



AGETOR®

Product Packaging Guide

1.	PREFACE	3
2.	OVERVIEW	3
3.	CREATING PRODUCT ARCHIVES	3
3.1	APA – AGETOR® PRODUCT ARCHIVE.....	3
3.2	FILE TYPES AND THE PRODUCT LIST.....	3
3.2.1	<i>Product file format</i>	4
3.2.2	<i>Example product file</i>	4
3.3	PRODUCT DESCRIPTOR.....	5
3.3.1	<i>Descriptor elements</i>	5
3.3.2	<i>Example product descriptor</i>	6
3.4	BUILDING.....	7

1. Preface

Starting with AGETOR® 2 (AGETOR® products with a version number from 2.0.0 and up), all products are installed using a web-based installation tool. These products are packaged in a certain format and must be installed using the Install Tool feature of the new AGETOR® Development Kit. This guide describes how to create these packages.

2. Overview

The Install Tool creates packages by compiling source files (Java, IDL, IML) and placing the resulting jar file in the product file together with other files, which are copied into their final destination.

The steps of this process are:

1. Read and validate the product descriptor to obtain information about the product
2. Read and analyze the product file list and generate file lists for each file type
3. Generate Java classes from any IDL files
4. Compile Java source files
5. Compile IML files
6. Create a jar containing the generated classes and other jar entries, if any
7. Generate API documentation for the product unless it is an update
8. Include source files if the flag has been set
9. Generate version classes
10. Add the generated to the final package together with all the other required files

During the build and, subsequently, install process the Install Tool reads properties from a file dedicated to the Install Tool. This file is placed in `AGETOR_HOME/install/ant`.

3. Creating product archives

3.1 APA – AGETOR® Product Archive

The APA format is simply a zip file which contains the various file types of an AGETOR® product in certain locations.

3.2 File types and the product list

The AGETOR® Install Tool distinguishes between different file types. The product list specifies which files must be compiled or included directly in the final package. Below is an overview of the different file types and the prefixes used in the product list:

File type	Prefix
Java source files	java
IDL source files	idl
IML source files	iml
Library files (jar and zip files needed on the classpath)	lib
Jar entries other than Java class files	jar-entry
Web content (HTML, JSP, CSS etc.)	web
Empty directories	dir
All other files	<none>

All entries in the product list are relative to a source directory. The source directories can be specified individually for each of these types in `install.properties`.

3.2.1 Product file format

Every entry, except the *other* file type, must be on the form `<prefix>:<file pattern>`. Comments and empty lines can be used to make the list more readable. Comments begin with a pound sign, `#`.

Install Tool uses Jakarta Ant to locate source files. This means, that you can use Ant's pattern language for specifying the files to include. The main features of this language are:

- `*` matches zero or more occurrences of any character
- `?` matches exactly one character
- `**` matches zero or more directories when used in place of a directory name

You should always use `/` as file separator. Please notice that wildcards are not allowed when specifying empty directories.

If a minus sign is placed at the beginning of a Java file pattern, the package will not be included in the API documentation, e.g., `java:-com/acme/impl/*.java`. If another pattern refers to the same package and does not contain the minus sign (e.g., `java:com/acme/impl/Order*.java`), the package will be included through this pattern, though.

3.2.2 Example product file

The example below illustrates most of the possibilities of the product list format.

```
#-----
# Java
#-----
java:com/acme/*.java
# Implementation classes - do not include in API doc
java:-com/acme/impl/*.java

#-----
# IDL
#-----
idl:order.idl
#-----
# IML
#-----
iml:**/acme/*.ihtml

#-----
# Library files
#-----
lib:*.jar
lib:classes123.zip

#-----
# Jar entries - resource bundles
#-----
jar-entry:**/*.properties

#-----
```

```

# Web
#-----
web:**/*.html
web:css/*.css
web:js/*.js
web:images/**/*.*gif
web:images/**/*.*jpg

#-----
# Empty directories
#-----
dir:data/acme/temp

#-----
# Other files - conf
#-----
conf/**/acme*.cfg
conf/**/acme*.xml
conf/**/acme*.properties

#-----
# Other files - bin
#-----
bin/acme_*

#-----
# Other files - doc
#-----
doc/acme/*.*

```

3.3 Product descriptor

The product descriptor is an XML file that serves two purposes: It identifies the product and its dependencies, and it defines how the product is configured during installation.

3.3.1 Descriptor elements

The following sections describe the most important elements of the product descriptor.

Product id and description

The product is identified by a product code, usually called its name, which must be unique among all products based on AGETOR®. In addition to this, a three-number version code is used to distinguish different versions of the product. The complete id has the form `<product name> <version number>.<upgrade number>.<patch number>`, e.g., `adk 2.0.0`. This corresponds to the generated product file, which will be called `adk2_0_0`.

Build number

Each product is also equipped with a build number that is generated at build time. The build number can be used for distinguishing different versions during development as it consists of the date and time of the build: `yyyyMMddHHmmss`. As the build number is generated it must have a special value (`@BUILD@`) in the original descriptor that is replaced during the build process.

Dependencies

If the product depends on other products these products are entered in the descriptor with their full id. Product updates contain an attribute, `UPDATE_FROM`, that indicates the patch number that is

updated and this will automatically be perceived by the Install Tool as a dependency; e.g., acme 3.1.5 update 6 can only be installed if acme 3.1.5 is already installed.

Configurations

A configuration element in the descriptor is bound to a certain file and has up to three functions during installation:

- A file can be moved from one (default) location to another, If it already exists the user can decide to overwrite it
- The user can be prompted for values of different variables which are then inserted into the file
- If the configuration is a service configuration it can add the service to the broker configuration

Each contains zero or more questions which corresponds to a variable input in the web interface. The values of the variables are replaced in the original file during copy to its final destination. The destination is either given explicitly, or implicitly, by letting a default directory "def" be part of the source path. For example, the file `acme.properties` in the descriptor below will be moved from `AGETOR_HOME/conf/properties/def` to `AGETOR_HOME/conf/properties`.

Service configuration makes it possible to add a service to the broker configuration using its relative port. To do this the service configuration must contain a service question with information on the relative port and the question numbers. A service configuration can contain service questions and ordinary questions at the same time.

3.3.2 Example product descriptor

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE PRODUCT
  PUBLIC "-//Bording Data//DTD Product Descriptor 1.0//EN"
  "ant/product-descriptor_1_0.dtd">

<PRODUCT NAME="acme" VERSION="3" UPGRADE="1" PATCH="5" BUILD="@BUILD@"
DESCRIPTION="Acme Turbo">
  <DEPENDENCY NAME="adk" VERSION="2" UPGRADE="0" PATCH="0"/>
  <CONFIGURATION
    DESCRIPTION="Acme properties"
    FILE="conf/properties/def/acme.properties"
  >
    <QUESTION
      DESCRIPTION="Minimal Acme faktor"
      NAME="min_acme"
      VALUE="42"
      TOKEN="MIN_ACME"
    />
    <QUESTION
      DESCRIPTION="Maximal Acme faktor"
      NAME="max_acme"
      VALUE="117"
      TOKEN="MAX_ACME"
    />
  </CONFIGURATION>
  <CONFIGURATION
    DESCRIPTION="Acme servlets"
    FILE="conf/def/acme_servlets.cfg"
    TOFILE="conf/servlets/acme_servlets.cfg"
```

```

/>
<SERVICE-CONFIGURATION
  DESCRIPTION="Acme service"
  FILE="conf/services/def/acme_services.cfg"
>
  <SERVICE-QUESTION
    DESCRIPTION="Relative port"
    NAME="rport"
    SERVICE="acme"
    VALUE="666"
    TOKEN="RPORT"
    QNO="9999"
  />
  <QUESTION
    DESCRIPTION="Acme service parameter"
    NAME="param"
    VALUE="990"
    TOKEN="PARAM"
  />
</SERVICE-CONFIGURATION>
</PRODUCT>

```

3.4 Building

The product archive is created by issuing a command in an AGETOR shell:

```
build-install <name> [-s]
```

The product list and the product descriptor must be placed in AGETOR_HOME/install and have the names <name>.list and <name>.xml respectively. The name is arbitrary but it is recommended that it follows the naming system of the final package:

```
<product name><version>_<upgrade>_<patch>[update<update patch>]
```

For instance:

```
acme3_1_5 (file names acme3_1_5.list and acme3_1_5.xml)
acme3_1_5update6 (file names acme3_1_5update6.list and acme3_1_5update6.xml)
```

The “-s” option instructs the Install Tool to create a separate zip file containing Java, IDL, and IML source files. This file is placed in the library directory.