

inifile

inifile

Supports patching/merging and explicit editing of 'standard' INI-format configuration files most frequently used by older Windows applications. Includes support for comments (denoted by ; or #), sections, and options-style auto-numbered properties.

The processing engine for INI files is an integrated version of the [ini4j framework](#). The options for attributes and nested elements reflect the functionality of this framework. The `<inifile>` configuration type can be used to patch or update a single configuration file, or to patch a fileset of INI files against a target directory containing INI files. Manual definition of section/key/value entries is supported by using the `<entry>` element

Parameters

Attributes marked *ini4j* expose features of the underlying ini4j framework. See the ini4j documentation for fuller details.

Attribute	Description	Required
fromfile	The source file containing the original configuration.	No. Patching destinations can also be updated by manual definition using nested <code><entry></code> elements. If specified, <code>fromfile</code> must exist.
tofile	The file to patch to.	Exactly one of these is required.
todir	The directory to patch to.	Similar to the Ant copy task, <code>fromfile</code> can be used with <code>tofile</code> or <code>todir</code> . Nested <code><fileset></code> elements can be used with <code>todir</code> , but not with <code>tofile</code> , since the use of <code><fileset></code> implies a bulk patching operation. Neither <code>tofile</code> nor <code>todir</code> have to exist, but an error will occur if they cannot be created, or if <code>create</code> is set to false.
targetfile	An alternative file to write the patched configuration to.	No. When specified, <code>tofile</code> itself is not modified. Cannot be used when nested <code><fileset></code> s are specified (since it makes no sense).
preservelastmodified	Give the new configuration file(s) the same last-modified time as the original source file(s).	No; defaults to false.
overwrite	Allow existing destination files to be overwritten.	No; defaults to true. If <code>overwrite</code> and <code>create</code> are both set to false, an error occurs.
create	Allow destination files to be created where they don't already exist.	No; defaults to true. If <code>overwrite</code> and <code>create</code> are both set to false, an error occurs.
cleanup	Deletes original configuration file(s) after new patched file(s) is/are successfully written. If any destination or target file could not be processed or written for any reason, the original file is never deleted, even if this is set to true.	No; defaults to false.
failonerror	If false, the listener will try to continue processing remaining actions even when errors in reading, processing, or writing files occur. Instead of immediately failing, a warning message is written to the log.	No; defaults to true.
enablemultiplemappings	If true the task will process to all the mappings for a given source path. If false the task will only process the first file or directory. This attribute is only relevant if there is a mapper subelement.	No; defaults to false.
condition	Specifies the name of an IzPack installer condition that must be fulfilled before the configuration is executed.	No.

keepOldKeys (<i>ini4j</i>)	<p>Whether to generally preserve entries with the same name (but not necessarily the same value) from the source configuration file, if they can be found.</p> <p>If <code>keepOldKeys="true"</code> and the entry occurs in the old configuration and also in the new configuration, the new configuration entry is left and the value is set according to <code>keepOldValues</code>. Otherwise, if the configuration entry found in an old configuration is not found in a new configuration, the old configuration entry and value is merged into the new configuration. Manual <code><entry></code> definitions will override the old value of a preserved key entry.</p>	No; defaults to true.
keepOldValues (<i>ini4j</i>)	<p>Whether to preserve the values of equal entries from an old configuration, if they can be found.</p> <p>Set false to overwrite old configuration values by default with the new ones, regardless whether they have been already set in an old configuration. Values from an old configuration can only be preserved, if the appropriate entries exist in an old configuration. Manual <code><entry></code> definitions will override the old value of a key entry even when <code>keepOldValues</code> is true.</p>	No; defaults to true.
escape (<i>ini4j</i>)	<p>Whether to parse the following sequences as escape characters (and write them accordingly to the target file) when they appear in a parsed configuration file:</p> <ul style="list-style-type: none"> • <code>\\</code> (escape character) • <code>\t</code> (tab) • <code>\n</code> (new line) • <code>\f</code> (form feed for printers) • <code>\b</code> (backspace) • <code>\r</code> (line feed) 	No; defaults to true.
escapeNewLine (<i>ini4j</i>)	<p>When true, lines are concatenated as far as the last character before the line break backslash (<code>\</code>). In this case the trailing line break is omitted, interpreting several lines ending on <code>\</code> as a single "long" line.</p>	No; defaults to true.
headerComment (<i>ini4j</i>)	<p>Allows header comments in the file. If this is true, the comment before the first entry is treated as the global header comment for the whole file, not logically assigning this comment to the first option itself. The header comment might be internally overwritten separately, and is separated using a newline</p>	No; defaults to false.
emptyLines (<i>ini4j</i>)	<p>Preserves empty lines. If false, empty lines in configuration files are discarded when saving the file.</p>	No; defaults to true.
operator (<i>ini4j</i>)	<p>Set this option to override the default operator key-value assignment operator.</p> <p>When a file is parsed, the first occurrence of the <code>=</code> or <code>:</code> character is respected as being the operator between a key and its value; but the target file is always saved using <code>=</code> with a leading and trailing space character, i.e. <code>" = "</code>. This works fine for 'standard' Windows INI files. This attribute overrides the operator to use when writing non-standard target files, e.g. for preserving <code>:</code> as the operator, or to explicitly set the string <code>"="</code> (i.e. no leading/trailing whitespaces).</p>	No; defaults to <code>" = "</code> . A string of one or more character is acceptable.
resolveExpressions (<i>ini4j</i>)	<p>Whether ini4j expressions should be resolved when a configuration is being patched. This is not the same as substituting IzPack variables, but resolves references to configuration entries in the same configuration file. See [ini4j] - Expression handling.</p>	No; defaults to false.

Nested Elements

The following nested elements can be specified to create manual definitions, or to define bulk patching operations.

entry

The nested `<entry>` is used to override the definition of a certain configuration key and its value against the default behavior defined with action-global attributes or their ini4j defaults. An `entry` can even be used without the `fromfile` attribute simply to modify the target configuration file with explicit values. Further, an entire configuration file can be defined entirely using `<entry>` elements where `tofile` did not previously exist.

Attribute	Description	Required
section	The INI section to lookup.	Yes.
key	The INI key to deal with. If this ends with a dot <code>.</code> , the key is automatically assumed to be an auto-numbered value, e.g. <code>key = "key."</code> would match a collection of keys <code>"key.0"</code> , <code>"key.1"</code> , etc. The operation below would apply on each of those keys.	Yes, unless <code>operation</code> is <code>"keep"</code> or <code>"remove"</code> (these operations may be applied to a whole section).
value	The new value to set the INI key to.	Yes, if <code>operation="set"</code> and <code>default</code> is not specified; optional otherwise. Defaults to <code>"1"</code> for numeric data types, and to <code>" "</code> for the <code>string</code> type (i.e. the empty string), only for increment/decrement operations.
default	A fallback value to set the INI key to if no other definition can be found, i.e. if parameter <code>value</code> is not specified, or if the key is not already defined in the INI file.	Yes, if <code>operation="set"</code> and <code>value</code> is not specified; optional otherwise. When <code>dataType="date"</code> , the keyword <code>"now"</code> is allowed. Where no previous value is defined, defaults to <code>"now"</code> for the <code>date</code> type, and to <code>"0"</code> for the <code>int</code> type, only for increment/decrement operations.
dataType	Instructs that entry values be treated as a certain data type during processing. Support currently exists for treating a property value as a date or an integer, instead of as a plain string. Can be used with the <code>pattern</code> parameter to define string parsing and output formatting, and/or with the <code>operation</code> parameter to perform basic calculations.	No; defaults to <code>"string"</code> . Can also be set to <code>"int"</code> or <code>"date"</code> .
operation	Overrides global patching behavior for specific entries, or allows basic calculations for numeric data types (i.e. <code>int</code> and <code>date</code>). When this attribute is omitted, the specified entry will be set to the specified <code>value</code> or <code>default</code> . Entries can be removed from the INI by specifying <code>operation="remove"</code> (<code>value/default</code> do not have to be set in this case). The entry is removed even if <code>keepOldValues</code> is <code>true</code> . Using <code>operation="keep"</code> preserves the original value of a key when using a <code>fromfile</code> , even if <code>keepOldKeys</code> is <code>false</code> . These operations are valid for all data types. Specifying <code>operation="+"</code> performs an increment operation can on numeric data types, or an append operation on the <code>string</code> type. For numeric types, a decrement operation can also be performed, where the <code>value</code> is deducted from the original value of a key (or the specified <code>default</code> , where no original value exists). Use parameter <code>unit</code> to define the scope of increment/decrement on the <code>date</code> type.	No; defaults to <code>"="</code> . Valid values for all data types are <code>"="</code> (set), <code>"+"</code> (increment/append), <code>"remove"</code> , <code>"keep"</code> . For numeric data types, the value <code>"-"</code> (decrement) is also valid.
unit	Defines the date component to modify when <code>dataType="date"</code> and <code>operation</code> is <code>"+"</code> or <code>"-"</code> .	No; defaults to <code>"day"</code> . Can only be specified when <code>dataType="date"</code> . Valid values are <code>"millisecond"</code> , <code>"second"</code> , <code>"minute"</code> , <code>"hour"</code> , <code>"day"</code> , <code>"week"</code> , <code>"month"</code> , <code>"year"</code> .
pattern	For numeric data types, a string formatting pattern can be specified using conventional Java formatting placeholders. This defines both the input parsing and output formatting pattern for this entry. See javadoc for DecimalFormat (applies to <code>int</code>) and SimpleDateFormat (applies to <code>date</code>).	No. The default pattern for <code>date</code> is <code>"yyyy/MM/dd HH:mm"</code> .

fileset

Specifies patch source files using one or Ant-style `<fileset>` elements. See the description of the `<fileset>` element. Cannot be used with the attribute `targetfile`, or with any manual definition `<entry>` elements.

mapper

Defines one or more file name transformations to apply to the configurable execution. See the description of the `<mapper>` element. Only one mapper element can be used in a configurable (use `compositemapper` to define a chain of transformations).

Examples

```
<infile fromfile="${INSTALL_PATH}/config.ini"
tofile="${INSTALL_PATH}/upgrade/config.ini" targetfile="${INSTALL_PATH}/config.ini"
  overwrite="true" keepOldKeys="false">
  <entry section="default" key="allowLegacyOps" operation="keep" />
</infile>
```

Merge the values from an existing file into a new version of the file, and overwrite the original file with the resulting INI configuration. Preserve values from the original file for matching keys in the new file, but discard keys not found in the new file. Make an explicit exception for the key `allowLegacyOps` - if specified in the original file, copy it to the new configuration even if it doesn't exist in the new file. Note the use of IzPack variable `INSTALL_PATH` in the file path specs.

```
<infile tofile="${INSTALL_PATH}/config.ini"
  overwrite="true" operator=":" condition="isUpgradeInstallation">
  <entry section="default" key="appVersion" value="${APP_VER}" />
  <entry section="appinfo" key="configVersion" operation="+" />
</infile>
```

Update the `appVersion` property in the `default` section of an existing INI file, and increment the `configVersion` property in the `appinfo` section. Use the `:` character (instead of the default `=`) to separate keys and values in the output file. Only make the modification if the IzPack condition `isUpgradeInstallation` is fulfilled. Note the use of IzPack variables `INSTALL_PATH` and `APP_VER`.