ZyXEL NSA Device

User-Made Packages Installation Guide

Version: 0.3

Introduction

The purpose of this document is to teach user how to generate a personal package and then install it to ZyXEL NSA series products. With this document, you will learn how to

- generate an user-made package that can be used in ZyXEL NSA series products
- prepare a package server in your NSA device
- install the user-made package to your NSA device

How to generate user-made package

- Download toolkit

  Please download the toolkit from ZyXEL official web site

  ftp://opensource.zyxel.com/NSA-220%20PLUS/v3.12/montavista_toolkit.tar.gz

- Compile your package

  Use the downloaded toolkit to build the application (open source or written by yourself).

- Generate package

  The package is composed of two parts:

```
Package --- control.tar.gz    → package information and some action scripts
        |- data.tar.gz        → all binary files, library, WebGUI codes, and conf files

<working directory>
package --- metafiles         → put all files needed by control.tar.gz
        |- data               → put all files needed by data.tar.gz
```

control.tar.gz

It contains ***control*** file to record the package information, and (not necessary) some action scripts to control the *install / uninstall* process.

control.tar.gz
- |– *control* → package information
- |– action scripts → scripts to control the install/uninstall process including *preinst*, *postinst*, *prerm*, and *postrm*

Let's enter **metafiles** folder and create these files.

control

It contains several fields:

Field	Description
Package	The package name. Ex. ***MyPkg***
Version	Format: **<PackageVersion>zypkg<SerialNO>** **PackageVersion** is the version of the open source application you used or the same as **SerialNO** if you are the author of this package. **SerialNO** is a serial number used to control package upgrading process. **There must be no space in the version string.** For example, If you want to generate a package based on the open source application ***OpenApp*** with version ***1.45***, the **PackageVersion** will be ***1.45***, and the **SerialNO** will be ***001***. The entire version is ***1.45zypkg001***. After several months, you want to do some modification for this package, **SerialNO** will be incremented to ***002***, and the entire version will be ***1.45zypkg002***. If you just write your own package ***MyPkg***, the version will be ***001zypkg001***.
Description	The description for this package. It will be displayed in package list.
Depends	What package is needed before install this one. Put the package name needed by this package here. If more than one, separate them by comma. For example, if we need ***MyDepPkg1*** and ***MyDepPkg2*** before running ***MyPkg***, the Depends field of ***MyPkg*** should be: Depends: MyDepPkg1, MyDepPkg2

Size	The compressed package size in bytes (just give a approximate number).
Installed-Size	The installed (uncompressed) package size in bytes.
Architecture	Always **arm** here now.
Zy-Model	Give **NSA-220** here.
IsBuiltin	Always **N** for user-made package
Filename	The final package file name. The format is **<PackageName>_<PackageVersion>_arm_<SerialNO>.zpkg** Ex. ***OpenApp_1.45_arm_001.zpkg*** ***MyPkg_001_arm_001.zpkg***

Take ***MyPkg*** as an example:

```
Package: MyPkg
Version: 001zypkg001
Description: This is a testing package.
Depends:
Size: 23595056
Installed-Size: 67543040
Architecture: arm
Zy-Model: NSA-220
IsBuiltin: N
Filename: MyPkg_001_arm_001.zpkg
```

Action scripts (optional) - *preinst*, *postinst*, *prerm*, *postrm*

Scripts	Description
preinst	A script to do something necessary *before installing process started*
postinst	A script to do something necessary *after installing process finished*

prerm	A script to do something necessary *before uninstalling process started*
postrm	A script to do something necessary *after uninstalling process finished*

The four scripts ***perinst***, ***postinst***, ***prerm***, and ***postrm*** are used to control the install/uninstall process. It follows the script format. That is, it must begin with **"#!/bin/sh", and be excutable**. There are some examples to describe when you need them as following:

If you need to check if this package is installed before installing this package, add ***preinst*** as:

```
#!/bin/sh

# this is preinst for MyPkg
if [ -f /usr/local/zy-pkgs/etc/init.d/MyPkg ]; then
   # already installed
   exit 0
fi
```

If you need to add a user "testman" after installing this package, just add ***postinst*** script with contents:

```
#!/bin/sh

# this is postinst for MyPkg
cat /etc/passwd | grep "^testman"
if [ "$?" != 0 ]; then
       echo 'testman:x:0:0:testman:/:/bin/sh' >> /etc/passwd
fi

cat /etc/shadow | grep "^testman"
if [ "$?" != 0 ]; then
       echo
'testman:$1$$iC.dUsGpxNNJGeOm1dFio/:13493:0:99999:7:::' >> /
fi
```

If you need to stop specified daemon before uninstalling this package, add ***prerm*** as:

```
#!/bin/sh

# this is prerm  for MyPkg
killall MyPkgd
sleep 1
```

If you need to remove some temporary files in **/tmp/MyPkg** after uninstalling this package, just add ***postrm*** script with contents:

```
#!/bin/sh

# this is postrm for MyPkg
rm –rf /tmp/MyPkg
```

When ***control*** and **action scripts** needed are ready, put them in the same folder **metafiles** and use tar to generate **control.tar.gz**

```
tar zcvf ../control.tar.gz *
```

data.tar.gz

It contains all data you want to install in system. All data in package will be installed to **/usr/local/zy-pkgs** automatically. The file structure will be:

```
data.tar.gz
    |– bin/          → put all binary files for this package
    |– lib/          → put all library files used for this package
    |– gui/          → if this package has web gui interface, put all related files here
    |– etc/          → put config files for thjs package here
        |– init.d/
            |– <pkg_name>
                → a functional script for this package. It can be used to:
                    – get package status and url link
                    – enable or disable package
                    – do something needed when system booting up
```

Let's enter **data** folder and generate all data files in it.

◆ bin

This folder keeps all binary files for your package. It is recommended that you put all binary files (in sbin, usr/bin, usr/sbin, … etc) in this folder (bin). After installing this package, all files in this folder will be put to **/usr/local/zy-pkgs/bin**.

◆ lib

It keeps all library files used for your package. After installing this package, all files in this folder will be put to **/usr/local/zy-pkgs/lib**.

◆ gui

If you have Web GUI interface to manage your package, put all related files (html/php/…etc) here. To avoid overwrite files of other package, you should create a folder(often the package name) to store them. Remember to put the file "**index.html**" as the entrance. For example, if you have some gui files for package **MyPkg**, you should put them in **gui/MyPkg**. There must be **index.html** in it. After installing this package, all files in this folder will be put to **/usr/local/zy-pkgs/gui**. And then you can access them via **http://<ip>:<port>/pkg/MyPkg**.

◆ etc/init.d/<pkg_name>

It's a script file to display package status and url, enable or disable package, or do something needed when booting up. There are 6 kinds of arguments:

Argument	Description
getlink	This argument returns the Web GUI URL to manage your package. It will be displayed in "Management Page" field of package list. If your package did't have Web GUI interface, just return empty string here. The URL format is: **<protocol>://<ip>:<port>/pkg/<path>** The corresponding path is: **/usr/local/zy-pkgs/gui/<path>** The <path> always begins with the package name, and there must be entrance file **index.html** in it. Ex. ***http://172.23.26.223:80/pkg/MyPkg*** The responding path is: **/usr/local/zy-pkgs/gui/MyPkg**

status	This argument returns the status of this package. There are two values: **Enabled** and **Disabled**.
startup	This argument will be called when system booting up. If the package is enabled when last shutdown, we should enable it here.
shutdown	This argument will be called when system shutdown. You should stop daemons triggered by the package here.
enable	Enable this package. It will be triggered when you click "**Enable**" button in package list.
disable	Enable this package. It will be triggered when you click "**Disable**" button in package list.

What you need to do next is to compress them into an archive file. Put all of them to the folder **data** and execute:

```
tar zcvf ../data.tar.gz *
```

When data.tar.gz and config.tar.gz are both ready, you can generate package file by:

```
cd .. ; tar zcvf MyPkg_001_arm_001.zpkg data.tar.gz config.tar.gz
```

A use-made package example

Let's take ssh daemon as an example. If we want to wrap a ssh daemon to a package named ***MySSHD***, we should do following steps:

1. Use toolkit to compile and generate all binary files. We got the following file structures:

```
MySSHD
    |- metafiles
    |- data    → put all files you generated here
        |- gui --- MySSHD --- index.html
        |- etc --- MySSHD --- sshd_config
        |    |- init.d --- MySSHD
        |- config
        |- bin --- sshd
                |- ssh-keygen
```

2. Generate **data.tar.gz**

```
cd MySSHD/data ; tar zcvf ../data.tar.gz *
```

3. Prepare **config** file

```
cd ../metafiles ; vi config
```

```
Package: MySSHD
Version: 001zypkg001
Description: This is a testing package to adding sshd to system.
Depends:
Size: 23595056
Installed-Size: 67543040
Architecture: arm
Zy-Model: NSA-220
IsBuiltin: N
Filename: MySSHD_001_arm_001.zpkg
```

4. Prepare action files

We needs **postinst** and **prerm** here.

vi postinst

```
#!/bin/sh

PKGNAME="MySSHD"
PKG_ROOT="/usr/local/zy-pkgs"
ZYPKG_DEPS="${PKG_ROOT}/etc/init.d/ZYPKG_DEPS"
CONFIG_PATH="${PKG_ROOT}/etc/${PKGNAME}"
STATUS_FILE="${CONFIG_PATH}/STATUS"

# main start

# create tty (allow 3 connections at the same time)
mknod /dev/ptyp0 c 2 0
mknod /dev/ptyp1 c 2 1
mknod /dev/ptyp2 c 2 2
mknod /dev/ttyp0 c 3 0
mknod /dev/ttyp1 c 3 1
mknod /dev/ttyp2 c 3 2

# generate key
${PKG_ROOT}/bin/ssh-keygen -t rsa1 -f ${CONFIG_PATH}/ssh_host_key -N ""
${PKG_ROOT}/bin/ssh-keygen -t dsa -f ${CONFIG_PATH}/ssh_host_dsa_key -N ""
${PKG_ROOT}/bin/ssh-keygen -t rsa -f ${CONFIG_PATH}/ssh_host_rsa_key -N ""

exit 0
```

vi prerm

```
#!/bin/sh

PKGNAME="MySSHD"
PKG_ROOT="/usr/local/zy-pkgs"
ZYPKG_DEPS="${PKG_ROOT}/etc/init.d/ZYPKG_DEPS"
CONFIG_PATH="${PKG_ROOT}/etc/${PKGNAME}"
STATUS_FILE="${CONFIG_PATH}/STATUS"

# main start
# stop daemon
killall sshd
sleep 1

# remove tty
rm /dev/ptyp0
rm /dev/ptyp1
rm /dev/ptyp2
rm /dev/ttyp0
rm /dev/ttyp1
rm /dev/ttyp2


# remove key
rm -rf ${CONFIG_PATH}

exit 0
```

5. Generate **config.tar.gz**

```
tar zcvf ../config.tar.gz *
```

6. Generate package

```
cd .. ; tar zcvf MySSHD_001_arm_001.zpkg data.tar.gz config.tar.gz
```

7. Done. A MySSHD package is generated.

Prepare package server in NSA device and insall the user-made package

After the package generated, you need to put it to a server that can be retrieved via Package page. To let the NSA device be a package server, you need to:

1. Go to **Share** page and create a public share named ***packages***.

2. Go to **Web Publishing** page, enable Web Publishing and let ***packages*** published.

3. Create folder in packages share according to your device Model Name (shown in **Status** page)

Model Name	folder
NSA-220	NSA-220/zypkg
NSA-220 PLUS	NSA-220_Plus/zypkg
NSA210	NSA210/zypkg
ServerBox	ServerBox/zypkg

4. Get the package archive from ZyXEL official web site

   http://www.zyxel.com.tw/zyxel/product/prod_download.php?ln=03&no=0000843

5. Extract the package archive to the folder created in step 3.

6. Use 7 zip to extract **i-data/md0/admin/zy-pkgs/ZYPKGS** from ZYPKG_INFO.tgz. Add the control contents to the end of it and overwrite the original file.

7. Use 7 zip to extract **usr/local/zy-pkgs/etc/init.d/ZYPKG_DEPS** from **ZYPKG_INFO.tgz**. Add this package to ***START-UP*** and ***SHUTDOWN*** list according to the dependency order and overwrite the original file. For example, if **MyPkg** depends on **MyDepPkg1**, the **ZYPKG_DEPS** file should be:

```
# START-UP (DON'T REMOVE THIS LINE!)
……..
/usr/local/zy-pkgs/etc/init.d/MyDepPkg1
/usr/local/zy-pkgs/etc/init.d/MyPkg

# SHUTDOWN (DON'T REMOVE THIS LINE!)
……
/usr/local/zy-pkgs/etc/init.d/MyPkg
/usr/local/zy-pkgs/etc/init.d/MyDepPkg1
```

   If no dependency for this package, just add it to the ends of ***START-UP*** and ***SHUTDOWN*** list.

8. Add **web_prefix** file to **zy-pkgs** folder in **admin** share. Let it contents be the Example in **Web Publishing** page. For example:

```
http://172.23.26.226:80/packages/
```

8. Go to Package page and click "**Retrieve List From Internet**" button, you'll see the package list with this package inside, just select it and click "**Install/Upgrade**" button. The package will be installed after a while.