Building the Code from Source Distributions or CVS

Some Important Points

Bouncy Castle is used across a wide range of VMs from a number of different vendors so the distributions are as minimalist as possible. While this does remove the clutter, it does mean that creating your own build will require a bit of thought. If you are using specific tools to build Java applications, or the Bouncy Castle APIs, and having trouble, your best first point of call is the user community involved with the tools you are using. Development environments tend to be very individual, so while the fact we issue signed jars does mean we can build from the source code, it does not mean that any of the techniques the Bouncy Castle developers use will work for you.

Finally, before you go any further, keep these two points in mind as well:

1. If you are build for the J2ME you must use an obfuscator to deal with the java.* package names the BC APIs include.
2. If you are trying to build a provider which works with the Sun JCE you need to have a Sun signing certificate to sign the provider jar file with.

Building From CVS or the Master Distribution

The naming of the master distributions follows the pattern crypto-*.zip or crypto-*.tar.gz. In both the master distribution and CVS the src and test/src directory are for JDK 1.5.

Compatibility classes for other VMs are as follows and can be found both under src/main and src/test for the different distributions:

1. JDK 1.4 - jdk1.4
2. JDK 1.3 - jdk1.3
3. JDK 1.2 - jdk1.2
4. JDK 1.1 - jdk1.1
5. JDK 1.0 - jdk1.0
6. J2ME - j2me

The clean room JCE, which will compile with everything from JDK 1.1 and up is in the jce/src directory.

The BZIP2 classes, which are modified versions of the Apache ANT files and covered by the Ant Apache License, can be found in bzip2/src.

The build scripts that come with the full distribution allow creation of the different releases by using the trees under core, prov, pkix, and pg excluding classes that are not appropriate and copying in the required compatibility classes from the directories containing compatibility classes appropriate for the distribution. Building is done using a mixture of ant and Unix shell scripts - in the case of the builds for JDK 1.3, JDK 1.4, and JDK 1.5 the shell scripts are only used to provide an environment for ant to run in - for example to build the full JDK 1.5 distribution (provider plus libraries):

```
sh build15+
```

will do the right thing, providing you have JAVA_HOME correctly set you have set bc-build.properties so that JavaMail, JAF, and JUnit are on your classpath.

If you want to try create a build for yourself, using your own environment, the best way to do it is to start with the build for the distribution you are interested in, make sure that builds, and then modify your build scripts to do the required exclusions and file copies for your setup, otherwise you are likely to get class not found exceptions. The final two caveats to this are that as the J2ME distribution includes some compatibility classes starting in the java package, you need to use an obfuscator to change the package names before attempting to import a midlet using the BC API, and that if you are trying to build a provider for the Sun JCE you need a Sun signing certificate.

Building From a Specific Source Distribution

Specific source distributions again contain just the source code required for a specific JDK and the pregenerated JavaDoc. They all end in .tar.gz or .zip. These just contain the source files in a src.zip file. You should just be able to load these into your favorite environment, the only caveats being that if you are dealing with JDK 1.3 or later you need to have JavaMail, JAF, and JUnit in your classpath, that if you are trying to build a provider for the Sun JCE you need a Sun signing certificate, and if you are using a J2ME build you need to use an obfuscator to change the java.* namespace packages that the BC distribution includes for compatibility purposes.