xmlfile

xmlfile

Supports configurable merging of well-formed XML files. As well as allowing one-way merges (specified individually, or in bulk from a source fileset to a destination directory), `<xmlfile>` also allows merging of multiple source files to a single destination file.

The processing engine for XML file merging is an integrated version of the xmlmerge module from the EL4J framework. The configuration options for merging reflect the functionality of this framework. It is not possible to explicitly add completely new markup to an XML file using this task, although fine-grained configuration of the merging operation is provided by xmlmerge.

Parameters

Attribute	Description	Required
fromfile	The source XML file.	No. Nested `<fileset>` elements can also supply the merge source(s). If specified, `fromfile` must exist.
tofile	The XML file to merge to.	Exactly one of these is required.
todir	The directory to patch to.	Similar to the Ant copy task, `fromfile` can be used with `tofile` or `todir`. Nested `<fileset>` elements can be used with `todir` (to perform bulk merging on sets of matching files), **and** with `tofile`, (to merge many source files into a single destination). Neither `tofile` nor `todir` have to exist, but an error will occur if they cannot be created, or if `create` is set to false.
targetfile	An alternative file to write the merged XML to.	No. When specified, `tofile` itself is not modified. Cannot be used when nested `<fileset>`s or `todir` are specified (since it makes no sense).
preservelastmodified	Give the new XML file(s) the same last-modified time as the original source file(s).	No; defaults to false.
overwrite	Allow existing target files to be overwritten.	No; defaults to true. If `overwrite` and `create` are both set to false, an error occurs.
create	Allow target files to be created where they don't already exist.	No; defaults to true. If `overwrite` and `create` are both set to false, an error occurs.
cleanup	Deletes original XML file(s) after new merged 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.

configfile	Specifies the path to a plain Java property file (i.e. line-separator-delimited `key=value` properties) that specifies any number of merge configuration directives (see <xpathproperty>, below).	No. Cannot be used if {{><

Nested Elements

The following nested elements can be specified to configure, and define the scope of, XML merging operations.

xpathproperty

The nested <xpathproperty> element is used to override the default XML merging handling of a certain XML content. Each key value ends either on *.default* (just one is allowed, and is not applicable to xpath.) or .<*number*>, where all keys with the same ending <*number*> are assigned to one and the same element matching *xpath.path.*<*number*>.

Attribute	Description	Required
key	The XPath property key to deal with.	Yes. Valid values are: • "matcher.default" • "action.default" • "mapper.default" • "xpath.path.<*number*>" • "matcher.path.<*number*>" • "action.path.<*number*>" • "mapper.path.<*number*>"

value	The XPath property value to set the above key value to.
	
	Using `"matcher."` keys
	Defines how to match a pair of elements from the original and the patch document. Possible values:
	
	• "TAG"
	Match the element names.
	• "ATTRIBUTE"
	Match element name and attributes (names and values)
	• "ID_ATTRIBUTE"
	Match element name and the value of the attribute with the name "ID".
	• "NAME_ATTRIBUTE"
	Match element name and the value of the attribute with the name "NAME".
	• "SKIP"
	Never match neither element names nor attributes.
	
	Using `"action."` keys
	Defines the merge engine behavior.
	
	• "FULLMERGE"
	Merge implementation traversing element contents independent of their order.
	Note: This does completely merge all matching elements and attributes regardless of their order in the document. This is more time-consuming than ORDEREDMERGE, but it is not necessary to rely on a well-ordered XML.
	• "ORDEREDMERGE"
	Merge implementation traversing source and destination element contents in parallel. Works when contents are in the same order in both elements.
	Note: This does completely merge all matching elements and attributes, assuming they appear in the same order comparing the source file with the destination file. This action executes faster than FULLMERGE, but can lead to bad results if the order of both sides is different.
	• "REPLACE"
	Copies the source element if it exists.
	• "OVERRIDE"
	Ignores the source element if it exist in the destination, keep the source element if no corresponding destination element exists.
	• "KEEP"
	Retains the destination element only if it exists the source; otherwise nothing is added.
	• "COMPLETE"
	Retains the destination element only if it does not exist in the source document; if a matching element is found in the source, retain the original.
	• "DELETE"
	Retains the source element only if it does not appear in the destination document.
	• "PRESERVE"
	Copies the source element regardless of the existence of a matching destination element.
	• "INSERT"
	Copies the destination element into the output by inserting it after source elements of the same name. Usually applied with the matcher option "SKIP".
	• "DTD"
	Copy the destination element into the output with the correct position according to the DTD defined in the document.
	
	Using `"mapper."` keys
	How the matching element should be further transformed during applying the action. More transformations may be supported in future, filtering out elements and attributes with a specified namespace.
	
	• "IDENTITY"
	Does not transform the element.
	
	Using `"xpath.path."` keys
	The path to the particular XML elements which should be handled by the so numbered action, matcher and mapper (or their defaults). The XPath is a quite complex language, for more information see XML Path Language (XPath) Version 1.0, w3schools.com XPath Tutorial, and many more freely-available documents and tutorials.
		Yes. Defaults are listed here.
		
		`matcher.` keys: `"ATTRIBUTE"`
		`action.` keys: `"FULLMERGE"`
		`mapper.` keys: `"IDENTITY"`

fileset

Specifies patch source files using one or Ant-style `<fileset>` elements. See the description of the `<fileset>` element. Can be used with the parameters `tofile`/`targetfile` to specify that many files are to be merged into one file.

mapper

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

Examples

```
<xmlfile fromfile="${INSTALL_PATH}/conf/resources.xml.configbak"
         tofile="${INSTALL_PATH}/conf/resources.xml"
         cleanup="true" condition="isUpgrade">
  <xpathproperty key="action.default" value="COMPLETE"/>
</xmlfile>
```

Merges two XML files, retaining all values and elements from the source file `resources.xml.configbak` and discarding matching elements in the destination `resources.xml`. The source file is deleted after the merge, and the destination file replaced with the new, merged version. The task is only executed when the IzPack condition `isUpgrade` is fulfilled.

```
<xmlfile fromfile="${INSTALL_PATH}/conf/functions.xml.configbak"
         tofile="${INSTALL_PATH}/conf/functions.xml"
   configfile="resources/config/functions_xml.properties"
         cleanup="true" condition="isUpgrade" />
```

functions_xml.properties

```
action.default=FULLMERGE

xpath.path1=/ROOT/PARAMETERS/
matcher.path1=NAME_ATTRIBUTE
action.path1=REPLACE

xpath.path2=/ROOT/FUNCTIONS/FUNCTION[@ID != &quot;FUNC1&quot; and @ID !=
&quot;FUNC2&quot;]
matcher.path2=ID_ATTRIBUTE
action.path2=REPLACE

xpath.path3=/ROOT/FUNCTIONS/FUNCTION[@ID = &quot;ANOTHER_FUNCTION&quot;]
matcher.path3=ID_ATTRIBUTE
action.path3=KEEP
```

This example uses an external properties file to define merging behavior for two XML files. The default behavior attempts to merge all matching elements and attributes in the two XML files. Three exceptions are defined for elements matching specific XPath condition. XPath `path1` is matched by element name and the value of attribute `name`, and copies in the element from the source file, completely replacing the version in the destination file; XPath `path2` is matched by element name and the value of attribute `ID`, and also is subject to the `REPLACE` action; XPath `path3` is also matched on element name and attribute `ID`, and retains only destination file elements with matching source file elements.