Ant Core Task Quick Reference

ANT
```
<ant antfile="subproject/subbuild.xml" dir="subproject" target="compile"/>
```

ANT
```
<ant dir="subproject"/>
```

ANT
```
<ant antfile="subproject/property_based_subbuild.xml">
  <property name="param1" value="version 1.x"/>
  <property file="config/subproject/default.properties"/>
</ant>
```

ANT
```
<ant inheritAll="false" antfile="subproject/subbuild.xml">
  <property name="output.type" value="html"/>
</ant>
```

ANTCALL
```
<target name="default">
  <antcall target="doSomethingElse">
    <param name="param1" value="value"/>
  </antcall>
</target>
```

ANTCALL Will run the target 'doSomethingElse' and echo 'param1=value'.
```
<target name="doSomethingElse">
  <echo message="param1=${param1}"/>
</target>
```

ANTCALL will copy the parent's definition of path1 into the new project using the id path2.
```
<antcall ... >
  <reference refid="path1" torefid="path2"/>
</antcall>
```

ANTSTRUCTURE
```
<antstructure output="project.dtd"/>
```

APPLY
```
<apply executable="ls">
  <arg value="-l"/>
  <fileset dir="/tmp">
    <patternset>
      <exclude name="**/*.txt"/>
    </patternset>
  </fileset>
  <fileset refid="other.files"/>
</apply>
```

APPLY
```
<apply executable="somecommand" parallel="false">
  <arg value="arg1"/>
  <srcfile/>
  <arg value="arg2"/>
  <fileset dir="/tmp"/>
</apply>
```

AVAILABLE sets the Myclass.present property to the value "true" if the class org.whatever.Myclass is found in Ant's classpath.
```
<available classname="org.whatever.Myclass" property="Myclass.present"/>
```

AVAILABLE sets the jaxp.jar.present property to the value "true" if the file ./lib/jaxp11/jaxp.jar is found.
```
<property name="jaxp.jar" value="./lib/jaxp11/jaxp.jar"/>
<available file="${jaxp.jar}" property="jaxp.jar.present"/>
```

AVAILABLE
```
<available file="/usr/local/lib" type="dir" property="local.lib.present"/
```

AVAILABLE
```
<available property="have.extras" resource="extratasks.properties">
  <classpath>
    <pathelement location="/usr/local/ant/extra.jar/>
  </classpath>
</available>
```

BASENAME will set jar.filename to myjar.jar, if lib.jarfile is defined as either a full-path filename (eg., /usr/local/lib/myjar.jar), a relative-path filename (eg., lib/myjar.jar), or a simple filename (eg., myjar.jar).
```
<basename property="jar.filename" file="${lib.jarfile}"/>
```

BASENAME will set cmdname to foo.
```
<basename property="cmdname" file="D:/usr/local/foo.exe" suffix=".exe"/>
```

BASENAME will set temp.dirname to the last directory element of the path defined for the TEMP environment variable.
```
<property environment="env"/>
<basename property="temp.dirname" file="${env.TEMP}"/>
```

BUILDNUMBER Read, increment, and write a build number to the default file, build.number.
```
<buildnumber/>
```

BUILDNUMBER Read, increment, and write a build number to the file mybuild.number.
```
<buildnumber file="mybuild.number"/>
```

BUNZIP2 expands test.tar.gz to test.tar
```
<gunzip src="test.tar.gz"/>
```

BUNZIP2 expands test.tar.bz2 to test.tar
```
<bunzip2 src="test.tar.bz2"/>
```

BUNZIP2 expands test.tar.gz to test2.tar
```
<gunzip src="test.tar.gz" dest="test2.tar"/>
```

BUNZIP2 expands test.tar.gz to subdir/test.tar (assuming subdir is a directory).
```
   <gunzip src="test.tar.gz" dest="subdir"/>
```

BZIP2
```
   <gzip src="test.tar" zipfile="test.tar.gz"/>
```

BZIP2
```
   <bzip2 src="test.tar" zipfile="test.tar.bz2"/>
```

CHECKSUM Generates a MD5 checksum for foo.bar and stores the checksum in the destination file foo.bar.MD5. foo.bar.MD5 is overwritten only if foo.bar is newer than itself.
```
   <checksum file="foo.bar"/>
```

CHECKSUM Generates a MD5 checksum for foo.bar and stores the checksum in foo.bar.MD5. If foo.bar.MD5 already exists, it is overwritten.
```
   <checksum file="foo.bar" forceOverwrite="yes"/>
```

CHECKSUM Generates a MD5 checksum for foo.bar and stores it in the Project Property foobarMD5.
```
   <checksum file="foo.bar" property="foobarMD5"/>
```

CHECKSUM Generates a MD5 checksum for foo.bar, compares it against foo.bar.MD5 and sets isMD5ok to either true or false, depending upon the result.
```
   <checksum file="foo.bar" verifyProperty="isMD5ok"/>
```

CHECKSUM Generates a SHA checksum for foo.bar and stores the checksum in the destination file foo.bar.asc. foo.bar.asc is overwritten only if foo.bar is newer than itself.
```
   <checksum file="foo.bar" algorithm="SHA" fileext="asc"/>
```

CHECKSUM Generates a MD5 checksum for foo.bar, compares it against the value of the property md5, and sets isEqual to either true or false, depending upon the result.
```
   <checksum file="foo.bar" property="${md5}" verifyProperty="isEqual"/>
```

CHECKSUM Works just like Example 1, but generates a .MD5 file for every file that begins with the name foo.
```
   <checksum>
      <fileset dir=".">
         <include name="foo*"/>
      </fileset>
   </checksum>
```

CHECKSUM Works like Example 4, but only sets isChecksumEqual to true, if the checksum matches - it will never be set to false. This example demonstrates use with the Condition task.
```
   <condition property="isChecksumEqual">
      <checksum>
         <fileset dir=".">
            <include name="foo.bar"/>
         </fileset>
      </checksum>
   </condition>
```

CHMOD makes the "start.sh" file readable and executable for anyone on a UNIX system.
```
   <chmod file="${dist}/start.sh" perm="ugo+rx"/>
```

CHMOD makes all ".sh" files below ${dist}/bin readable and executable for anyone on a UNIX system.
```
   <chmod dir="${dist}/bin" perm="ugo+rx" includes="**/*.sh"/>
```

CHMOD makes all files below shared/sources1 (except those below any directory named trial) writable for members of the same group on a UNIX system. In addition all files belonging to a FileSet with id other.shared.sources get the same permissions.
```
   <chmod perm="g+w">
      <fileset dir="shared/sources1">
         <exclude name="**/trial/**"/>
      </fileset>
      <fileset refid="other.shared.sources"/>
   </chmod>
```

CONCAT Concatenate a string to a file:
```
   <concat destfile="README">Hello, World!</concat>
```

CONCAT Concatenate a series of files to the console:
```
   <concat>
      <fileset dir="messages" includes="*important*"/>
   </concat>
```

CONCAT Concatenate a single file, appending if the destination file exists:
```
   <concat destfile="NOTES" append="true">
      <filelist dir="notes" files="note.txt"/>
   </concat>
```

CONCAT Concatenate a series of files, overwriting if the destination file exists:
```
   <concat destfile="${docbook.dir}/all-sections.xml">
      <filelist dir="${docbook.dir}/sections" files="introduction.xml,overview.xml"/>
      <fileset dir="${docbook.dir}"  includes="sections/*.xml"
              excludes="introduction.xml,overview.xml"/>
  </concat>
```

CONDITION sets the property javamail.complete if both the JavaBeans Activation Framework and JavaMail are available in the classpath.
```
   <condition property="javamail.complete">
      <and>
         <available classname="javax.activation.DataHandler"/>
         <available classname="javax.mail.Transport"/>
      </and>
   </condition>
```

CONDITION sets the property isMacOsButNotMacOsX if the current operating system is MacOS, but not MacOS X - which Ant considers to be in the Unix family as well.
```
   <condition property="isMacOsButNotMacOsX">
      <and>
         <os family="mac"/>
         <not>
            <os family="unix"/>
         </not>
      </and>
   </condition>
```

CONDITION sets the property isSunOSonSparc if the current operating system is SunOS and if it is running on a sparc architecture.

```
<condition property="isSunOSonSparc">
   <os name="SunOS" arch="sparc"/>
</condition>
```

COPY a single file

```
<copy file="myfile.txt" tofile="mycopy.txt"/>
```

COPY a single file to a directory

```
<copy file="myfile.txt" todir="../some/other/dir"/>
```

COPY a directory to another directory

```
<copy todir="../new/dir">
                    <fileset dir="src_dir"/>
</copy>
```

COPY a set of files to a directory

```
<copy todir="../dest/dir">
      <fileset dir="src_dir">
         <exclude name="**/*.java"/>
      </fileset>
</copy>

<copy todir="../dest/dir">
      <fileset dir="src_dir" excludes="**/*.java"/>
</copy>
```

COPY a set of files to a directory, appending .bak to the file name on the fly

```
<copy todir="../backup/dir">
      <fileset dir="src_dir"/>
      <mapper type="glob" from="*" to="*.bak"/>
</copy>
```

COPY a set of files to a directory, replacing @TITLE@ with Foo Bar in all files.

```
<copy todir="../backup/dir">
      <fileset dir="src_dir"/>
      <filterset>
         <filter token="TITLE" value="Foo Bar"/>
      </filterset>
</copy>
```

COPYDIR copies the directory ${src}/resources to ${dist}.

```
<copydir src="${src}/resources" dest="${dist}"/>
```

COPYDIR copies the directory ${src}/resources to ${dist} recursively. All java files are copied, except for files with the name Test.java.

```
<copydir src="${src}/resources" dest="${dist}"
           includes="**/*.java" excludes="**/Test.java"/>
```

COPYDIR copies the directory ${src}/resources to ${dist} recursively. All java files are copied, except for the files under the mypackage/test directory.

```
<copydir src="${src}/resources" dest="${dist}"
         includes="**/*.java" excludes="mypackage/test/**"/>
```

COPYFILE

```
<copyfile src="test.java" dest="subdir/test.java"/>
```

COPYFILE

```
<copyfile src="${src}/index.html" dest="${dist}/help/index.html"/>
```

CVS checks out the package/module "jakarta-ant" from the CVS repository pointed to by the cvsRoot attribute, and stores the files in "${ws.dir}".

```
<cvs cvsRoot=":pserver:anoncvs@cvs.apache.org:/home/cvspublic"
   package="jakarta-ant"
   dest="${ws.dir}"/>
```

CVS updates the package/module that has previously been checked out into "${ws.dir}".

```
<cvs dest="${ws.dir}" command="update"/>
```

CVS silently (-q) creates a file called patch.txt which contains a unified (-u) diff which includes new files added via "cvs add" (-N) and can be used as input to patch.

```
<cvs command="-q diff -u -N" output="patch.txt"/>
```

CVS Updates from the head of repository ignoring sticky bits (-A) and creating any new directories as necessary (-d).

```
<cvs command="update -A -d"/>
```

CVSCHANGELOG Generates a change log report for all the changes that have been made under the dve/network directory. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
          destfile="changelog.xml"/>
```

CVSCHANGELOG Generates a change log report for any changes that were made under the dve/network directory in the past 10 days. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
          destfile="changelog.xml"
          daysinpast="10"/>
```

CVSCHANGELOG Generates a change log report for any changes that were made between February 20, 2002 and March 20, 2002 under the dve/network directory. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
          destfile="changelog.xml"
          start="20 Feb 2002"
          end="20 Mar 2002"/>
```

CVSCHANGELOG Generates a change log report for any changes that were made after February 20, 2002 under the dve/network directory. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
          destfile="changelog.xml"
          start="20 Feb 2002"/>
```

CVSCHANGELOG Generates a change log report for all the changes that were made under the dve/network directory, substituting the name "Peter Donald" in the <author> tags anytime it encounters a change made by the user ID "donaldp". It writes these changes into the file changelog.xml.
```
<cvschangelog dir="dve/network"
          destfile="changelog.xml"/>
  <user displayname="Peter Donald" userid="donaldp"/>
</cvschangelog>
```

CVSPASS Adds an entry into the ~/ cvspass password file.
```
<cvspass cvsroot=":pserver:anoncvs@jakarta.apache.org:/home/cvspublic"
  password="anoncvs"/>
```

CVSTAGDIFF Generates a tagdiff report for all the changes that have been made in the jakarta-ant module between the tags ANT_14 and ANT_141. It writes these changes into the file tagdiff.xml.
```
<cvstagdiff cvsRoot=":pserver:anoncvs@cvs.apache.org:/home/cvspublic"
          destfile="tagdiff.xml" package="jakarta-ant"
          startTag="ANT_14" endTag="ANT_141"/>
```

CVSTAGDIFF Generates a tagdiff report for all the changes that have been made in the jakarta-ant module in january 2002. In this example cvsRoot has not been set. The current cvsRoot will be used (assuming the build is started from a folder stored in cvs. It writes these changes into the file tagdiff.xml.
```
<cvstagdiff destfile="tagdiff.xml" package="jakarta-ant"
          startDate="2002-01-01" endDate="2002-31-01"/>
```

CVSTAGDIFF Generates a tagdiff report for all the changes that have been made in the jakarta-ant module in january 2002, with rootdir indicating that the actual location of the jakarta-ant module in cvs is jakarta/ant rather than jakarta-ant. In this example cvsRoot has not been set. The current cvsRoot will be used (assuming the build is started from a folder stored in cvs. It writes these changes into the file tagdiff.xml.
```
<cvstagdiff destfile="tagdiff.xml" package="jakarta-ant"
          rootdir="jakarta/ant" startDate="2002-01-01"
          endDate="2002-31-01"/>
```

DELETE deletes the file /lib/ant.jar.
```
<delete file="/lib/ant.jar"/>
```

DELETE deletes the lib directory, including all files and subdirectories of lib.
```
<delete dir="lib"/>
```

DELETE deletes all files with the extension .bak from the current directory and any subdirectories.
```
<delete>
  <fileset dir="." includes="**/*.bak"/>
</delete>
```

DELETE deletes all files and subdirectories of build, including build itself.
```
<delete includeEmptyDirs="true">
  <fileset dir="build"/>
</delete>
```

DELTREE deletes the directory dist, including its files and subdirectories.
```
<deltree dir="dist"/>
```

DELTREE deletes the directory ${dist}, including its files and subdirectories.
```
<deltree dir="${dist}"/>
```

DEPENDSET In this example derived HTML files in the ${output.dir} directory will be removed if any are out-of-date with respect to: the DTD of their source XML files, a common DTD (imported by the main DTD), a subordinate XSLT stylesheet (imported by the main stylesheet), or the buildfile
```
<dependset>
  <srcfilelist dir="${dtd.dir}" files="paper.dtd,common.dtd"/>
  <srcfilelist dir="${xsl.dir}" files = "common.xsl"/>
  <srcfilelist dir="${basedir}" files="build.xml"/>
  <targetfileset dir="${output.dir}" includes="**/*.html"/>
</dependset>
```

DIRNAME will set antfile.dir to the directory path for ${ant.file}.
```
<dirname property="antfile.dir" file="${ant.file}"/>
```

DIRNAME will set foo.dirname to the project's basedir.
```
<dirname property="foo.dirname" file="foo.txt"/>
```

EAR
```
<ear destfile="${build.dir}/myapp.ear" appxml="${src.dir}/metadata/application.xml">
  <fileset dir="${build.dir}" includes="*.jar,*.war"/>
</ear>
```

ECHO
```
<echo message="Hello, world"/>
```

ECHO
```
<echo>This is a longer message stretching over
        two lines.</echo>
```

ECHO As XML parsers are wont to do, the first newline in the text element has been included in the text.
```
<echo>
This is a longer message stretching over
three lines; the first line is a blank
</echo>
```

ECHO A message which only appears in -debug mode.
```
<echo message="Deleting drive C:" level="debug"/>
```

ECHO A message which appears even in -quiet mode.
```
<echo level="error">
Imminent failure in the antimatter containment facility.
Please withdraw to safe location at least 50km away.
</echo>
```

ECHO Generate a shell script by echoing to a file. Note the use of a double $ symbol to stop Ant filtering out the single $ during variable expansion
```
<echo file="runner.csh" append="false">#\!/bin/tcsh
  java-1.3.1 -mx1024m ${project.entrypoint} $$*</echo>
```

Ant Core Task Quick Reference

EXEC starts emacs on display 1 of the X Window System.
```
   <exec executable="emacs">
      <env key="DISPLAY" value=":1.0"/>
   </exec>
```

EXEC adds ${basedir}/bin to the PATH of the system command.
```
   <exec ... >
      <env key="PATH" path="${java.library.path}:${basedir}/bin"/>
   </exec>
```

FAIL will exit the current build with no further information given.
```
   <fail/>
```

FAIL will exit the current build and print something like the following to wherever your output goes
```
   <fail message="Something wrong here."/>
```

FILTER will copy recursively all the files from the src.dir directory into the dest.dir directory replacing all the occurrences of the string @year@ with 2000.
```
   <filter token="year" value="2000"/>
   <copy todir="${dest.dir}" filtering="true">
      <fileset dir="${src.dir}"/>
   </copy>
```

FILTER will read all property entries from the deploy_env.properties file and set these as filters.
```
   <filter filtersfile="deploy_env.properties"/>
```

GENKEY
```
   <genkey alias="apache-group" storepass="secret" dname="CN=Ant Group, OU=Jakarta Division,
O=Apache.org, C=US"/>
```

GENKEY
```
   <genkey alias="apache-group" storepass="secret" >
      <dname>
         <param name="CN" value="Ant Group"/>
         <param name="OU" value="Jakarta Division"/>
         <param name="O"  value="Apache.Org"/>
         <param name="C"  value="US"/>
      </dname>
   </genkey>
```

GET Gets the index page of http://jakarta.apache.org/, and stores it in the file help/index.html.
```
   <get src="http://jakarta.apache.org/" dest="help/index.html"/>
```

GET Gets the nightly ant build from the tomcat distribution, if the local copy is missing or out of date. Uses the verbose option for progress information.
```
   <get src="http://jakarta.apache.org/builds/tomcat/nightly/ant.zip"
    dest="optional.jar" verbose="true" usetimestamp="true"/>
```

GET Fetches some file from a server with access control. Because https is being used the fact that basic auth sends passwords in plaintext is moot.
```
   <get src="https://insecure-bank.org/statement/user=1214"
    dest="statement.html" username="1214" password="secret"/>
```

GUNZIP (SEE BUNZIP2)

GZIP (SEE BZIP2)

INPUT Will pause the build run until return key is pressed when using the default InputHandler, the concrete behavior is defined by the InputHandler implementation you use.
```
   <input/>
```

INPUT Will display the message "Press Return key to continue..." and pause the build run until return key is pressed (again, the concrete behavior is implementation dependent).
```
   <input>Press Return key to continue...</input>
```

INPUT Will display the message "Press Return key to continue..." and pause the build run until return key is pressed (see above).
```
   <input message="Press Return key to continue..."/>
```

INPUT Will display the message "All data is going to be deleted from DB continue (y/n)?" and require 'y' to continue build or 'n' to exit build with following message "Build aborted by user.".
```
   <input message="All data is going to be deleted from DB continue (y/n)?"
          validargs="y,n"
         addproperty="do.delete"/>
  <condition property="do.abort">
   <equals arg1="n" arg2="${do.delete}"/>
  </condition>
  <fail if="do.abort">Build aborted by user.</fail>
```

INPUT Will display the message "Please enter db-username:" and set the property db.user to the value entered by the user.
```
   <input  message="Please enter db-username:"  addproperty="db.user"/>
```

JAR jars all files in the ${build}/classes directory into a file called app.jar in the ${dist}/lib directory.
```
   <jar destfile="${dist}/lib/app.jar" basedir="${build}/classes"/>
```

JAR jars all files in the ${build}/classes directory into a file called app.jar in the ${dist}/lib directory. Files with the name Test.class are excluded.
```
   <jar destfile="${dist}/lib/app.jar"  basedir="${build}/classes"
     excludes="**/Test.class"/>
```

JAR jars all files in the ${build}/classes directory into a file called app.jar in the ${dist}/lib directory. Only files under the directory mypackage/test are used, and files with the name Test.class are excluded.
```
  <jar destfile="${dist}/lib/app.jar"  basedir="${build}/classes"
     includes="mypackage/test/**"  excludes="**/Test.class"/>
```

JAR jars all files in the ${build}/classes directory and also in the ${src}/resources directory together into a file called app.jar in the ${dist}/lib directory. Files with the name Test.class are excluded. If there are files such as ${build}/classes/mypackage/MyClass.class and ${src}/resources/mypackage/image.gif, they will appear in the same directory in the JAR (and thus be considered in the same package by Java).
```
  <jar destfile="${dist}/lib/app.jar">
   <fileset dir="${build}/classes"
         excludes="**/Test.class"
   />
   <fileset dir="${src}/resources"/>
  </jar>
```

JAVAC compiles all .java files under the ${src} directory, and stores the .class files in the ${build} directory. The classpath used includes xyz.jar, and compiling with debug information is on.

```
   <javac srcdir="${src}" destdir="${build}" classpath="xyz.jar"  debug="on"/>
```

JAVAC compiles all .java files under the ${src} directory, and stores the .class files in the ${build} directory. This will fork off the javac compiler using the default javac executable.

```
   <javac srcdir="${src}" destdir="${build}" fork="true"/>
```

JAVAC compiles all .java files under the ${src} directory, and stores the .class files in the ${build} directory. This will fork off the javac compiler, using the executable named java$javac.exe. Note that the $ sign needs to be escaped by a second one.

```
   <javac srcdir="${src}"  destdir="${build}" fork="java$$javac.exe"/>
```

JAVAC compiles .java files under the ${src} directory, and stores the .class files in the ${build} directory. The classpath used includes xyz.jar, and debug information is on. Only files under mypackage/p1 and mypackage/p2 are used. All files in and below the mypackage/p1/testpackage directory are excluded from compilation.

```
   <javac srcdir="${src}" destdir="${build}"
           includes="mypackage/p1/**,mypackage/p2/**"
          excludes="mypackage/p1/testpackage/**"
          classpath="xyz.jar"  debug="on"/>
```

JAVAC is the same as the previous example, with the addition of a second source path, defined by the property src2.

```
  <javac srcdir="${src}:${src2}"
       destdir="${build}"
       includes="mypackage/p1/**,mypackage/p2/**"
       excludes="mypackage/p1/testpackage/**"
       classpath="xyz.jar"
       debug="on"
  />
```

JAVAC Same as above with nested <src> tags

```
  <javac destdir="${build}"
       classpath="xyz.jar"
       debug="on">
   <src path="${src}"/>
   <src path="${src2}"/>
   <include name="mypackage/p1/**"/>
   <include name="mypackage/p2/**"/>
   <exclude name="mypackage/p1/testpackage/**"/>
  </javac>
```

JAVADOC

```
 <javadoc packagenames="com.dummy.test.*"
        sourcepath="src"  excludepackagenames="com.dummy.test.doc-files.*"
        defaultexcludes="yes"  destdir="docs/api"  author="true"  version="true"
        use="true"  windowtitle="Test API">
   <doctitle><![CDATA[<h1>Test</h1>]]></doctitle>
   <bottom><![CDATA[<i>Copyright &#169; 2000 Dummy Corp. All Rights
Reserved.</i>]]></bottom>
   <tag name="todo" scope="all" description="To do:" />
   <group title="Group 1 Packages" packages="com.dummy.test.a*"/>
   <group title="Group 2 Packages" packages="com.dummy.test.b*:com.dummy.test.c*"/>
   <link offline="true" href="http://java.sun.com/products/jdk/1 2/docs/api/" packagelistLoc="C:\tmp"/>
   <link href="http://developer.java.sun.com/developer/products/xml/docs/api/"/>
 </javadoc>
```

JAVADOC

```
 <javadoc  destdir="docs/api" author="true" version="true" use="true" windowtitle="Test API">
   <packageset dir="src" defaultexcludes="yes">
     <include name="com/dummy/test/**" />
     <exclude name="com/dummy/test/doc-files/**"/>
   </packageset>
   <doctitle><![CDATA[<h1>Test</h1>]]></doctitle>
   <bottom><![CDATA[<i>Copyright &#169; 2000 Dummy Corp. All Rights
Reserved.</i>]]></bottom>
   <tag name="todo" scope="all" description="To do:" />
   <group title="Group 1 Packages" packages="com.dummy.test.a*"/>
   <group title="Group 2 Packages" packages="com.dummy.test.b*:com.dummy.test.c*"/>
   <link offline="true" href="http://java.sun.com/products/jdk/1 2/docs/api/" packagelistLoc="C:\tmp"/>
   <link href="http://developer.java.sun.com/developer/products/xml/docs/api/"/>
 </javadoc>
```

LOADFILE Load file message.txt into property "message"; an <echo> can print this.

```
   <loadfile property="message" srcFile="message.txt"/>
```

LOADFILE Load a file using the latin-1 encoding

```
   <loadfile property="encoded-file"  srcFile="loadfile xml"  encoding="ISO-8859-1"/>
```

LOADFILE Load a file, don't fail if it is missing (a message is printed, though)

```
   <loadfile  property="optional.value"  srcFile="optional.txt"  failonerror="false"/>
```

LOADFILE Load a property which can be used as a parameter for another task (in this case mail), merging lines to ensure this happens.

```
   <loadfile  property="mail.recipients"  srcFile="recipientlist.txt">
     <filterchain>
       <striplinebreaks/>
     </filterchain>
   </loadfile>
```

LOADFILE Load an XML file into a property, expanding all properties declared in the file in the process.

```
   <loadfile  property="system.configuration.xml" srcFile="configuration.xml">
       <expandproperties/>
   </loadfile>
```

LOADPROPERTIES Load contents of file.properties as Ant properties.
```
<loadproperties srcFile="file.properties"/>
```

LOADPROPERTIES Read the lines that contain the string "import." from the file "file.properties" and load them as Ant properties.
```
<loadproperties srcFile="file.properties">
  <filterchain>
    <linecontains>
      <contains value="import."/>
    </linecontains>
  </filterchain>
</loadproperties>
```

MAIL Sends an email from me to you with a subject of Results of nightly build and includes the contents of the file build.log in the body of the message.
```
<mail from="me" tolist="you"  subject="Results of nightly build"  files="build.log"/>
```

MAIL Sends an eMail from me@myisp.com to all@xyz com with a subject of Test Build and attaches any zip files from the dist directory. The task will attempt to use JavaMail and fall back to UU encoding or no encoding in that order depending on what support classes are available. ${buildname} will be replaced with the buildname property's value.
```
<mail mailhost="smtp.myisp.com" mailport="1025" subject="Test build">
  <from address="me@myisp.com"/>
  <to address="all@xyz.com"/>
  <message>The ${buildname} nightly build has completed</message>
  <fileset dir="dist">
    <includes name="**/*.zip"/>
  </fileset>
</mail>
```

MANIFEST Creates or replaces the file MANIFEST.MF. Note that the Built-By attribute will take the value of the Ant property ${user.name}. The same is true for the ${version} and ${TODAY} properties. This example produces a MANIFEST.MF that contains package version identification for the package common.
```
<manifest file="MANIFEST.MF">
  <attribute name="Built-By" value="${user.name}"/>
  <section name="common">
    <attribute name="Specification-Title" value="Example"/>
    <attribute name="Specification-Version" value="${version}"/>
    <attribute name="Specification-Vendor" value="Example Organization"/>
    <attribute name="Implementation-Title" value="common"/>
    <attribute name="Implementation-Version" value="${version} ${TODAY}"/>
    <attribute name="Implementation-Vendor" value="Example Corp."/>
  </section>
  <section name="common/class1.class">
    <attribute name="Sealed" value="false"/>
  </section>
</manifest>
```

MKDIR creates a directory ${dist}.
```
<mkdir dir="${dist}"/>
```

MKDIR creates a directory ${dist}/lib.
```
<mkdir dir="${dist}/lib"/>
```

MOVE a single file (rename a file)
```
<move file="file.orig" tofile="file.moved"/>
```

MOVE a single file to a directory
```
<move file="file.orig" todir="dir/to/move/to"/>
```

MOVE a directory to a new directory
```
<move todir="new/dir/to/move/to">
  <fileset dir="src/dir"/>
</move>
```

MOVE a set of files to a new directory
```
<move todir="some/new/dir">
  <fileset dir="my/src/dir">
    <include name="**/*.jar"/>
    <exclude name="**/ant.jar"/>
  </fileset>
</move>
```

MOVE Append ".bak" to the names of all files in a directory.
```
<move todir="my/src/dir">
  <fileset dir="my/src/dir">
    <exclude name="**/*.bak"/>
  </fileset>
  <mapper type="glob" from="*" to="*.bak"/>
</move>
```

PARALLEL This example represents a typical pattern for testing a server application. In one thread the server is started (the wlrun task). The other thread consists of a three tasks which are performed in sequence. The sleep task is used to give the server time to come up. Another task which is capable of validating that the server is available could be used in place of the sleep task. The test harness is then run. Once the tests are complete, the server is stopped (using wlstop in this example), allowing both threads to complete. The parallel task will also complete at this time and the build will then continue.
```
<parallel>
  <wlrun ... >
  <sequential>
    <sleep seconds="30"/>
    <junit ... >
    <wlstop/>
  </sequential>
</parallel>
```

PARALLEL This example shows two independent tasks being run to achieve better resource utilization during the build. In this instance, some servlets are being compiled in one thead and a set of JSPs is being precompiled in another. As noted above, you need to be careful that the two tasks are independent, both in terms of their dependencies and in terms of their potential interactions in Ant's external environment.
```
<parallel>
  <javac ...> <!-- compiler servlet code -->
  <wljspc ...> <!-- precompile JSPs -->
</parallel>
```

PATCH applies the diff included in module.1.0-1.1.patch to the files in base directory guessing the filename(s) from the diff output.
```
<patch patchfile="module.1.0-1.1.patch"/>
```

PATCH like above but one leading directory part will be removed.

```
   <patch patchfile="module.1.0-1.1.patch" strip="1"/>
```

PATHCONVERT will generate the path shown below and store it in the property named wl.path.unix.
/weblogic/lib/weblogicaux.jar:/weblogic/classes:/weblogic/mssqlserver4/classes:/WINNT/SYSTEM32

```
   <path id="wl.path">
      <pathelement location="${wl.home}/lib/weblogicaux.jar"/>
    <pathelement location="${wl.home}/classes"/>
    <pathelement location="${wl.home}/mssqlserver4/classes"/>
    <pathelement location="c:\winnt\System32"/>
   </path>

  <pathconvert targetos="unix" property="wl.path.unix" refid="wl.path">
      <map from="${wl.home}" to="${wl.home.unix}"/>
    <map from="c:" to=""/>
  </pathconvert>
```

PATHCONVERT will convert the list of files to the following Unix path:
/usr/local/ant/lib/njavac.jar:/usr/local/ant/lib/xproperty.jar

```
  <filelist id="custom_tasks.jars"
       dir="${env.HOME}/ant/lib"
       files="njavac.jar,xproperty.jar"/>

   <pathconvert targetos="unix" property="custom_tasks.jars" refid="custom_tasks.jars">
     <map from="${env.HOME}" to="/usr/local"/>
   </pathconvert>
```

PATHCONVERT This example takes the set of files determined by the fileset (all files ending in .java), joins them together separated by commas, and places the resulting list into the property javafiles. The directory separator is not specified, so it defaults to the appropriate character for the current platform. Such a list could then be used in another task, like javadoc, that requires a comma separated list of files.

```
  <fileset dir="${src.dir}" id="src.files">
    <include name="**/*.java"/>
  </fileset>

  <pathconvert pathsep="," property="javafiles" refid="src.files"/>
```

PROPERTY sets the property foo.dist to the value "dist".

```
  <property name="foo.dist" value="dist"/>
```

PROPERTY reads a set of properties from a file called "foo.properties".

```
  <property file="foo.properties"/>
```

PROPERTY reads a set of properties from a resource called "foo.properties".

```
  <property resource="foo.properties"/>
```

RECORD how to use the recorder to record just the <javac> task:

```
    ...
  <compile >
      <record name="log.txt" action="start"/>
      <javac ...
      <record name="log.txt" action="stop"/>
  <compile/>
    ...
```

RECORD set up two recorders: one to file "records-simple.log" at logging level info (the default) and one to file "ISO.log" using logging level of verbose.

```
    ...
  <record name="records-simple.log"/>
  <record name="ISO.log" loglevel="verbose"/>
```

RENAME Renames the file foo.jar to ${name}-${version}.jar (assuming name and version being predefined properties). If a file named ${name}-${version}.jar already exists, it will be removed prior to renaming foo.jar.

```
   <rename src="foo.jar" dest="${name}-${version}.jar"/>
```

REPLACE replaces occurrences of the string "multi line\ntoken" with the string "wombat", in all HTML files in the directory ${src}. Where \n is the platform specific line separator.

```
   <replace dir="${src}" value="wombat">
      <include name="**/*.html"/>
      <replacetoken><![CDATA[multi line token]]></replacetoken>
   </replace>
```

RMIC runs the rmic compiler for the class com.xyz.FooBar. The compiled files will be stored in the directory ${build}/classes.

```
   <rmic classname="com.xyz.FooBar" base="${build}/classes"/>
```

RMIC runs the rmic compiler for all classes with .class files below ${build}/classes whose classname starts with Remote. The compiled files will be stored in the directory ${build}/classes.

```
   <rmic base="${build}/classes" includes="**/Remote*.class"/>
```

SEQUENTIAL This example shows how the sequential task is used to execute three tasks in sequence, while another task is being executed in a separate thread.

```
   <parallel>
      <wlrun ... >
      <sequential>
         <sleep seconds="30"/>
         <junit ... >
         <wlstop/>
      </sequential>
   </parallel>
```

SIGNJAR signs the ant.jar with alias "apache-group" accessing the keystore and private key via "secret" password

```
   <signjar jar="${dist}/lib/ant.jar" alias="apache-group" storepass="secret"/>
```

SLEEP Sleep for about 10 mS.

```
  <sleep milliseconds="10"/>
```

SLEEP Sleep for about 2 seconds.

```
<sleep seconds="2"/>
```

SLEEP Sleep for one hour less 59:58, or two seconds again

```
<sleep hours="1" minutes="-59" seconds="-58"/>
```

SLEEP Sleep for no time at all. This may yield the CPU time to another thread or process.

```
<sleep/>
```

SQL Connects to the database given in url as the sa user using the org.database.jdbcDriver and executes the SQL statements contained within the file data.sql

```
<sql driver="org.database.jdbcDriver" url="jdbc:database-url"
        userid="sa" password="pass" src="data.sql"/>
```

SQL Connects to the database given in url as the sa user using the org.database.jdbcDriver and executes a SQL statement

```
<sql driver="org.database.jdbcDriver" url="jdbc:database-url"
        userid="sa"  password="pass"><![CDATA[
    update some_table set column1 = column1 + 1 where column2 < 42;
]]></sql>
```

SQL The following connects to the database given in url as the sa user using the org.database.jdbcDriver and executes the SQL statements contained within the files data1.sql, data2.sql and data3.sql and then executes the truncate operation on some_other_table.

```
<sql driver="org.database.jdbcDriver" url="jdbc:database-url"
        userid="sa"  password="pass">
    <transaction  src="data1.sql"/>
    <transaction  src="data2.sql"/>
    <transaction  src="data3.sql"/>
    <transaction>
        truncate table some_other_table;
    </transaction>
</sql>
```

STYLE

```
<style basedir="doc" destdir="build/doc" extension=".html" style="style/apache.xsl"/>
```

STYLE Using an xmlcatalog

```
<xslt basedir="doc" destdir="build/doc" extension=".html" style="style/apache.xsl">
  <xmlcatalog refid="mycatalog"/>
</xslt>
```

STYLE

```
<xslt basedir="doc" destdir="build/doc" extension=".html" style="style/apache.xsl">
<xmlcatalog>
  <dtd publicId="-//ArielPartners//DTD XML Article V1.0//EN"
         location="com/arielpartners/knowledgebase/dtd/article.dtd"/>
</xmlcatalog>
</xslt>
```

STYLE Using XSL parameters - Then if you declare a global parameter "date" with the top-level element <xsl:param name="date"/>, the variable $date will subsequently have the value 07-01-2000.

```
<xslt basedir="doc" destdir="build/doc" extension=".html" style="style/apache.xsl">
  <param name="date" expression="07-01-2000"/>
</xslt>
```

STYLE Using output properties

```
<xslt in="doc.xml" out="build/doc/output.xml" style="style/apache.xsl">
  <outputproperty name="method" value="xml";/>
  <outputproperty name="standalone" value="yes"/>
  <outputproperty name="encoding" value="iso8859_1"/>
  <outputproperty name="indent" value="yes"/>
</xslt>
```

TAR tars all files in the htdocs/manual directory into a file called manual.tar in the ${dist} directory, then applies the gzip task to compress it.

```
<tar tarfile="${dist}/manual.tar" basedir="htdocs/manual"/>
<gzip zipfile="${dist}/manual.tar.gz" src="${dist}/manual.tar"/>
```

TAR tars all files in the htdocs/manual directory into a file called manual.tar in the ${dist} directory. Files in the directory mydocs, or files with the name todo.html are excluded.

```
<tar destfile="${dist}/manual.tar"
     basedir="htdocs/manual"
     excludes="mydocs/**, **/todo.html"
/>
```

TAR Writes the file docs/readme.txt as /usr/doc/ant/README into the archive. All *.html files in the docs directory are prefixed by /usr/doc/ant, so for example docs/index.html is written as /usr/doc/ant/index.html to the archive.

```
<tar destfile="${basedir}/docs.tar">
  <tarfileset dir="${dir.src}/docs"
        fullpath="/usr/doc/ant/README"
        preserveLeadingSlashes="true">
    <include name="readme.txt"/>
  </tarfileset>
  <tarfileset dir="${dir.src}/docs"
        prefix="/usr/doc/ant"
        preserveLeadingSlashes="true">
    <include name="*.html"/>
  </tarfileset>
</tar>
```

TAR This example shows building a tar which uses the GNU extensions for long paths and where some files need to be marked as executable (mode 755) and the rest are use the default mode (read-write by owner). The first fileset selects just the executable files. The second fileset must exclude the executable files and include all others.

```
<tar longfile="gnu" destfile="${dist.base}/${dist.name}-src.tar" >
  <tarfileset dir="${dist.name}/.." mode="755" username="ant" group="ant">
    <include name="${dist.name}/bootstrap.sh"/>
    <include name="${dist.name}/build.sh"/>
  </tarfileset>
  <tarfileset dir="${dist.name}/.." username="ant" group="ant">
    <include name="${dist.name}/**"/>
    <exclude name="${dist.name}/bootstrap.sh"/>
    <exclude name="${dist.name}/build.sh"/>
  </tarfileset>
</tar>
```

TASKDEF makes a task called myjavadoc available to Ant. The class com.mydomain.JavadocTask implements the task.

```
<taskdef name="myjavadoc" classname="com.mydomain.JavadocTask"/>
```

TEMPFILE will set temp.file to the name of a new temporary file.

```
<tempfile property="temp.file"/>
```

TEMPFILE will set temp.file to the name of a new temporary file with a suffix of .xml.

```
<tempfile property="temp.file" suffix=".xml"/>
```

TEMPFILE will set temp.file to the name of a new temporary file located in the build sub-directory.

```
<tempfile property="temp.file" destdir="build"/>
```

TOUCH creates myfile if it doesn't exist and changes the modification time to the current time.

```
<touch file="myfile"/>
```

TOUCH creates myfile if it doesn't exist and changes the modification time to Jun, 28 2000 2:02 pm (14:02 for those used to 24 hour times).

```
<touch file="myfile" datetime="06/28/2000 2:02 pm"/>
```

TOUCH changes the modification time to Oct, 09 1974 4:30 pm of all files and directories found in src_dir.

```
<touch datetime="09/10/1974 4:30 pm">
  <fileset dir="src_dir"/>
</touch>
```

TSTAMP sets the standard DSTAMP, TSTAMP, and TODAY properties according to the default formats.

```
<tstamp/>
```

TSTAMP sets the standard properties as well as the property TODAY_UK with the date/time pattern "d-MMMM-yyyy" using English locale (eg. 21-May-2001).

```
<tstamp>
  <format property="TODAY_UK" pattern="d-MMMM-yyyy" locale="en"/>
</tstamp>
```

TSTAMP Creates a timestamp, in the property touch.time, 5 hours before the current time. The format in this example is suitable for use with the <touch> task. The standard properties are set also.

```
<tstamp>
    <format property="touch.time" pattern="MM/dd/yyyy hh:mm aa"
            offset="-5" unit="hour"/>
</tstamp>
```

TSTAMP Sets three properties with the standard formats, prefixed with "start.": start.DSTAMP, start.TSTAMP, and start.TODAY.

```
<tstamp prefix="start"/>
```

TYPEDEF makes a data type called urlset available to Ant. The class com.mydomain.URLSet implements this type.

```
<typedef name="urlset" classname="com.mydomain.URLSet"/>
```

UN[JAR|ZIP|TAR]

```
<unzip src="${tomcat_src}/tools-src.zip" dest="${tools.home}"/>
```

UN[JAR|ZIP|TAR]

```
<gunzip src="tools.tar.gz"/>
```

UN[JAR|ZIP|TAR]

```
<untar src="tools.tar" dest="${tools.home}"/>
```

UN[JAR|ZIP|TAR]

```
<unzip src="${tomcat_src}/tools-src zip"
   dest="${tools.home}">
<patternset>
    <include name="**/*.java"/>
    <exclude name="**/Test*.java"/>
</patternset>
</unzip>
```

UN[JAR|ZIP|TAR]

```
<unzip dest="${tools.home}">
<patternset>
    <include name="**/*.java"/>
    <exclude name="**/Test*.java"/>
</patternset>
<fileset dir=".">
    <include name="**/*.zip"/>
    <exclude name="**/tmp*.zip"/>
</fileset>
</unzip>
```

UPTODATE sets the property xmlBuild.notRequired to true if the ${deploy}/xmlClasses.jar file is more up-to-date than any of the DTD files in the ${src}/xml directory.

```
<uptodate property="xmlBuild.notRequired" targetfile="${deploy}\xmlClasses.jar" >
    <srcfiles dir= "${src}/xml" includes="**/*.dtd"/>
</uptodate>
```

WAITFOR waits up to 30 seconds for a file called errors.log to appear.

```
<waitfor maxwait="30" maxwaitunit="second">
      <available file="errors.log"/>
</waitfor>
```

WAITFOR waits up to 3 minutes (and checks every 500 milliseconds) for a web server on localhost to serve up the specified URL.

```
<waitfor maxwait="3" maxwaitunit="minute" checkevery="500">
   <http url="http://localhost/myapp/index.html"/>
</waitfor>
```

WAITFOR waits up to 10 seconds for a server on the dbserver machine to begin listening on port 1521 and for the http://webserver/mypage.html web page to become available.

```
<waitfor maxwait="10" maxwait="second">
      <and>
      <socket server="dbserver" port="1521"/>
      <http url="http://webserver/mypage.html"/>
      </and>
</waitfor>
```

WAR
```
  <war destfile="myapp.war" webxml="src/metadata/myapp.xml">
    <fileset dir="src/html/myapp"/>
    <fileset dir="src/jsp/myapp"/>
    <lib dir="thirdparty/libs">
      <exclude name="jdbc1.jar"/>
    </lib>
    <classes dir="build/main"/>
    <zipfileset dir="src/graphics/images/gifs"
                prefix="images"/>
  </war>
```

XMLPROPERTY Load contents of somefile.xml as Ant properties, generating the property names from the file's element and attribute names.
```
  <xmlproperty file="somefile xml" />
 XML FILE:
   <root-tag myattr="true">
     <inner-tag someattr="val">Text</inner-tag>
     <a2><a3><a4>false</a4></a3></a2>
   </root-tag>
  PROPERTIES:
 root-tag(myattr)=true
 root-tag inner-tag=Text
 root-tag.inner-tag(someattr)=val
 root-tag.a2.a3.a4=false
```

XSLT (SEE STYLE)

ZIP zips all files in the htdocs/manual directory into a file called manual.zip in the ${dist} directory.
```
 <zip destfile="${dist}/manual.zip"  basedir="htdocs/manual"/>
```

ZIP zips all files in the htdocs/manual directory into a file called manual.zip in the ${dist} directory. If manual.zip doesn't exist, it is created; otherwise it is updated with the new/changed files.
```
 <zip destfile="${dist}/manual.zip" basedir="htdocs/manual" update="true"/>
```

ZIP zips all files in the htdocs/manual directory. Files in the directory mydocs, or files with the name todo html are excluded.
```
 <zip destfile="${dist}/manual.zip" basedir="htdocs/manual" excludes="mydocs/**, **/todo.html"/>
```

ZIP zips all files in the htdocs/manual directory. Only html files under the directory api are zipped, and files with the name todo.html are excluded.
```
 <zip destfile="${dist}/manual.zip" basedir="htdocs/manual" includes="api/**/*.html"
     excludes="**/todo.html"/>
```

ZIP zips all files in the htdocs/manual directory, and also adds the file ChangeLog.txt in the current directory. ChangeLog.txt will be added to the top of the ZIP file, just as if it had been located at htdocs/manual/ChangeLog.txt.
```
 <zip destfile="${dist}/manual.zip">
   <fileset dir="htdocs/manual"/>
   <fileset dir="." includes="ChangeLog.txt"/>
 </zip>
```

ZIP zips all files in the htdocs/manual directory into the docs/user-guide directory in the archive, adds the file ChangeLog27.txt in the current directory as docs/ChangeLog.txt, and includes all the html files in examples.zip under docs/examples.
```
 <zip destfile="${dist}/manual.zip">
   <zipfileset dir="htdocs/manual" prefix="docs/user-guide"/>
   <zipfileset dir="." includes="ChangeLog27.txt" fullpath="docs/ChangeLog.txt"/>
   <zipfileset src="examples.zip" includes="**/*.html" prefix="docs/examples"/>
 </zip>
```

Original Authors:

Stephane Bailliez (sbailliez@imediation.com) , Nicola Ken Barozzi (nicolaken@apache.org)
Jacques Bergeron (jacques.bergeron@dogico.com) , Stefan Bodewig (stefan.bodewig@epost.de)
Patrick Chanezon (chanezon@netscape.com) , James Duncan Davidson (duncan@x180 com)
Tom Dimock (tad1@cornell.edu) , Peter Donald (donaldp@apache.org)
dIon Gillard (dion@apache.org) , Erik Hatcher (ehatcher@apache.org)
Diane Holt (holtdl@yahoo.com) , Bill Kelly (bill kelly@softwired-inc.com)
Arnout J. Kuiper (ajkuiper@wxs.nl), Conor MacNeill
Stefano Mazzocchi (stefano@apache.org) , Erik Meade (emeade@geekfarm.org)
Sam Ruby (rubys@us.ibm.com), Nico Seessle (nico@seessle.de)
Jon S. Stevens (jon@latchkey.com), Magesh Umasankar
Roger Vaughn (rvaughn@seaconinc.com) , Dave Walend (dwalend@cs.tufts.edu)
Phillip Wells (philwells@rocketmail.com) , Craeg Strong (cstrong@arielpartners.com)

Contributors:

Work compiled by: Tim O'Brien tobrien@transolutions net