DIBS Advanced Guide(English)

Table of contents

Introduction

Setup Build System

This setup guide will assume that the "Setup Build Environment" steps of "Tizen SDK Development Guide" are already done.

Setup Package Server

Create Package Server

You can create package server using "pkg-svr" command. Here is the command for creating package server.

```
## pkg-svr create -n <server name> -d <distribution> [-u <remote server url>] [-l <location>]
## -n option : Your package server name
## -d option : Your package server distribution name
## -u option : Your package server distribution's parent distribution url  (ex:
http://127.0.0.1/testserver/unstable)
## -l option : Your package server location
```

And here are the simple steps

1. Create a package server root directory and move to it
   - ex) **$> mkdir pkgsvr; cd pkgsvr**
2. Create package server using "pkg-svr" command
   - ex) **$> pkg-svr create -n pkgserver -d unstable**
3. Check the config file of your package server
   - ex) **$> cat ~/.build_tools/pkg_server/pkgserver/config**

If you want to create your own package server by pulling all packages from other server, use "-u" option.

1. In your package server directory, type this
   - ex) **$> pkg-svr create -n mypkgserver -d develop -u <package_server_address>/<package_server_name>/<distribution_name>**
2. Then server will include the latest packages which are from the original server.
3. If you want to refresh your package server from remote server, use "pkg-svr syncronize" command

Start Package Server

After creating you package server, just start using the following command

```
## pkg-svr start -n <server name> -p <port> [-w <passwd>]
## -n option : Your package server name
## -p option : Port number
## -w option : Password
```

For example, next command will start your package server using port number 3333

1. Just type the start command
   - ex) $> pkg-svr start -n pkgserver -p 3333

Setup Build Server(s)

Create Build Server

You can create build server using "build-svr" command. Here is the command for creating build server.

```
## build-svr create -n <server name> -u <package server url> -d <package server address> -t <ftp server url>
## -n option : Your build server name
## -u option : Package server url (ex: http://127.0.0.1/testserver/unstable)
## -d option : Package server address (ex: 127.0.0.1:3333)
## -t option : FTP server url (ex: ftp://dibsftp:dibsftp@127.0.0.1)
```

To create build-server, package server url must be specified. This url will be used to get(download) the packages from the package server. And it can be absolute directory path on your local system, if you have package-server in the same machine.

And you need FTP server for file transfer between your servers. You can setup new ftp server on you build system or can use your existing ftp server.

Here are simple steps.

1. Create the root directory of build-system and change the directory to it
   - ex) $> mkdir buildsvr; cd buildsvr
2. Create a build server
   - ex) $> build-svr create -n buildsvr -u http://<package_server_address>/<package_server_name>/<distribution_name> -d <your_ip>:<build_server_port_number> -t ftp://<ftpid>:<ftppw>@<package_server_ip>
   - ex) $> build-svr create -n buildsvr -u /home/userid/<package_server_location>/<package_server_name>/

<distribution_name> -d <your_ip>:<build_server_port_number> -t ftp://<ftpid>:<ftppw>@<package_server_ip>

3. Check the config file of your server
    - ex) $> cat ~/.build_tools/build_server/buildsvr/server.cfg

Add Projects to Build Server

You should register information for a project ~~what~~which you want to build, before building a project. Here is the command for registering.

```
## build-svr add-prj -n <server name> -N <project name> (-g <git repository> -b <git branch>|-P <package name>)
[-w <password>] [-o <os list>]
## -n option : Your build server name
## -N option : Project name
## -g option : Git repository for project (ex: gerrithost:/sdk/tools/sdk-build)
## -b option : Git branch (ex: develop)
## -P option : Prefix of package name
## -w option : Password for managing project. If you set password, it is used when building project
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
```

Project has two types, normal project and binary project.

- Normal project builds from build server and uploads to package server
    - Normal project uses -g and -b option to clone source using git.
- binary project is just an upload package file to the build server and the build server uploads it to the package server.
    - binary project uses -P option that is needed in package naming
    - binary package is named <prefix of package name>_<package version>.tar.gz

For example,

1. Register information for project
    - ex) $> build-svr add-prj -n buildsvr -N dibs -g gerrithost:/sdk/tools/sdk-build -b develop
    - ex) $> build-svr add-prj -n buildsvr -N test -g gerrithost:test/a -b develop -w <project_passwd> -o ubuntu-32,windows-32

Edit Server Configurations

You can edit server configurations manually. Configuration file is in "~/.build_tools/build_server/{build server name}/server.cfg"

```
ID=develop
PATH=/build/build_server/private/develop
PSERVER_URL=http://<package_server_address>/<package_server_name>/<distribution_name>
PSERVER_ADDR=<package_server_ip>
PSERVER_ID=
GIT_BIN_PATH=/usr/bin/git
ALLOWED_GIT_BRANCH=develop
MAX_WORKING_JOBS=2
JOB_LOG_URL=
SEND_MAIL=NO
JOB_KEEP_TIME=86400
FTP_ADDR=<ftp_address>
FTP_USERNAME=<ftp_user_name>
FTP_PASSWD=<ftp_user_password>
```

If your build server is windows, then you should check GIT_BIN_PATH. Example)

```
GIT_BIN_PATH=/C/msysgit/bin/git.exe
```

Start Build Server

You can start build server using the following command

```
## build-svr start -n <server name> -p <port>
## -n option : Your build server name
## -p option : Server port number
```

Here is an example

1. Type the next command
    - ex) **$> build-svr start -n <build_server_name> -p <build_server_port>**

Add Remote Build Server

To support multi-OS or distribute your build jobs, you need more build-servers. Here are the steps for adding remote build server

First, you have to setup new build-server on your remote server by the steps mentioned above.

- Install Tizen SDK -> Create Build Server -> Add Project to Build Server -> Start Build Server

After creating remote server, you have to add this server to main(entry) build-server

```
## build-svr add-svr -n <server name> (-d <friend server address>|-u <remote pkg server url>)  [--proxy <proxy url>]
## -n option : Your build server name
## -d option : Remote server address
```

Here is an example.

1. Type the command
   - ex) $> build-svr add-svr -n buildsvr -d <remote_build_server_ip>:<remote_build_server_port>

Register Binary-Only/Source-Archive Packages

After you set up your build server, you need to register some packages manually.

- Binary-only: this is the package that cannot be built by build-server for some reason or policies.
  - ex) mingw32-msys-1.0, indigo_pde, ...
- Source-Archive: this is the source code that is used to build your project. This source code is open-source project that is used but not modified by the project
  - ex) gcc-linaro-4.5-2012.01.tar.bz2, binutils-2.22.tar.gz, ...

You can register the packages using following command

```
## build-svr register -n <server name> -P <pacakge file>
## -n option : build server name
## -P option : package/archive file path
```

For example

1. Just type next command
   - ex) $> build-svr register -n buildsvr01 -P mingw32-msys-1.0_0.21.3_windows.zip
   - ex) $> build-svr register -n buildsvr01 -P gcc-linaro-4.5-2012.01.tar.bz2

Bootstrapping Package Server

Now, that all servers are working and we need to build all your projects and upload them to package server. The following command will help in this process.

```
## build-svr fullbuild -n <server name>
## -n option : build server name
```

And here is an example.

1. Type the command
   - ex) **$> build-svr fullbuild -n buildsvr**

Optional Steps

Setup FTP server

- You should setup FTP server for transferring package files.
  - You should restrict local users to their home directories.
- Ex) Installing vsftpd on ubuntu.

```
$> sudo apt-get install vsftpd
$> vi /etc/vsftpd.conf (Remove the annotation of following sentence)
   write_enable=YES
   chroot_local_user=YES
```

Setup Web Server for log view

This guide will describe the steps for using "Apache2" web server. You can also setup other web servers by referring to these steps.

linux

- First install Apache2 for your machine

```
sudo apt-get install apache2
```

- Added new directory of build server root to config file (/etc/apache2/sites-availables/default)

```
VirtualHost *:80>
    ...
    Alias /build-server-name}/ /build/server/root/path/
    <Directory /build/server/root/path>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>
    ...
```

- Start tomcat server

```
/etc/init.d/apache2 start
```

- Added job-log url to build-server configuration(~/.build_tools/build_server/build-server-name/server.cfg).

```
 ...
 JOB_LOG_URL=http://<local_pc_ip>/<build_server_location>/<build-server-name>/jobs
 ...
```

windows

- First install Apache Tomcat for your machine
  - Tomcat's install path is TOMCAT_HOME
- Added new directory of build server root to config file (TOMCAT_HOMEconfserver)

```
<Service ...
...
<Engine ...
...
<Host name="localhost"  ...

    <Context path="/build-server-name"
             docBase="/path/to/build/server/root"
             reloadable="true"
             crossContext="true" />
</Host>
</Engine>
</Service>
```

- Run tomcat: Execute 'TOMCAT_HOMEbinstartup'
- Add job-log url to build-server configuration(~/.build_tools/build_server/build-server-name/server.cfg).

```
...
JOB_LOG_URL=http://<local_pc_ip>/<build_server_location>/<build-server-name>/jobs
...
```

- Execute 'TOMCAT_HOME \ bin \ startup' file
- Added job-log url to build-server configuration(~/.build_tools/build_server/build-server-name/server.cfg). In this example,

```
...
JOB_LOG_URL=http://<local_pc_ip>/<build_server_location>/<build-server-name>/jobs
...
```

Setup Email Notification

- Install "postfix" application in your machine

```
sudo apt-get install postfix
```

- Setup "postfix" configuration.

```
sudo dpkg-reconfigure postfix
```

- Change ‘send mail’ option of build-server configuration(~/.build_tools/build_server/build-server-name/server.cfg)

```
...
SEND_EMAIL=YES
...
```

Package server integrity check

- Package server does integrity check when package is registered.
    - Check for install, build and source dependent packages if it exists in package server.
- But, if you want to skip this integrity check then modify config file(~/.build_tools/package_server/package-server-name/config)

```
...
integrity check : NO
```

```
...
```

DIBS Command Reference

pkg-svr: Package Server

- "DIBS" can create the package server , register the package and delete the package server.
- In a single PC you can have multiple package servers and each package server should have a unique name.
- In a single package server you can have multiple distributions, and each distribution should have a unique name.
- Package server can have a parent package server, the child package server can be synchronized with the parent package server.
- You can control package server using "pkg-svr" command.

```
Package-server administer service command-line tool.

Usage: pkg-svr <SUBCOMMAND> [OPTS] or pkg-svr (-h|-v)

Subcommands:
          create          Create a package-server.
          add-dist        Add a distribution to package-server.
          add-os          Add supported os.
          register        Register a package in package-server.
          remove          Remove a package-server.
          remove-dist     Remove a distribution to package-server.
          remove-pkg      Remove a package in package-server.
          remove-snapshot Remove a snapshot in package-server.
          gen-snapshot    Generate a snapshot in package-server.
          sync            Synchronize the package-server from parent package server.
          start           Start the package-server.
          stop            Stop the package-server.
          clean           Delete unneeded package files in package-server.
          list            Show all pack

Subcommand usage:
          pkg-svr create -n <server name> -d <distribution> [-u <remote server url>] [-l <location>]
          pkg-svr add-dist -n <server name> -d <distribution> [-u <remote server url>] [--clone]
          pkg-svr add-os -n <server name> -d <distribution> -o <os>
          pkg-svr register -n <server name> -d <distribution> -P <package file list> [--gen] [--test]
          pkg-svr remove -n <server name>
          pkg-svr remove-dist -n <server name> -d <distribution>
          pkg-svr remove-pkg -n <server name> -d <distribution> -P <package name list> [-o <os>]
          pkg-svr remove-snapshot -n <server name> -d <distribution> -s <snapshot list>
          pkg-svr gen-snapshot -n <server name> -d <distribution> -s <snapshot name> [-b <base snapshot name>]
          pkg-svr sync -n <server name> -d <distribution> [--force]
          pkg-svr clean -n <server name> -d <distribution> [-s <snapshot list>]
          pkg-svr start -n <server name> -p <port>
          pkg-svr stop -n <server name> -p <port>
          pkg-svr list [-n <server name>]
```

```
Options:
        -n, --name <server name>        package server name
        -d, --dist <distribution>       package server distribution
        -u, --url <server url>          remote server url: http://127.0.0.1/dibs/unstable
        -o, --os <operating system>     target operating system
        -P, --pkgs <package file list>  package file path list
        -s, --snapshot <snapshot>       a snapshot name or snapshot list
        -b, --base <base snapshot>      base snapshot name
        -l, --loc <location>            server location
        -p, --port <port>               port number
            --clone                     clone mode
            --force                     force update pkg file
            --test                      upload for test
            --gen                       generate snapshot
        -h, --help                      display help
        -v, --version                   display version
```

Create Package Server

You can create package server and server's distribution using "create" command.

```
## pkg-svr create -n <server name> -d <distribution> [-u <remote server url>] [-l <location>]
## -n option : New package server's name
## -d option : New distribution's name
## -u option : Parent package server distribution's URL
## -l option : Set the package server's location
```

- -u option
  - If you want to create package server refer the existing package server, this option will be helpful.
  - this option is the URL of package server/distribution, not the package server IP address.
- -l option
  - You can create new package server at this location.
  - If not specified, current directory will be used as server location

Add Distribution

You can add distribution to package server using "add-dist" command

```
## pkg-svr add-dist -n <server name> -d <distribution> [-u <remote_server_url>] [--clone]
## -n option : Package server's name
## -d option : New distribution's name
## -u option : Parent package server distribution's URL
## --clone option : Clone mode
```

- --clone option
    - This option will create new distribution from parent URL that is specified with -u option.
    - But the new distribution does not have the parent-child relation with parent package server/distribution

Add OS

You can add supported OS in package server.

```
## pkg-svr add-os -n <server name> -d <distribution> -o <os>
## -n option : Package server's name
## -d option : Distribution's name
```

Synchronization

You can synchronize package server using "sync" command. This "Synchronization" will download latest packages from parent package server

```
## pkg-svr sync -n <server name> -d <distribution> [--force]
## -n option : Your package server name
## -d option : Your distribution name
## --force option : Synchronize all package include local registered package
```

- There are some rules for update
    - Update will happen when parent server has newer version of package
    - If the packages are locally updated or uploaded, then they will be updated.
    - But if --force option is used, every packages will be updated with parent server/distribution forcibly.

Register Package(s)

You can register a package in package server using "register" command.

```
## pkg-svr register -n <server name> -d <distribution> -P <package file list> [--gen] [--test]
## -n option : Your package server name
## -d option : Your distribution name
## -P option : Package location list
## --gen option : Snapshot is generated automatically
## --test option : Test mode flag.
```

- You can also register multiple packages at the same time, by specifying package files with "," separator.
- --test option
  - The package will be uploaded into temp directory of package server, so original package will not be changed.
  - This will be used for testing your package without changing package server
- There are two types of packages: Binary packages and Source-Archive packages.
  - Binary package includes "pkginfo.manifest" which describes package version and dependency information. This package file has ".zip" extension.
  - Source-Archive packages are the package of source code that are used directly without modification . This package file has ".tar.gz" extension.

Generate Snapshot

You can generate snapshot using "gen-snapshot" command.

```
## pkg-svr gen-snapshot -n <server name> -d <distribution> -s <snapshot name> [-b <base snapshot name>]
## -n option : Your package server name
## -d option : Your distribution name
## -s option : Snapshot name
## -b option : Base snapshot name
```

- -b option
  - If you want to generate a snapshot from some specific snapshot, this option can be used.

Start Package Server

You can start package server using "start" command.

```
## pkg-svr start -n <server name> -p <port> [-w <password>]
## -n option : Your package server name
## -p option : Port number
## -w option : Password
```

- If you want to register package from remote server, then you must start package server.
  - If you are using build server then package server must started.

Stop Package Server

You can stop package server using "stop" command

```
## pkg-svr stop -n <server name> -p <port> [-w <password>]
## -n option : Your package server name
## -p option : Port number
## -w option : Password
```

Clean Package Server

Use “clean” command to delete the unnecessary packages from the package server.

```
## pkg-svr clean -n <server name> -d <distribution> [-s <snapshot list>]
## -n option : Your package server name
## -d option : Your distribution name
## -s option : Snapshot list which want to remain snapshot
```

- When a new package is registered old package information is deleted but package file remains.
    - clean command deletes old package files and snapshots.

Remove Package Server

If you want to delete package server, use "remove" command

```
## pkg-svr remove -n <server name>
## -n option : Your package server name
```

- If you use this command, then following message is shown.

```
Do you want to really? then input "YES"
```

- If you input "YES" in console, then package server will be deleted.
- But beware that this will delete all package server information and the deleted server cannot be recovered.

Remove Distribution

If you want to delete distribution in package server, use "remove-dist" command

```
## pkg-svr remove-dist -n <server name> -d <distribution>
## -n option : Your package server name
## -d option : Your distribution name
```

- If you use "remove-dist" command, then above message will be shown.

```
Do you want to really? then input "YES"
```

  - If you input "YES" in console, then distribution is deleted. If you input other command then remove-dist command is cancelled.
  - After Distribution is deleted then below message is shown.

```
Remove server!
```

- Remove-dist command will delete all distribution information.
- Distribution does not recovered.

Remove Package(s)

If you want to delete package in package server, using "remove-pkg" command

```
## pkg-svr remove-pkg -n <server name> -d <distribution> -P <package name list> [-o <os>]
## -n option : Your package server name
## -d option : Your distribution name
## -P option : Package name list which you want to delete
## -o option : Package's os list
```

- -o option
  - If you want to delete some specific OS, use this option
  - Multiple OS can also be specified with "," separator
  - Or, you can specify "all" in this option. This makes to delete the packages with all OS (ubuntu-32, windows-32, ...)

Remove Snapshot

If you want to delete snapshot in package server, use "remove-snapshot" command

```
## pkg-svr remove-snapshot -n <server name> -d <distribution> -s <snapshot list>
## -n option : Your package server name
## -d option : Your distribution name
## -s option : Snapshot name list which you want to delete
```

List Up Packages with Information

If you want to see package server information, use "list" command. This command will show all the packages in your PC or all distribution in package server

```
## pkg-svr list [-n <server name>]
## -n option : Your package server name
```

- If you want to see package server list then does not using -n option
- If you want to see distribution in package server then using -n <package server name> option

pkg-cli: Package Client

- DIBS provides package client tool for installing package and testing.

```
Request service to package-server and control packages service command-line tool.

Usage: pkg-cli <SUBCOMMAND> [OPTS] or pkg-cli (-h|-v)

Subcommands:
          update        Update to the latest package in your SDK environment.
          clean         Delete the package in your SDK environment.
          download      Download the package in your SDK environment.
          install       Download the package from package-server and install the package in your SDK
environment.
          install-file  Install the package in your SDK environment.
          uninstall     Uninstall the package in your SDK environment.
          upgrade       Upgrade your SDK environment.
          check-upgrade Check packages to upgrade.
          show-rpkg     Show the package in the package-server.
          list-rpkg     Show the all packages in the package-server.
          show-lpkg     Show the package in your SDK environment.
          list-lpkg     Show the all packages in your SDK environment.
          build-dep     Show build-dependency packages
          install-dep   Show install-dependency packages

Subcommand usage:
          pkg-cli update [-u <remote server url>]
          pkg-cli clean [-l <location>] [--force]
          pkg-cli download -P <package name> [-o <os>] [-l <location>] [-u <package server url>] [--trace]
          pkg-cli install -P <package name> [-o <os>] [-l <location>] [-u <package server url>] [--trace] [--
force]
          pkg-cli install-file -P <package file> [-l <location>] [-u <package server url>] [--trace] [--force]
          pkg-cli uninstall -P <package name> [-l <location>] [--trace]
          pkg-cli upgrade [-l <location>] [-o <os>] [-u <package server url>] [--trace]
```

```
           pkg-cli check-upgrade [-l <location>] [-o <os>] [-u <package server url>]
           pkg-cli show-rpkg -P <package name> [-o <os>] [-u <package server url>]
           pkg-cli list-rpkg [-o <os>] [-u <package server url>]
           pkg-cli show-lpkg -P <package name> [-l <location>]
           pkg-cli list-lpkg [-l <location>]
           pkg-cli build-dep -P <package name> [-o <os>]
           pkg-cli install-dep -P <package name> [-o <os>]

Options:
        -P, --pkg <package name/file>    package name or package file name
        -o, --os <operating system>      target operating system: ubuntu-32/ubuntu-64/windows-32/windows-
64/macos-64
        -u, --url <server url>           package server url: http://127.0.0.1/dibs/unstable
        -l, --loc <location>             install/download location
            --trace                      enable trace dependent packages
            --force                      enable force
        -h, --help                       display help
        -v, --version                    display version
```

Update Package List

You should have package list of server in your host before listing, installing and downloading packages. So, if you want to install the latest package, then you should update your package list before installing.

```
## pkg-cli update [-u <package server url>]
## -u option : Package server URL which contains binary and development packages.
```

- -u option
  - If this option is omitted, previous server URL will be used.

- Step

1. Update package list from server
   - $> pkg-cli update -u http://<package_server_address>/<package_server_name>/<distribution_name>
2. Check package list
   - $> pkg-cli list-rpkg

Upgrade Installed Packages

You can upgrade your installed packages from server.

```
## pkg-cli upgrade [-l <location>] [-o <os>] [-u <package server url>] [--trace]
## -l option  : Install root location of target SDK.
##       If omitted, current working directory path will be set
## -o option: target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
##       If ommited, it will host os.
## -u option: Package server URL which contains binary and development packages.
##       If ommited, it will use previous server URL.
##       ex) http://172.21.17.55/dibs/unstable
## --trace option: Install the package with all dependent packages
```

- Step

1. Check package for upgrading
   - $> pkg-cli update -u http://<package_server_address>/<package_server_name>/<distribution_name>
   - $> pkg-cli check-upgrade -l ~/tizen-sdk
2. Upgrade packages
   - $> pkg-cli upgrade -l ~/tizen-sdk

- You can also upgrade packages with updating as follows
  - $> pkg-cli upgrade -l ~/tizen-sdk -u http://<package_server_address>/<package_server_name>/<distribution_name>

Install Package

Installing a SDK package is also very simple. Here is the command for installing package files

```
## pkg-cli install -P <package name> [-o <os>] [-u <package server url>] [-l <location>] [--trace] [--force]
## -P option: Binary package name which you want to install
## -o option : target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
## -u option : Package server URL which contains binary and development packages.
##       If ommited, it will use previous server URL.
##       ex) http://172.21.17.55/dibs/unstable
## -l option : Install root location of target SDK
##        If omitted, current working directory path will be set
## --trace option : Install the package with all dependent packages
## --force option : Install the package by force
##          This option will allow installing the package that has lower or equal version compare to installed
```

- Step

1. Update package list from server
   - $> pkg-cli update -u http://<package_server_address>/<package_server_name>/<distribution_name>
2. Install "nativecommon-eplugin" package with dependent packages to new location("~/tizen-sdk2")

    - $> pkg-cli install -P nativecommon-eplugin -l ~/tizen-sdk2 --trace
3. Check local installed package list
    - $> pkg-cli list-lpkg -l ~/tizen-sdk2

Install Local Package File

You can also install package as package .zip file

```
## pkg-cli install-file -P <package file> [-l <location>] [-u <package server url] [--trace] [--force]
## -P option : Binary package file path (.zip)
## -l option : Install root location of target SDK
##      If omitted, current working directory path will be set
## -u option : Package server URL which contains binary and development packages.
##       If ommited, it will use previous server URL.
##       ex) http://172.21.17.55/dibs/unstable
## --trace option : Install the package with all dependent packages
## --force option : Install the package by force
##           This option will allow installing the package that has lower or equal version compare to installed
```

- Step

1. Install "nativecommon-eplugin" by network to new location("~/tizen-sdk2")
    - $> pkg-cli install-file -P nativecommon-eplugin_1.0.5_linux.zip -l ~/tizen-sdk2 --trace
2. Check local package list
    - $> pkg-cli list-lpkg -l ~/tizen-sdk2

Uninstall Package

You can uninstall package from local installed location. But, files generated after installing will be not removed

```
## pkg-cli uninstall -P <package file> [-l <location>] [--trace]
## -P option : Binary package name which you want to uninstall
## -l option : Installed package location
##      If omitted, current working directory path will be set
## --trace option : Uninstall the package with all dependent packages
```

- Step

1. Check local package list
    - $> pkg-cli list-lpkg -l ~/tizen-sdk2
2. Uninstall "nativecommon-eplugin" with dependent packages from the location ("~/tizen-sdk2")

        - $> pkg-cli uninstall -P nativecommon-eplugin -l ~/tizen-sdk2 --trace

Download Package File

You can also install package as package .zip file

```
## pkg-cli download -P <package name> [-o <os>] [-l <location>] [-u <package server url>] [--trace]
## -P option : Binary package name which you want to download
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
## -l option : Install root location of target SDK
##      If omitted, current working directory path will be set
## -u option : Package server URL which contains binary and development packages.
##      If ommited, it will use previous server URL.
##      ex) http://172.21.17.55/dibs/unstable
## --trace option : Install the package with all dependent packages
```

- Step

1. Update package list from server
    - $> pkg-cli update -u http://<package_server_address>/<package_server_name>/<distribution_name>
2. Download "nativecommon-eplugin" package with dependent packages
    - $> pkg-cli download -P nativecommon-eplugin -l ~/Downloads --trace

Installed package information

Check Upgrade

You can check for available package upgrades.

```
## pkg-cli check-upgrade [-l <location>] [-o <os>] [-u <package server url>]
## -l option : Install root location of target SDK
##      If omitted, current working directory path will be set
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
## -u option : Package server URL which contains binary and development packages.
##      If ommited, it will use previous server URL.
##      ex) http://172.21.17.55/dibs/unstable
```

Remote server package list

You can query package list in Remote package server.

```
## pkg-cli list-rpkg [-o <os>] [-u <package server url>]
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
## -u option : Package server URL which contains binary and development packages.
##      If ommited, it will use previous server URL.
##      ex) http://172.21.17.55/dibs/unstable
```

Remote server package information

You can query package information in package server.

```
## pkg-cli show-rpkg -P <package name> [-o <os>] [-u <package server url>]
## -P option : Binary package name
## -o option : Target operating system: ubutnu-32/ubuntu-64/windows-32/windows-64/macos-64
## -u option : Package server URL which contains binary and development packages.
##      If ommited, it will use previous server URL.
##      ex) http://172.21.17.55/dibs/unstable
```

Local package list

You can query installed package list in your local machine.

```
## pkg-cli list-lpkg [-l <location>]
## -l option : Install root location of target SDK
```

Remote server package information

You can query installed package information in your local machine.

```
## pkg-cli show-lpkg -P <package name> [-l <location>]
## -P option : Binary package name
## -l option : Install root location of target SDK
```

Build dependency package list

You can query package list about build-dependency.

```
## pkg-cli build-dep -P <package name> [-o <os>]
## -P option : Binary package name
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
```

Install dependency package list

You can query package list about install-dependency.

```
## pkg-cli install-dep -P <package name> [-o <os>]
## -P option : Binary package name
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
```

build-svr: Build Server

- You can control build server using "build-svr" command.

```
Build-server administer service command-line tool.

Usage: build-svr <SUBCOMMAND> [OPTS] or build-svr (-h|-v)

Subcommands:
          create       Create the build-server.
          remove       Remove the build-server.
          start        Start the build-server.
          stop         Stop the build-server.
          add-svr      Add build-server for support multi-OS or distribute build job.
          add-prj      Register information for project what you want build berfore building a project.
          register     Register the package to the build-server.
          fullbuild    Build all your projects and upload them to package server.

Subcommand usage:
          build-svr create -n <server name> -u <package server url> -d <package server address> -t <ftp server
url>
          build-svr remove -n <server name>
          build-svr start -n <server name> -p <port>
          build-svr stop -n <server name>
          build-svr upgrade –n <server name>
          build-svr add-svr -n <server name> (-d <friend server address>|-u <remote pkg server url>) [--proxy
<proxy url>]
          build-svr add-prj -n <server name> -N <project name> (-g <git repository> -b <git branch>|-P <package
name>) [-w <password>] [-o <os list>]
          build-svr add-os -n <server name> -o <os>
          build-svr register -n <server name> -P <package file>
          build-svr fullbuild -n <server name>

Options:
       -n, --name <server name>        build server name
       -u, --url <package server url>  package server url: http://127.0.0.1/dibs/unstable
           --proxy <proxy url>         proxy url : http://172.21.111.100:2222
       -d, --address <server address>  server address: 127.0.0.1:2224
       -p, --port <port>               server port number: 2224
       -P, --pkg <package name/file>   package file path or name
       -o, --os <target os list>       ex) ubuntu-32,windows-32
       -N, --pname <project name>      project name
       -g, --git <git repository>      git repository
       -b, --branch <git branch>       git branch
       -w, --passwd <password>         password for managing project
       -t, --ftp <ftp server url>      ftp server url: ftp://dibsftp:dibsftp@127.0.0.1:1024
       -h, --help                      display this information
       -v, --version                   display version
```

Create Build Server

You can create build server using "build-svr" command. Here is the command for creating build server.

```
## build-svr create -n <server name> -u <package server url> -d <package server address> -t <ftp server url>
## -n option : Your build server name
## -u option : Package server url (ex: http://127.0.0.1/testserver/unstable)
## -d option : Package server address (ex: 127.0.0.1:3333)
## -t option : FTP server url (ex: ftp://dibsftp:dibsftp@127.0.0.1)
```

- Step

1. Create a directory and move to it
   - $> mkdir buildsvr; cd buildsvr
2. Create a build server
   - $> build-svr create -n buildsvr -u http://<package_server_address>/<package_server_name>/<distribution_name> -d <build_server_ip>:<build_server_port> -t ftp://<ftp_id>:<ftp_pw>@<package_server_ip>
3. You can check a config file for server
   - $> cat ~/.build_tools/build_server/buildsvr/server.cfg

Remove Build Server

You can remove build server using "build-svr" command. Here is the command for removing build server.

```
## build-svr remove -n <server name>
## -n option : Your build server name
```

Add New Project

You should register information for a project that you want to build before building.. Here is the command for registering.

```
## build-svr add-prj -n <server name> -N <project name> (-g <git repository> -b <git branch>|-P <package name>)
[-w <password>] [-o <os list>]
## -n option : Your build server name
## -N option : Project name
## -w option : Password for managing project. If you set password, it is used when building project
## -g option : Git repository for project (ex: gerrithost:/sdk/tools/sdk-build)
## -b option : Git branch (ex: develop)
## -P option : Prefix of package name
## -o option : Support os list
```

Project has two types, normal project and binary project.

- Normal project builds from build server and uploads to package server
    - normal project uses -g and -b option to clone source using git.
- binary project just uploads package file to build server then build server uploads to package server
    - binary project uses -P option that is needed in package naming.
    - binary package is named <prefix of package name>_<package version>.tar.gz

- Step

1. Register information for project
    - $> build-svr add-prj -n buildsvr -N dibs -g gerrithost:/sdk/tools/sdk-build -b develop -o ubuntu-32,windows-32
    - $> build-svr add-prj -n buildsvr -N dibs -P llvm -o ubuntu-32,windows-32

Start Build Server

- You can start build server using following command

```
## build-svr start -n <build-server-name> -p <port>
## -n option : Your build server name
## -p option : Server port number
```

- Step

1. Start build server
    - $> build-svr start -n buildsvr -p 2224

Stop Build Server

You can stop build server using following command.

```
## build-svr stop -n <server name>
## -n option : Your build server name
```

- Step

1. Stop build server
    - $> build-svr stop -n buildsvr

Add Build Server

You can add build server for supported multi-OS or distribute build job.

```
## build-svr add-svr -n <server name> -d <friend server address> [--proxy <proxy url>]
## -n option : Your build server name
## -d option : Add build server address (ex: 127.0.0.1:3333)
## --proxy option : proxy server url
```

Register package

You can register packages like binary package to build server. And upload the package to package server.

```
## build-svr register -n <server name> -P <package file> [--proxy <proxy url>]
## -n option : Your build server name
## -P option : Prefix of package name
## --proxy option : proxy server url
```

Full build

You can build full packages and upload to package server.

```
## build-svr fullbuild -n <server name>
## -n option : Your build server name
```

Server upgrade

You can upgrade build server it self, using new version of dibs in package server

```
## build-svr upgrade -n <server name>
## -n option : Your build server name
```

build-cli: Build Client

- You can build project using "build-cli" command.

```
Requiest service to build-server command-line tool.

Usage: build-cli <SUBCOMMAND> [OPTS] or build-cli (-h|-v)

Subcommands:
          build        Build and create package.
          resolve      Request change to resolve-status for build-conflict.
          query        Query information about build-server.
          query-system  Query system information about build-server.
          query-project Query project information about build-server.
          query-job     Query job information about build-server.
          cancel        Cancel a building project.
          register      Register the package to the build-server.

Subcommand usage:
          build-cli build -N <project name> -d <server address> [-o <os>] [-w <password>] [--async]
          build-cli resolve -N <project name> -d <server address> [-o <os>] [-w <password>] [--async]
          build-cli query -d <server address>
          build-cli query-system -d <server address>
          build-cli query-project -d <server address>
          build-cli query-job -d <server address>
          build-cli cancel -j <job number> -d <server address> [-w <password>]
          build-cli register -P <file name> -d <server address> -t <ftp server url> [-w <password>]

Options:
       -N, --project <project name>     project name
       -d, --address <server address>   build server address: 127.0.0.1:2224
       -o, --os <operating system>      target operating system: ubuntu-32/ubuntu-64/windows-32/windows-
64/macos-64
           --async                      asynchronous job
       -j, --job <job number>           job number
       -w, --passwd <password>          password for managing project
       -P, --pkg <package file>         package file path
       -t, --ftp <ftp server url>       ftp server url: ftp://dibsftp:dibsftp@127.0.0.1
       -h, --help                       display help
       -v, --version                    display version
```

Build/Upload Project

- You can request to build project to build server. After that, package will be uploaded to package server
- The project name should be set before through "build-svr add-prj" command

```
## build-cli build -N <project name> -d <server address> [-o <os>] [-w <password>] [--async]'
## -N option : Project name. This should be set before through "build-svr add-prj" command
## -d option : Build server address: 127.0.0.1:2224
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
## -w option : Password for managing project. If a password is set before through "build-svr add-prj" command,
you should input the password
## --async option : asynchronous job
```

- Step

1. Request to build project to build server
   - ex) $> build-cli build -N dibs -d <build_server_ip>:<build_server_port>

Resolve Build-Dependency Conflict

You can request to resolve the build-conflict of a project. These conflicts happen when other projects use the APIs of a project which are changed in newer version. These can be detected by checking "Checking reverse-build dependency fail" message of your build log when you build your project. For example,

```
...
Info: Checking reverse build dependency ...
Info:  * Checking reverse-build ... testc
Error: Job is stopped by ERROR
```

In this case, you can request to "resolve" instead of "build". The process of resolving is similar to building except that it will be pending when there are conflicts. Then you can request new "build" job to resolve the pending project.

```
## build-cli resolve -N <project name> -d <server address> [-o <os>] [-w <password>] [--async]
## -N option : Project name : This should be set before through "build-svr add-prj" command
## -d option : Build server address: 127.0.0.1:2224
## -o option : Target operating system: ubuntu-32/ubuntu-64/windows-32/windows-64/macos-64
## -w option : Password for managing project. If a password is set before through "build-svr add-prj" command,
you should input the password
## --async option : asynchronous job
```

For example, think about this situation

- Project A depends on Project B.
- And the packages of Project A, B are already built and uploaded.
- In this situation, let's assume that the APIs of Project B which are used by Project A must be changed.
- But the update of A to A' will invoke the failure of checking reverse-build-dependency.

1. Request to resolve project B'
    - ex) **$> build-cli resolve -N B -d <build_server_ip>:<build_server_port>**
2. Then you can see that your job enters PENDING state.
3. Update Project A to Project A' according the APIs of Project B'
4. Create a new terminal and request to build Project A'
    - ex) **$> build-cli build -N A -d <build_server_ip>:<build_server_port>**
5. If A' can solve the conflict of B', then A' will enter WORKING state again.
6. These two separate process will be done and upload their packages to package server.

Query Build Server

- You can query information for build server.

```
## build-cli query -d <server address>
## -d option : build server address which you want to query
```

- Step

1. Query information for build server
   - $> build-cli query -d <build_server_ip>:<build_server_port>

Cancel Build Job

- You can cancel a building project

```
## build-cli cancel -j <job number> -d <server address> [-w <password>]
## -j option : Build job number
## -d option : Build server address: 127.0.0.1:2224
## -w option : Password for managing project. If a password is set before through "build-svr add-prj" command,
you should input the password
```

- Step

1. Cancel build project
   - $> build-cli cancel -j 10 -d <build_server_ip>:<build_server_port>

Register package

- You can upload binary package file
  - Binary package is not build in build server. That is just uploaded to package server using build server

```
## build-cli register -P <file name> -d <server address> -t <ftp server url> [-w <password>]
## -P option : Package file path. It must binary package not archive package
## -d option : Build server address: 127.0.0.1:2224
## -t option : Ftp server url: ftp://dibsftp:dibsftp@127.0.0.1
## -w option : Password for managing project. If a password is set before through "build-svr add-prj" command,
you should input the password
```

Package file name must besame as already registered in server project. And build-cli only register in binary package.

- Step

1. Register package
   - $> build-svr register -n buildsvr01 -P mingw32-msys-1.0_0.21.3_windows.zip -d 127.0.0.1:2224 -t ftp://<ftp_id>:<ftp_pw)@<ftp_address> -w <project_passwd>

pkg-build: Local build client

- You can do build & packaging in your local PC

```
Build and packaging service command-line tool.

Usage: pkg-build -u <package server url> [-o <os>] [-c] [-h] [-v]

Options:
    -u, --url <package server url>   remote package server url: http://127.0.0.1/dibs/unstable
    -o, --os <os>                    operating system
    -c, --clean                      clean build
    -h, --help                       display help
    -v, --version                    display version
```

Local package build

- You want to build source to packaging file, use this command
- DIBS downloads depends other package automatically using package server in -u option's address
- If you want to packaging for different OS in host PC then using -o option
- If you want to remove already installed and downloaded files then using -c option

- Step
  1. Local packaging
     - pkg-build -u http://<package_server_address>/<package_server_name>/<distribution_name> -o ubuntu-32 -c
  2. Now you can see the package files(*.zip, *.tar.gz) in your source directory

DIBS package information

- DIBS's package has version information in pkginfo.manifest file
- Versions are denoted using a standard triples of integers : Major.Minor.Patch

- If you want remote build using build server then you must increase version.