Using the JAR Utility

Prepared by Jeff Hunter, Sr. DBA *01-SEP-2002*

Overview

Sun Microsystems's implementation of Java (JDK 1.1 and later) includes a number of tools for Java developers. One of those tools is the Java Archive (JAR) utility. *jar* is a tool that is used to create and manipulate JAR files.

A JAR file is a ZIP file that contains Java classes, auxiliary resource files required by those classes, and optional meta-information. Included in the manifest file is meta-information that lists the contents of the JAR archive and provides auxiliary information about each file.

Use the jar command to create JAR files, list the contents of JAR files, and extract files from a JAR archive. With Java 1.2 and later, the jar command can also add files to an existing archive or update the manifest file of an archive. In Java 1.3 and later, it can also add an index entry to the JAR file.

Keep in mind that this document covers the details of Sun's implementation of the jar command. If you are using a jar command other than Sun's, you should consult your vendor's documentation.

Synopsis

```
jar c|t|u|x[f][m][M][0][v][jar][manifest][-C directory][files]
jar -i [jar]
```

Options

For anyone comfortable with the *tar* (tape archive) command under UNIX, you will notice that the syntax is nearly identical. The first letter of the first argument (of which there are four of them) specifies what action *jar* is to perform and is a required parameter. Here are the four possible action options:

Action Options

`c` Creates a new JAR archive. A list of input files and/or directories must be specified as the final arguments to jar. The newly create JAR file will have a `META-INF/MANIFEST.MF` file as its first entry. This file is automatically created and will list the contents of the JAR file and contain a message digest for each file.

`t` Lists the contents of the JAR file.

`u` Updates the contents of a JAR archive file. This option will add any files listed on the command line to the archive. When used with the `m` option, this adds the specified manifest information to the JAR file. This option is only available in Java 1.2 and later.

`x` Extracts the contents of the JAR file. Any files or directories listed on the command line are extract and created in the current working directory. If no files are listed names are specified, all the files and directories in the JAR archive are extracted.

Modifier Options

Each of the four action options (above) can be followed by additional letters that provide further detail about the operation being performed.

`f` Indicates that *jar* is to operate on a JAR file who name is specified on the command line. If the `f` option is present, the command line will require the JAR file to operate on. If this option is not specified, *jar* will read the JAR file from standard input and/or writes a JAR file to standard output.

`m` Whenever *jar* creates or updates a JAR archive, it automatically creates (or updates) a manifest file named `META-INF/MANIFEST.MF` within the JAR file. By default, the manifest option simply lists the contents of the JAR file. For JAR files that require additional information to be included in the manifest; the `m` option tells the jar command that a manifest template is specified on the command line. *jar* will read the included manifest file and store its content into the `META-INF/MANIFEST.MF` file it creates (or updates). Note that the `m` option should be used only with the `c` or `u` commands, not with the `t` or `x` commands.

`M` To be used with the `c` and `u` commands to tell *jar* NOT to create a default manifest file.

`v` Used to tell *jar* to product verbose output.

`0` Note that this option is the number zero and not the letter O. To be used with the `c` and `u` commands to tell jar to store files in the JAR archive WITHOUT compressing them.

Additional Options

`-C` *`dir`* Used with the list of files to process. It tells *jar* to change to the specified *`dir`* while processing the subsequent files and directories. The subsequent files and directories are interpreted relative to *`dir`* and are inserted into the JAR archive without *`dir`* as a prefix. Any number of `-C` options can be used; each remains in effect until the next is encountered. The directory specified by a `-C` option is interpreted relative to the current working directory, not the directory specified by the previous `-C` option. (Java 1.2 and later)

`-i` *`jarfile`* Introduced in Java 1.3, the `-i` option is used instead of the `c`, `t`, `u`, and `x` commands. It tells *jar* to produce an index of all JAR files referenced by the specified JAR *`jarfile`*. The index will be stored in a file called `META-INF/INDEX.LIST`. The Java interpreter (or applet viewer) can use the information in the index to optimize its class and resource "lookup" algorithm and to avoid downloading unnecessary JAR files.

Examples

The following is an array of examples that are most often used when working with JAR archives.

- To create a simple JAR file called *utilities.jar* that contains all the class files in the current (working) directory and all files in subdirectories *images* and *data*:

```
% jar cvf utilities.jar *.class images data
```

- To verbosely list the contents of a JAR file called *utilitie.jar*:

```
% jar tvf utilities.jar
```

- To extract only the manifest file from a JAR archive called *utilities.jar*:

```
% jar xvf utilities.jar META-INF/MANIFEST.MF
```

- To update the JAR file *utilities.jar* and add the image file *images/icon_button.gif*:

```
% jar uvf utilities.jar images/icon_button.gif
```

- To update the manifest file of a JAR file named *utilities.jar*:

  ```
  % jar uvfm utilities.jar manifest.template
  ```

- Temporarily change directories during execution of the jar command. For example,

  ```
  % jar cvf utilities.jar *.class -C images ico.gif –C data .
  ```

  would first add all `*.class` files in the current directory to the archive, then change to the *images* directory and add the `ico.gif` file while not including the *images* directory to the archive. The command would then change to the *data* directory and add all files there and again, like the *images* directory, it would not include the *data* directory in the archive.