Resources

Resources

Several panels, such as the license panel, the information and the shortcut panel, require additional data to perform their task.

This data is supplied in the form of resources.

The manual's panel description page lists any resources that the panel needs.

The **`<resources>`** element is a child of the **`<installation>`** element

Currently, no checks are made to ensure resources needed by any panel have been included.

Here is an example of specifying two resources:

```
<resources>
  <res id="InfoPanel.info" src="doc/readme.txt" parse="yes"/>
  <res id="LicencePanel.licence" src="legal/License.txt"/>
</resources>
```

The **<resources>** contains the **`<res>`** elements that identify each resource. The following attributes may be specified:

- **`id`**
  The identifier of the resource The **panels** documentation lists the **id**s to be used.
- **`src`**
  The path in your staging area to the resource.
- **`parse`** (optional, **yes** or **no**, default: **no**)
  Specifies whether the resource will be parsed at compilation time. Parsing will replace variables in text / xml / properties files.
- **parsexml** (optional, valid values: `true`|`false`, default: `false`)
  If set true, the resource will not be added to the installer "as is", but it is parsed as XML and the result of this parsing is added as valid XML file to the installer.
  One use case for this is if there are used <xi:xinclude /> elements in the referred resources, which must be resolved at compile time and replaced by the included XML code, in case the href attribute <xi:xinclude/> is no longer valid at installation time.
- **`type`** (optional)
  Specifies the parsing mode to adapt to specific file formats: **plain** (default), **java, shell, at, ant, javaprop, xml**.

Please note that in general a resource id is unique.

If you define multiple resources with the same id, the later definition will overwrite the previous definition. This can happen if a resource is defined in a referenced pack-file.

However there is an exception for **packsLang.xml_xyz** files (see internationalization of the PacksPanel).

If multiple packsLang-files were defined, all files will be merged into a single temporary file. This allows refpack files to provide their own internationalization-information.

The following sample shows a <resources> element in an installation file. It specifies resources required for 2 of the panels.

Sample IzPack installation description with resources

```
<installation version="5.0"
              xmlns:izpack="http://izpack.org/schema/installation"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://izpack.org/schema/installation
http://izpack.org/schema/5.0/izpack-installation-5.0.xsd">

  <info>
    <appname>Test</appname>
    <appversion>0.0</appversion>
    <appsubpath>myapp</appsubpath>
    <javaversion>1.6</javaversion>
  </info>

  <guiprefs width="800" height="600" resizable="no">
    <splash>images/peas_load.gif</splash>
  </guiprefs>

  <resources>
    <res id="InfoPanel.info" src="doc/readme.txt" parse="yes"/>
    <res id="LicencePanel.licence" src="legal/License.txt"/>
  </resources>

  <panels>
    <panel classname="HelloPanel"/>
    <panel classname="InfoPanel"/>
    <panel classname="LicencePanel"/>
    <panel classname="TargetPanel"/>
    <panel classname="InstallPanel"/>
    <panel classname="FinishPanel"/>
  </panels>

  <packs>
    <pack name="Core" required="yes">
      <description>The Sample core files.</description>
      <file targetdir="$INSTALL_PATH" src="doc/readme.txt"/>
      <file targetdir="$INSTALL_PATH" src="legal/License.txt"/>
    </pack>
  </packs>

</installation>
```