ConfigurationInstallerListener

Configuration Install Actions - `ConfigurationInstallerListener`

Introduction

The ConfigurationInstallerListener is a powerful feature introduced in IzPack 5.0. Using this action there can be done several configuration tasks usually needed for installations, including

- merging of configurations
- patching of configurations
- explicitly set, delete, add or keep certain configuration entries

which can be applied on and between

- option and property files
- INI files
- XML files
- Windows registry entries.

ConfigurationListener is useful for software updates and merges option (properties), INI and XML files from a previous installation to those files with the same path coming with a new version of your software. It does a three-way merge - patching an original file against a new file resulting in a target file. For using in-place patches you must ensure the previously installed files are automatically renamed when installing the new files coming with an update installer, before ConfigurationListener starts. This can be achieved by using the overrideRenameTo attribute for installed files or file sets, see Packs.

Configuration actions are defined in an IzPack resource ConfigurationActionsSpec.xml, which has to be necessarily defined when the ConfigurationInstallerListener is used.

Configuration actions can be only used during installation, not for reverting configurations when uninstalling.

Using Configuration Actions

Using the ConfigurationInstallerListener requires the following definitions in the install descriptor:

- There might be a couple of additional libraries bound to the installer depending on the needs:
  - In order to use XML merge/patch actions in common you need to add `jdom:jdom2` artifact to the installer.
    JDom 2.0.5 is the current minimum requirement IzPack has been tested against.
  - In order to use XPath queries you need also to add the `jaxen:jaxen` artifact to the installer.
    JDom 2.0.5 currently optionally depends on Jaxen 1.1.4.
  - In order to use the *"dtd"* XML merge action you need to add `com.wutka:dtdparser` artifact to the installer.
    DTD Parser 1.21 is the current minimum requirement IzPack has been tested against.

Example of adding all optional libraries

```
<jar src="dtdparser-1.21.jar" stage="install"/>
<jar src="jaxen-1.1.4.jar" stage="install"/>
<jar src="jdom-2.0.5.jar" stage="install"/>
```

- Define the configuration action descriptor as resource "ConfigurationActionsSpec.xml"

```
<resources>
    <res id="ConfigurationActionsSpec.xml" src="resources/config-actions-spec.xml"/>
</resources>
```

- Include the ConfigurationsInstallerListener as custom action

```
<listeners>
    <listener classname="ConfigurationInstallerListener" stage="install"/>
</listeners>
```

Configuration Actions Example

The following example demonstrates the possibilities of configuration actions in an installations. It makes usage of the new feature automatic renaming instead of overwriting installed files, appending a certain suffix (.configbak) to the original file name.

```
<izpack:configurationactions version="5.0"
                             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:izpack="http://izpack.org/schema/configurationactions"

xsi:schemaLocation="http://izpack.org/schema/configurationactions
http://izpack.org/schema/5.0/izpack-configurationactions-5.0.xsd">

  <pack name="My Application Core">
    <configurationaction order="afterpack">

      <!-- Patch and merge single all property files -->
      <configurableset type="options" cleanup="true"
                       keepOldKeys="false" keepOldValues="true" overwrite="true"
                       todir="${INSTALL_PATH}/confs"
                       condition="isUpgrade">
        <fileset dir="${INSTALL_PATH}/conf">
          <include name="*.properties.configbak"/>
        </fileset>
        <mapper type="glob" from="*.properties.configbak" to="*.properties"/>
      </configurableset>

      <configurable type="options" cleanup="true"
                    keepOldKeys="false" keepOldValues="false"
                    patchfile="${INSTALL_PATH}/wrapper.conf.configbak"
                    tofile="${INSTALL_PATH}/wrapper.conf"
                    condition="keepWrapperJavaVersion">
        <entry key="wrapper.java.command" operation="keep"/>
        <entry key="wrapper.java.classpath." value="plugins/.*" lookupType="regexp"
operation="keep"/>
        <entry key="wrapper.app.parameter."
               value="org\.codehaus\.myapp\.SomePrefix.*" lookupType="regexp"
operation="keep"/>
      </configurable>

      <!-- Patch and merge single XML files -->
      <configurable type="xml" cleanup="true"
```

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

      <configurable type="xml" cleanup="true"
                    patchfile="${INSTALL_PATH}/conf/functions.xml.configbak"
                    tofile="${INSTALL_PATH}/conf/functions.xml"
                    condition="isUpgrade">
        <xpathproperty key="action.default" value="FULLMERGE"/>
        <xpathproperty key="xpath.path1" value="/ROOT/PARAMETERS/PARAM"/>
        <xpathproperty key="matcher.path1" value="NAME_ATTRIBUTE"/>
        <xpathproperty key="action.path1" value="REPLACE"/>
        <xpathproperty key="xpath.path2"
                       value="/ROOT/FUNCTIONS/FUNCTION[@ID != &quot;FUNC1&quot; and
@ID != &quot;FUNC2&quot;]"/>
        <xpathproperty key="matcher.path2" value="ID_ATTRIBUTE"/>
        <xpathproperty key="action.path2" value="REPLACE"/>
        <xpathproperty key="xpath.path3" value="/ROOT/FUNCTIONS/FUNCTION[@ID =
&quot;ANOTHER_FUNCTION&quot;]"/>
        <xpathproperty key="matcher.path3" value="ID_ATTRIBUTE"/>
        <xpathproperty key="action.path3" value="KEEP"/>
      </configurable>

      <configurable type="xml" cleanup="true"
                    patchfile="${INSTALL_PATH}/conf/data1.xml.configbak"
                    tofile="${INSTALL_PATH}/conf/data1.xml"
                    condition="isUpgrade">
        <xpathproperty key="action.default" value="COMPLETE"/>
      </configurable>
    </configurationaction>
```

```
  </pack>
</izpack:configurationactions>
```

Configuration Actions Reference

Basic format

The root element of an configuration actions descriptor is <configurationactions>. On the next level there are to be defined the packs using the <pack> element, defining a set of configuration actions for a certain pack of the IzPack installation:

```
<configurationactions>
  <pack name="...">
    <configurationaction order="...">

    ... variables and configuration actions go here ...

    </configurationaction>
  </pack>
</configurationactions>
```

Attributes of the <pack> tag:

- ***name***
  The name of the pack, for which the nested configuration actions should be executed.
  The nested configuration actions will not apply in case the pack defined by *name* is omitted by the user or automatically during the installation.

Attributes of the <configurationaction> tag:

- ***order*** := "beforepack" | "beforepacks" | "afterpack" | "afterpacks"
  The installation phase for executing the nested set of configuration actions according to the enclosing installation package. See Custom Actions description for more details.

Configuration Action Variables

In the configuration action descriptor, there can be defined variables similar to dynamic variables for each configuration action, but just in scope of the configuration action itself. Those local variables are embedded in the variables tag like this:

Basic configuration action variable syntax

```
<configuration_action_tag>
  <variables>
    <variable/>
    <variable/>
    ...
  </variables>
</configuration_action_tag>
```

The syntax for the single variable tags is the same like for dynamic variables in the main installation descriptor, see Dynamic Variables.

Configuration Actions

<configurable> Configuration Action - Patch a Single File or Registry Entry

This action is used for handling a single configuration file of a certain type.

A configuration action is added using the element **<configurable>**.

Attributes

Attribute	Description	Required	Allowed Values (Default)
type	The file type to handle.	yes	"options" | "ini" | "xml" | "registry" ("options")
toFile	The target file to write the patch result to. This is equal to the file a patch should be apllied against, if the attribute *originalFile* is not set.	yes (if type != "registry")	absolute or relative (against $INSTALL_PATH) path denoting a file
toKey	The registry root key to patch to, provided that type = "registry".	yes (if type = "registry")	registry path
patchFile	The file to patch from. This is the source file with the full original contents, which might be preserved, overridden etc. according to the patch rules defined in other attributes. Nested *fileset* elements can be used instead to patch from a couple of files to the target.	no (if the nested *file set* tag or simply the nested *ent ry* tag is used)	absolute or relative (against $INSTALL_PATH) path denoting a file
originalFile	The file to patch against, if it should be different than the final target file. This might be useful on updates, if the previous file is at a different place than the resulting one; in this case the patch can be applied against the old one and the result written to the file defined by the *toFile* attribute (thus to the new location). This attribute has only effect if the *patchFile* attribute is set.	no	absolute or relative (against $INSTALL_PATH) path denoting a file
fromKey	The registry root key to patch to, provided that type = "registry".	yes (if type = "registry")	registry path
cleanup	Whether to remove the patch file after the operation which don't act also as target file with the same name. This attribute has only effect if the *patchFile* attribute is set.	no	"true" | "false" ("false")
create	Whether to create the target file when it doesn't already exist	no	"true" | "false" ("true")
keepOldKeys	Whether to generally preserve equal entries but not necessarily their values from the patch configuration file, if they can be found. This means if keepOldKeys="true" 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 the *keepOldValues* setting, 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. This attribute has only effect if the *patchFile* attribute is set.	no	"true" | "false" ("true")
keepOldValues	Whether to preserve the values of equal entries from an old configuration, if they can be found. 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. This attribute has only effect if the *patchFile* attribute is set.	no	"true" | "false" ("true")

escape	Whether to parse the following sequences as escape characters and write them as those while saving as they appear in the parsed configuration file: • \\ (escape character), • \t (tab), • \n (new line), • \f (form feed for printers), • \b (backspace), • \r (line feed).	no	"true" | "false" ("true")
escapeNewLine	When this option is true, lines are concatenated as far as the last char before the line break is a backslash ('\'). In this case the trailing line break is omitted, interpreting several lines ending on '\' as a single "long" line.	no	"true" | "false" ("true")
headerComment	Whether to allow header comments in the file. If this option is set 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 overwritten internally separately and is separated using a newline	no	"true" | "false" ("false")
emptyLines	Whether to preserve empty lines. If set false, empty lines in configuration files get lost when saving the file.	no	"true" | "false" ("true")
operator	Set this option to override the default operator for option and INI files completely. Currently, when a configuration files is parsed, there can be used '=' or ':' as operator between keys and values, but the file is always saved using '=' with a leading and trailing space character, which works fine for Java property and Windows INI files. The operator might be overload for saving non-standard formats, for instance for leaving the ':' as operator. Another use case is saving Java Service Wrapper configuration files, which isn't correctly parsed if there are leading and trailing whitespace characters on the operator - for that case, the operator can be set to the string "=" preventing usage of leading and trailing whitespaces for saving.	no	string of 1 or more characters (" = ")
resolveExpression	Whether expressions should be resolved during patching of configurations. This is not similar to substituting IzPack variables, but by means of resolving references to configuration entries in the same configuration file, see [ini4j] - Expression handling. This attribute has only effect if the *patchFile* attribute is set.	no	"true" | "false" ("false")
overwrite	Whether to allow overwriting an existing target file.	no	"true" | "false" ("false")
condition	An Izpack condition, which must be fulfilled to start the according configuration action.	no	string referring to a condition id

Nested Elements

<entry>

The nested <entry> element is used to override the handling of a certain configuration key and its value against the default behavior defined using the action-global attributes above or their defaults. This means, <entry> overrides modifications of the target file after patching, and can be also used without the patchFile attribute simply to modify the target configuration file with explicit values.

Limitations: The nested entry element works currently only for type = "option", "ini" or "registry".

Attribute	Description	Required	Allowed Values (Default)
section	The entry INI section to lookup a key within.	no (required if configurable attribute type = "ini"	
key	The entry key to deal with. In case the key ends with a dot ('.'), the key is automatically assumed as a autonumbered value. Example key = "key." would match all keys "key.0", "key.1", ...). The operation below applies on each of those keys unless a limitation by lookup by value isn't done (attribute *lookupType*).	yes	

value	configurable/type="option": The entry value to set or to lookup for. configurable/type="ini": The entry value to set. configurable/type="registry" - The registry value (not data, but for looking up the right data, in terms of Microsoft)	no (required if attribute lookupType is set)	
data	configurable/type="registry": The registry data. This attribute is allowed just for registry entries.	no (required if configurable/type is "registry" and a value should be set)	
lookupType	The lookup type used if the entry should be looked up by value. Setting this attributes automatically activates lookup by value in the original file (the file to patch to). The "value" attribute must be set with the plain or regular expression to lookup by value. "plain" means looking up the plain value according to *value*, "regexp" means treating *value* as a Java regular expression. This applies currently just for values of configurable/type="options".	no	"plain" | "regexp" ("plain")
operation	The operation to apply on the entry value: "+" or "=" (default) for all datatypes; "-" for date and int only), "keep" to keep the single value from the patch file for the according key, or "delete" for deleting the matching key-value pair completely.	no	"+" | "=" | "-" | "keep" | "delete" ("=")
dataType	The data type the entry value should be treated as for certain operations.	no	"int" | "date" | "string" ("string")
unit	The unit of the value to be applied to "date" +/- operations.	no	"millisecond" | "second" | "minute" | "hour" | "day" | "week" | "month" | "year" ("day")
default	Initial value to set for a property if it is not already defined in the property file. For type "date", an additional keyword is allowed: "now"	no	
pattern	For "int" and "date" type only. If present, the value will be parsed and formatted accordingly.	no	

<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, not for `xpath.`) or `.<number>`, where all keys with the same ending `<number>` a re assigned to one and the same element matching `xpath.path.<number>`.

Limitations: The nested entry element works only for type ="xml".

Attribute	Description	Required	Allowed Value (Default)
key	The XPath property key to deal with.	yes	"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.	yes	• Key begins on "matcher.":
			Defines how to gather a matching 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.
			Default: "ATTRIBUTE"
			
			• Key begins on "action.":
			  • "FULLMERGE"
			  Merge implementation traversing element contents undependend of their order.
			  Note: This does completely merge all matching elements and attributes regardless of there order. This is more time-consuming than ORDEREDMERGE, but it is not necessary to rely on a well-ordered XML.
			  • "ORDEREDMERGE"
			  Merge implementation traversing parallelly both element contents. 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 patched file with the patch file. This action executes faster than FULLMERGE, but can lead to bad results if the order of both sides is different.
			  • "REPLACE"
			  Add or replace - if there is a matching patch element, add it or replace an existing matching original element by it, otherwise keep the original matching element.
			  • "OVERRIDE"
			  Just replace an existing matching original element by a matching patch element, but don't add anything if the original element does not exist. If there is no matching patch element, keep the original matching element
			  • "KEEP"
			  Copies the patch element if it exists also in original, otherwise nothing is added. The original element won't be added in any case.
			  • "COMPLETE"
			  Just add a matching patch element, but only in case there is no matching original element. Leave an original matching element otherwise, don't overwrite it.
			  • "DELETE"
			  Keeps the original element only if it does not appear in the patch document.
			  • "PRESERVE"
			  Copies the original regardless of the existence of patch element.
			  • "INSERT"
			  Copies the patch element into the output by inserting it after already existing elements of the same name. Usually applied with the matcher option "SKIP".
			  • "DTD"
			  Copy the patch element in the output parent with the correct position according to the DTD defined in the document.
			  Default: ("FULLMERGE")
			• Key begins on "mapper.":
			How the matching element should be further transformed during applying the action. There might be more transformations supported in future, filtering out elements and attributes with a specified namespace
			  • "IDENTITY"
			  Does not transform the element.
			  Default: "IDENTITY"
			• Key begins on "xpath.path.":
			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, WikiPedia and many more documents and tutorials.

<configurableset> Configuration Action - Patch a Set of Files

This action is used for handling a set of configuration files of one and the same global type at once.

A configuration set action is added using the element **<configurableset>**.

Limitations: This configuration action works currently only for type = "option" or "ini".

Attributes

Attribute	Description	Required	Allowed Values (Default)
type	The global file type of all files for which this ConfigurableSet is defined. It is not allowed to mix several file types in this action.	yes	"options" | "ini" ("options")
toDir	The target directory to recursively patch from source files of the same names to.	no (if toFile is used)	absolute or relative (against $INSTALL_PATH) path denoting a directory
toFile	The target file to patch to from all source files.	no (if toDir is used)	absolute or relative (against $INSTALL_PATH) path denoting a file
fromFile	The source file to patch from. Nested *fileset* elements can be used instead to patch from a couple of files to the target.	no (if the nested *file set* tag is used)	absolute or relative (against $INSTALL_PATH) path denoting a file
cleanup	Whether to remove all source files after the operation which don't act also as target files with the same name.	no	"true" | "false" ("false")
keepOldKeys	Whether to generally preserve equal entries but not necessarily their values from an old configuration file, if they can be found. This means if keepOldKeys="true" 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 the *keepOldValues* setting, 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.	no	"true" | "false" ("true")
keepOldValues	Whether to preserve the values of equal entries from an old configuration, if they can be found. 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.	no	"true" | "false" ("true")
resolveExpression	Whether expressions should be resolved during patching of configurations. This is not similar to substituting IzPack variables, but by means of resolving references to configuration entries in the same configuration file, see [ini4j] - Expression handling.	no	"true" | "false" ("false")
overwrite	Whether to allow overwriting existing files.	no	"true" | "false" ("false")
condition	An Izpack condition, which must be fulfilled to start the according configuration action.	no	string referring to a condition id
failOnError	Whether to abort the installation if an error occurs executing this action. If set "false", a debug log warning is written in case of an error.	no	"true" | "false" ("true")
includeEmptyDirs	Whether to include empty source directories should be also recursively copied during patching of a whole fileset to a target directory specified by *toDir*.		"true" | "false" ("true")
preservelastmodified	Whether to preserve the source file timestamps on the target files during patching.		"true" | "false" ("false")

enablemultiplemappings	Whether to apply multiple file name mappings of one and the same file name mapper for a given source file. If false the task will only process the first file or directory to map to, and not renaming one and the same file or directory to multiple files or directories. This attribute is only relevant if there is a nested *mapper* element used. Whether the situation of a potential multiple mapping can happen depends on the type of the nested *mapper* element.		"true" | "false" ("false")

Nested Elements

<fileset>

See description of the FileSet element.

<mapper>

See description of the File Name Mapper element.