Overview

The document covers local environment setup for following projects:

- APDInterface [lib]
- APDInterfaceEMV [lib]
- APDInterfaceMoneris [lib]
- ActiveNetPackage [web]
- ActiveNetServlet [lib]
- EntryPoint [lib]
- EntryPointApplet [lib]
- PassProduction [lib]
- PointOfSale [lib]

[lib] indicates the project is not a runnable project but generates jar files to be referred by other projects.

[web] indicates the project is runnable.

Preparation

1. svn checkout buildtools at: https://fndsvn.dev.activenetwork.com/foundations/core/buildtools/trunk, the folder name has to be exact **buildtools.**
2. svn checkout projects mentioned in overview into same directory, or alternatively, checkout buildtools into existing projects dir. The structure is like:

```
├── AN
|   ├── APDInterface
|   ├── APDInterfaceEMV
|   ├── APDInterfaceMoneris
|   ├── ActiveNetClients
|   ├── ActiveNetPackage
|   ├── ActiveNetServlet
|   ├── EntryPoint
|   ├── EntryPointApplet
|   ├── PassProduction
|   ├── PointOfSale
|   ├── buildtools
|   └── cert
```

Note: cert/ is needed to sign jars locally, Third\ Party\ Stuff is no longer needed to build these projects.

Build from command line

Preparation[windows only]

The preparation is needed on windows only, for OS X or linux, jump to next section. Download and install cygwin at: http://cygwin.com/install.html, do remember to add cygwin bin/ dir to system path, with cygwin installed, unix commands are available in windows command line, e.g try *ls* to verify the installation.

Build a project

1. open up command line, change directory to project you want to build, e.g `cd ActiveNetServlet`
2. commands on windows all start with `ant`, it starts with `./ant.sh` on OS X or linux.
    1. to build on windows: `ant fullclean main debug`
        - `fullclean`: is to do a full clean up for previous build
        - `main`: is to compile source code
        - `debug`: is to start the application in debug mode after compilation, note the command applies to web server project only, for AN it's ActiveNetServlet project
    2. to build on OS X: `./ant.sh fullclean main debug`, run `chmod +x *.sh` if it gives permission denied error
3. More on `main`: it compiles the project and also package and copy the executable to a specific location, on windows it's C:\active{project}, on OS X it's ~/active/{project}, it is also same runtime structure on other environments include Production.
4. web server can be reached via: http://localhost:8080/linux01/servlet/processAdminLogin.sdi, *linux01* is site name can be configured in config/service.properties: `moreJavaOptions=-Dsite.name=/linux01/`, the url structures are same cross environments.

Available commands

- `fullclean`: remove executable structure from previous build
- `main`: compilation
- `debug`: startup web application in debug mode, default debug port is 5005
- `run`: startup web application in non-debug mode
- `test`: run unit test
- `coverage`: run test coverage, coverage report will be generated when there's no unit test failure
- `dependencies`: generate dependencies report
- `pmd`: run PMD check
- `findbugs`: run findbugs check
- `install`: for lib project only, install api jar to local repository, on windows it's C:\ivy2, on OS X it's ~/.ivy2
- `checkstyle`: run checkstyle, which uses an embedded style rule in buildtools

All above commands can be combined into a single one like: `ant fullclean main test` `coverage pmd checkstyle findbugs debug`

Run executable

Other than build and run a project at same time, the executable structure allows to run separately as well.

runtime structure:

```
├── ActiveNetServlet
|    ├── config
|    |    ├── btrace
|    |    ├── lib
|    |    └── xsd
|    ├── jetty
|    |    ├── servlet
|    |    ├── webapps
|    |    └── work
|    ├── lib
|    ├── logs
|    └── ui
```

This is also the exact structure gets deployed onto environments include Production, the web application can be started by start_service.sh/bat in config.

- config: all kinds of configuration files include start-stop batch/shell scripts, spring configuration files, sdi.ini
- jetty: web files, includes html, jsp, static resource files
- lib: dependency jars
- logs: log files include application log, gc log, tracer log
- ui: used to be empty

Setup Eclipse

IvyDE[required]

Install from update site

1. Follow instruction on https://ant.apache.org/ivy/ivyde/download.html to install plugin via update site.

Install manually

1. download ivyDE.zip from https://fndsvn.dev.activenetwork.com/foundations/infrastructure/ivyde-eclipse-plugin/ivyde.zip, unzip it.

2. For windows, go to eclipse install path, copy features/ and plugins/ to same folders.
3. For OS X, goto applications, find eclipse.app, right click -> show content -> Contents -> Eclipse, merge features/ and plugins/ to same folders

Configuration

1. Eclipse File -> Import -> Existing Projects into Workspace -> Next -> Browse to find buildtools -> Finish to import buildtools
2. Eclipse -> Preference -> Ivy -> Settings -> Workspace -> find buildtools/ivy/ivysettings.xml -> OK
3. cd to project dir to be imported, run `g eclipse` to generate eclipse .project and .classpath file if the project was never imported into eclipse
4. Eclipse -> File -> Import -> Existing Project into Workspace -> Next -> Browser to find projects to be imported -> Finish
5. Right click on imported project -> Build path -> Configure build path -> Java build path -> Libraries, remove all libs except JRE System Library; Add Library -> IvyDE Managed Dependencies -> Next -> Browser ivy.xml in project root -> checked runtime and test if they're not -> Finish -> OK; this is to switch to use IvyDE to manage dependencies in IDE
6. To resolve dependencies in eclipse, right click on the project -> Ivy -> Resolve, recommend to resolve from command line first before
7. Repeat above steps to import all ANET java projects

EclEmma

1. download latest plugin at http://eclemma.org/download.html, unzip it and merge features/ plugins/ into eclipse installation directory, then restart to take effect
2. to run with coverage, select a junit java in Project explorer, right click -> Coverage As -> Unit test

FindBugs

1. download latest plugin at http://sourceforge.net/projects/findbugs/files/findbugs%20eclipse%20plugin/, unzip and copy the entire folder into eclipse plugins/, then restart to take effect
2. to scan code with findbugs, right click on a project -> Find Bugs -> Find Bugs

PMD

1. Eclipse Help -> Eclipse Marketplace -> Find: pmd to find pmd -> Install

Daily development

Code changes in ActiveNetServlet

1. `ant fullclean main debug` to startup ActiveNetServlet locally with default debug port 5005
2. commit code changes to SVN after testing locally
3. execute build job in http://anjenkinsm.dev.activenetwork.com:8080/view/Linux-View/
    1. 001-BUILD-AUI-1540: build 15.40
    2. 001-BUILD-AUI-1600: build 16.0
    3. 001-BUILD-AUI-NFS-TRUNK: build trunk

Code changes in lib projects[take ActiveNetPackage as an example]

1. make sure the changes is unit tested
2. `cd ActiveNetPackage`
3. `ant fullclean main install`: this would build ActiveNetPackage into a versioned jar, the version # is displayed in console after `install`, which can also read from build-ivy.xml in project root, say the version # is 16.1.0.1-SNAPSHOT
4. open ActiveNetServlet/ivy.xml, find ActiveNetPackage, update the version to be above
5. `cd ActiveNetServlet`, `ant fullclean main debug` to test Servlet with latest ActiveNetPackage
6. when changes is tested locally, commit ActiveNetPackage changes to SVN
7. go to http://anjenkinsm.dev.activenetwork.com:8080/view/Linux-View/job/Release-Lib/ -> Build with Parameters, select APP, fill in SVNPATH, note you don't have to specify lib project in SVNPATH, e.g APP = ActiveNetPackage, SVNPATH = https://fndsvn.dev.activenetwork.com/ActiveNet/trunk/ instead of ~~https://fndsvn.dev.activenetwork.com/ActiveNet/trunk/ActiveNetPackage~~ is to release trunk ActiveNetPackage
8. release email will be sent with a formal version #, update the version # into ActiveNetServlet/ivy.xml to let servlet use latest ActiveNetPackage, then retest and commit ivy.xml into SVN and release ActiveNetServlet