements :

Element	Usage	Required
<appname>	The name of the application that will be installed.	Yes
<appversion>	The version of the application	Yes
<appsubpath>	The subpath for the default of the installation path. This will be appended to the installation folder selected by the user and will be created during the installation if it does not exist. A variable substitution will be done at compile time and a maskable slash-backslash conversion will be done at run-time. If this tag is not defined, the application name will be used instead.	No
<url>	The URL of the application's official website. Note: You can add the project URL from the Maven POM file by setting the configuration option <autoIncludeUrl>true</autoIncludeUrl> of the izpack-maven-plugin, see the IzPack Maven Plugin Reference .	No

<authors>

Specifies the author(s) of the application. It must contain at least one <author> element; see below.

Yes

Nested elements: <author> with the following attributes:

Attribute Name of <author>	Description
name	the author's name
email	the author's email address

Example:

```
<authors>
  <author
    name="Bud
    Spencer"
    email="buddy@bugg
    y.com" />
</authors>
```

Note: You can add authors listed in the Maven POM file by setting the configuration option <autoIncludeDevelopers>true</autoIncludeDevelopers> of the [izpack-maven-plugin](#), see the [IzPack Maven Plugin Reference](#).

<p><code><uninstaller></code></p>	<p>specifies whether to create an uninstaller after installation, and which name to use for it. This tag has the attributes:</p> <ul style="list-style-type: none"> • write (optional, values "yes" or "no", default value "yes". If yes, the uninstaller will be written. • name (optional, no default) It can be used to change the default name of the generated uninstaller, i.e. uninstaller.jar. • condition (optional, no default) This can be used to specify a condition which has to be fulfilled for creating the uninstaller. • path (optional, defaults to <code>#{INSTALL_PATH}</code> This can be used to define the destination path where the uninstaller is written to, i.e. <code>#{INSTALL_PATH}/Uninstaller</code>. 	<p>No</p>
<p><code><javaversion></code></p>	<p>This specifies the minimum version of Java required to install your program. Values can be 1.2, 1.2.2, 1.4, etc. The test is a lexical comparison against the <code>java.version</code> System property on the install machine.</p>	<p>Yes</p>
<p><code><requiresjdk></code></p>	<p>Valid values: yes or no. Specifies whether a JDK is required for the software to be installed and executed. If "yes" and the JDK is not already installed, then the user will be informed and given the option to proceed with the installation process or abort.</p>	<p>Yes</p>
<p><code><webdir></code></p>	<p>If this element is present, a web installer will be created. The contents of the tag specifies the URL from which packages are retrieved at install time. The content of the element must be a properly formed URL that points to the remote folder where the packages reside.</p>	<p>No</p>

<code><summarylogfilepath></code>	<p>If this element is present, it specifies the path for the logfile of the <code>`SummaryLoggerInstallerListener`</code>. If it is not specified the logfile is not created.</p>	No
<code><writeinstallationinformation></code>	<p>Valid values are <code>yes</code> or <code>no</code>. Default is <code>yes</code>. Specifies if the file <code>.installinformation</code> should be written. This file includes the information about installed packs.</p>	No
<code><pack200/></code>	<p>Adding this element will cause every JAR file that you will add to your packs to be compressed using Pack200. As a special exception, signed JARs are not compressed using Pack200, as it would invalidate the signatures. Compressing makes the compilation process a little bit longer, but it usually results in drastically smaller installer files. The decompression is relatively fast. Please note that Pack200 compression is destructive, i.e., after decompression a JAR won't be identical to its original version (yet the code in the class files remains semantically equivalent).</p>	
<code><tempdir/></code>	<p>If this element is included then a randomly named temporary directory will be created at the start of the install and deleted when the installation completes.</p> <p>This feature is useful in a couple of scenarios.</p>	No

1. If your installer needs to install a third party product or library by executing that products installer, but you don't want those artifacts to be kept afterwards. You can do this with the delete attribute on the execute element but that won't clean up any supporting files.
2. If you have some SQL scripts or any other files which need to be run through an interpreter and you don't want them hanging around after the install completes.
3. If you need to create a file, populate it with values from the installers gui and pass that file off to another program for some reason, after which you no longer need the file.
A temporary directory allows all of the temporary files created during installation to be kept together and neatly cleaned up automatically at the end of the install, and does so in an OS independent manner.
The following XML attributes are supported:
 - **variablename**: The name of the variable which will contain the absolute path of the temporary directory at runtime. The default value is **TEMP_DIRECTORY**.
 - **prefix**: A string which will be used as the start of the randomly generated directory name. The default value is **IzPack**.
 - **suffix**: A string which will be used as the end of the randomly generated directory name. The default value is **Install**.Multiple **tempdir** elements are supported with unique variablename attributes.

<p><code><run-privileged/></code></p>	<p>Adding this element will make the installer attempt to launch itself with administrator permissions. It also supports a condition attribute to refer to a condition ID so that the elevation is not always attempted (e.g., you may want to activate it only for Windows Vista). This is not supported on all platforms, in which case a message will be provided to the user before continuing the installation. You can disable this feature for the uninstaller by specifying <code>uninstaller="yes"</code> as an attribute. Only use this feature if you really need to be an administrator as part of your installation process.</p> <ul style="list-style-type: none"> • condition This lists the operating systems where the installer should run with administrator permission. The operating systems are specified as literal names separated by the " " (pipe) character. Valid operating system names are <ul style="list-style-type: none"> • <code>izpack.windowsinstall.7</code> • <code>izpack.windowsinstall.vista</code> <p>An example for a windows installer would be <code><run-privileged condition="izpack.windowsinstall.7 izpack.windowsinstall.vista"/></code> which would acquire administrator permissions for installers running on Windows 7 or Vista.</p>	<p>No</p>
<p><code><rebootaction></code></p>	<p>Defines what to do if there were pending installation operations left which require a reboot; otherwise any of the options below will be ignored. Possible values are:</p>	<p>No</p>

- **ignore** (default) Doesn't reboot at all even if there are pending operations. Pending operations can be recognized only on the installer command line output (for all options).
- **notice** Doesn't reboot, but notifies the user interactively at the end of an installation, which must be confirmed. Notification works only for interactive installation types (no auto-installation).
- **ask** Reboots only if the user confirms interactively at the end of an installation.
- **always** Reboots always without any confirmation at the end of an installation.

The usage of `<rebootaction>` is requires the use of the attribute `blockable` with the values `auto` or `force` at least in one of the elements `<file>`, `<fileset>` or `<singlefile>`, which indicates blocked files which would result in a failing installation of the file if izPack tries to write these files during the installation.

`<rebootaction>` only works and makes sense on Windows, where target files (as device drivers, EXE or DLL files) might be blocked during an installation. On platforms other than Windows the `<rebootaction>` element will be ignored. `<rebootaction>` supports the `condition` attribute to limit reboot processing on particular conditions. Without setting at least one attribute `blockable="auto"` or `blockable="force"`, `<rebootaction>` will not have any effect.

See also the description of the `blockable` attribute in the documentation of the `<pack>` subtags `<file>`, `<fileset>` and `<singlefile>`.

Here is an example of a typical `<info>` section :

```
<info>
  <appname>Super extractor</appname>
  <appversion>2.1 beta 6</appversion>

  <appsubpath>myCompany/SExtractor</appsubpath
  >
  <url>http://www.superextractor.com/</url>
  <authors>
    <author name="John John Doo"
email="jjd@jjd-mail.com"/>
    <author name="El Goyo"
email="goyoman@mymail.org"/>
  </authors>
  <javaversion>1.2</javaversion>
</info>
```

Here is one where the privileges elevation is attempted on Windows Vista and Mac OS X :

```
<info>
  <appname>IzPack</appname>
  <appversion>4.2.0</appversion>
  <authors>
    <author email="" name="Julien Ponge
(project founder)"/>
    <author email="" name="The fantastic
IzPack developers and contributors"/>
  </authors>
  <url>http://izpack.org/</url>
  <javaversion>1.5</javaversion>
  <requiresjdk>no</requiresjdk>
  <run-privileged
condition="izpack.windowsinstall.vista|izpac
k.macinstall"/>

  <summarylogfilepath>$INSTALL_PATH/installinf
o/Summary.htm</summarylogfilepath>
</info>
```