Tizen SDK Package Guide

Table of Contents

Introduction

- New packaging format has been implemented for the development of Tizen SDK.
- This guide provides the format/description of files required to generate Tizen SDK packages.
- There four kinds of files
  - package/pkginfo.manifest : Package Information
  - build.{BUILD HOST OS} : build script
  - {package name}.install.{TARGET OS} : post install script
  - {package name}.remove.{TARGET OS} : post remove script

package/pkginfo.manifest

- This file defines various information that are required for packaging
- This file consists of several sections, and each section defines several fields

Section and Field

Section

- Section means group of fields that is separated by two newline characters (" \ n \ n").
- There are two kinds of sections: common section and package section
- Common Section
  - It has information of all the files in this package.
  - Available fields : Source Name(**Source**), Version Info(**Version**), Maintainer's name/email(**Maintainer**)
- Package Section
  - It has information for each package.
  - Available fields: Package name(**Package**), Target OS(**OS**), Build OS(**Build-host-os**), Install dependent packages(**Install-dependency**), Build dependent packages(**Build-dependency**), Source dependent(**Source-dependency**), **Attribute**, **Description**
- Common section must be at the top of the sections list

Field

- The field includes details for each package.
- “Field” format is like this : {field name} : {field value} [, {field value}]...
- Most of fields are separated by a newline characters (" \ n").
  - Description field can have multi line
- Some fields are optional and can be skipped

Common Section

Source (Required)

- It describes a source project name.
- Ex. Source:nativecommon-eplugin

Version (Required)

- It describes the version information of the package.
- Format : {major}.{minor}.{patch level}
  - {major}, {minor}, {patch level} are integers
- Version must be increased for Tizen SDK Package when it is uploaded to Package-server.
- Ex. Version: 0.14.32

Maintainer (Required)

- It describes a maintainer's name and email information.
- It consists of "{name} <{email}>" and could be described by more than one maintainers using comma separator.
- Ex. Maintainer: JongHwan Park <jonghwan2.park@samsung.com>

Package Section

- Information for each package
- The file name of generated package has the following format: {Package}_{Version}_{OS}.zip

Package (Required)

- It describes the package name.
- The first character must be an alphabet. And allowed characters are: Alphabets(a-z), digits(0-9), and dash(-)
- Ex. Package: native-ide

OS (Required)

- It describes the name of the OS required to install the package.
- Allowed OS names : ubuntu-32, ubuntu-64, windows-32, windows-64, macos-64
- You can specify compatibility over various OSs by describing several OSs separated by comma.
- Ex. OS: ubuntu-32
- Ex. OS: ubuntu-32, ubuntu-64

Build-host-os (Required)

- It describes the name of Host-OS on which the package is built.
- Allowed OS names : ubuntu-32, ubuntu-64, windows-32, windows-64, macos-64
- You can specify a name of the OS.
- If you want additional Build-Host, you can use "pkginfo.manifest.local"
- Ex. Build-host-os: ubuntu-32

Install-dependency (Option)

- It describes the dependent packages that are needed to install this package.
- Format : Package name(Version condition)

- The described packages must have same OS.
- “Version condition” is composed of operators and versions in parentheses.
- Allowed Version condition operators are : "<<", "<=", "=", ">=", ">>"
- Version condition may be omitted, which means that there is no condition.
- Multiple packages can be specified by separating them with comma(,)
- Ex. Install-dependency: CodeCoverage (>= 0.3.0), rootstrap-slp-device-1.0

Build-dependency (Option)

- Describes the dependent packages that are needed to build this package.
- Format : Package name(Version condition)[OS]
- “Version condition” is composed of operators and versions in parentheses.
- Allowed Version condition operators : "<<", "<=", "=", ">=", ">>"
- Version condition may be omitted, which means that there is no condition.
- If OS is specified, the packages of the target OS are needed. It need not be same as its target OS.
- Allowed Os names : ubuntu-32, ubuntu-64, windows-32, windows-64, macos-64
- Multiple packages can be specified by separating them with commas(,)
- Ex. Build-dependency: CodeCoverage (>= 0.3.0) [ubuntu-32], rootstrap-slp-device-1.0

Source-dependency (Option)

- It describes the archived source file and list for build packages.
- When you build the package, the source file list is downloaded from Package-server.
- It can be used when packages do not have any modifications in the source code.
- Format : Source-dependency : {source} [, {source}]...
- EX. Source-dependency: gcc-linaro4.5-2012.01.tar.gz2, gmp-5.0.1.tar.gz, mpc-0.9.tar.gz, mpfr-3.0.1.tar.gz, binutils-2.22.tar.gz

Attribute (Option)

- Specify additional attributes
- Attribute
  - root : it shows install package list of installer.
  - install : it shows install type of installer.

        - public : it shows install package list of public installer.
        - partner : it shows install package list of partner installer.binary : package Does not build on the build server.
    - EX. Attribute : binary
    - EX. Attribute : root

Label (Option)

- This text is displayed in the installer
- text can include space
- EX. Label : Native IDE

Description (Option)

- It is package description
- Only this field has multi line value
- Ex. Description : Install native ide

package/pkginfo.manifest.local

- This file is for supporting local build on other build host OS.
- And it is valid only for "pkg-build" command.
- it can specify various BUILD_HOST_OS at a time

  EX. Build-host-os : ubuntu-64, macos-64

- To use pkginfo.manifest.local you must generate the build.{other BUILD HOST OS}

Include

- includes other file
- It is usually used to include pkginfo.manifest in pkginfo.manifest.local file

  EX. Include : pkginfo.manifest

build.{BUILD HOST OS}

- This file contains the build and packaging script on {BUILD HOST OS}
- You can use {OS Category}(linux, winodws, macos) instead of {BUILD HOST OS}

- this script is based on bash shell script
- You must implement some required functions
  - clean() : clean job before build
  - build() : build job
  - install() : make Tizen SDK folder structure using Build result
- You can use some built-in environment variables
  - SRCDIR : source dir path
  - ROOTDIR : path dependent package will be installed
  - TARGET_OS : OS name to be installed
  - TARGET_OS_CATEGORY : OS category name to be installed
  - VERSION : package version

clean()

- This function specifies the code to clear the various files that are generated during the build.
  - temporary file
  - result package file: *_*_*.zip
- if build uses Makefile then type "make clean"

  EX. rm -rf ${SRCDIR}/*.zip
  EX. make clean

build()

- Defines the tasks that generates the various files to be included in the package.
- If there is a Source-Dependency defined in pkginfo.manifest file then you can unzip that file too.

  EX. tar xvf ${ROOTDIR}/gcc-linary-4.5-2012.01.tar.gz2

- Set build environments and build
- If build uses Makefile then call "make" here

  EX. make

install()

- Define copying of the various files generated by the build process into their designated position in Tizen SDK directory structure.
- You need to create the package root directory of Each package
  - Package root directory : package/{package name}.package.{TARGET_OS}/data

    - Package root directory is mapped to actual Tizen SDK install directory

  EX. mkdir -p ${SRCDIR}/package/cross-arm-gcc-4.5.package.${TARGET_OS}/data

- Copy all the files to be included in the package.

  EX. mkdir -p ${SRCDIR}/package/cross-arm-gcc-4.5.package.${TARGET_OS}/data/tools/gcc-4.5
  EX. mv ${SRCDIR}/bin/* ${SRCDIR}/package/cross-arm-gcc-4.5.package.${TARGET_OS}/data/tools/gcc-4.5

- If you use "makefile" to build your packages, then use "make install" here in general

  EX. make install --prefix=${PKG_INSTALL_DIR}

{Package Name}.install.{TARGET_OS}, {Package Name}.remove.{TARGET_OS}

- *.install.* : Describes the additional actions required after files have been copied to the installation location, when the package is installed
- *.remove.* : Describes the additional actions after files have been removed from the installation location, when the package is uninstalled.
- It is not required to create these files.
- You can use {OS Category} instead of {TARGET_OS}
- EX. pkg-a.install.linux
- Supports batch script for windows and bash shell script for other os
- You can use environment variable : INSTALLED_PATH
    - INSTALLED_PATH : TIZEN SDK Install directory
    - for windows (batch script)
- EX. SET PATH=%INSTALLED_PATH% \ tools \ smart-build-interface \ bin;%PATH%
    - for linux (shell script)
- EX. PATH=${INSTALLED_PATH}/tools/smart-build-interface/bin:${PATH}
- Installer determines script's success or failure, using exit code of the script
    - 0 : all the work is successfully completed.
    - 1 ~ 9 : A few minor errors, but you can ignore.

        - else : Critical error occurs, the installation does not proceed further.
- EX. exit 1