

Application Test Ant Task

Application Test ANT task executes DevTest tests. Most continuous build and integration systems recognize the XML output files and integrate the build dashboard with the test results.

This task is a direct subclass of the JUnit task. Therefore, the `junitlisa` task has the same attributes and nested elements as the JUnit task.

In addition to the attributes inherited from the JUnit task, the **junitlisa** task has the following attributes:

- **suite**
The file name of a suite document.
Example: suite="AllTestsSuite.ste"
- **test**
The file name of a test case.
Example: test="multi-tier-combo.tst"
- **stagingDoc**
The file name of a staging document.
Example: stagingDoc="Run1User1Cycle.stg"
- **config**
A named internal configuration set or a file name.
Example: config="project.config"
- **outfile**
The file name that is used to write reporting data. If the value does not comply with the standard naming scheme for `junitlisareport`, specify a fully configured `junitreport` task instead.
Example: outfile="report"
- **registry**
A pointer to the registry to use when you want to stage the test cases remotely.
Example: registry="tcp://testbox:2010/Registry"
- **preview**
Enables you to write out the name and description of each test case, without executing the test cases.
Example: preview="true"
- **user**
Specifies the user name for ACL.
Example: user="admin"
- **password**
Specifies the password for ACL.
Example: password="admin"
- **mar**
The file name of a MAR document.
Example: mar="example.mar"
- **mari**
The file name of a MAR info document.
Example: mari="example.mari"

The **junitlisa** task includes a nested element named **lisatest**. This element has the following attributes:

- **suite**
The file name of a suite document.
Example: suite="AllTestsSuite.ste"

- **test**
The file name of a test case.
Example: test="multi-tier-combo.tst"
- **stagingDoc**
The file name of a staging document.
Example: stagingDoc="Run1User1Cycle.stg"
- **mar**
The file name of a MAR document.
Example: mar="example.mar"
- **mari**
The file name of a MAR info document.
Example: mari="example.mari"

The attribute values can use curly braces, which are resolved in the usual way.

You are required to specify at least one test or suite. You can specify a test or suite in a **lisatest** nested element or in the **test** or **suite** attribute. You can specify multiple tests and suites by adding more **lisatest** elements. The tests and suites are executed in the order in which they appear in the XML.

When you run a single test with the **test** attribute, the test has the following default behavior:

- Staged with a single vuser.
- Run once.
- 100 percent think time.

To change this default behavior, wrap the test in a suite and specify an alternative staging document.

lisareport Ant Task

You can produce HTML-based reports with the **junitlisareport** task or the regular **junitreport** task.

The **junitlisareport** task is a subclass of the regular **junitreport** element, except that sensible defaults are specified. It is equivalent to the following code:

```
<junitreport todir="${testReportDir}">
    <fileset dir="<todir specified in the junitreport tag>">
        <include name="TEST-*.xml"/>
    </fileset>

    <report format="frames" todir="<todir specified in the junitreport tag>">/
>

</junitreport>
```

You can specify your own file set and report. Because the task is a direct subclass of **junitreport**, all the attributes and nested elements that **junitreport** has are supported.

We recommend specifying the inherited **toDir** attribute in most cases, although it defaults to the current working directory.

Sample LISA ANT build file –



Integration with Bamboo

Configure an ANT task in Bamboo - <https://confluence.atlassian.com/display/BAMBOO/Ant>

To configure an Ant task:

1. Navigate to the **Tasks** configuration tab for the job (this will be the default job if creating a new plan).
2. Click the name of an existing Ant task, or click **Add Task** and then **Ant** to create a new task.
3. Complete the following settings:

Task Description	A description of the task, which is displayed in Bamboo.
Disable this task	Check, or clear, to selectively run this task.
Executable	<p>The Ant executable that is available to perform the task. The executable that you select will become one of the task's (and so, the job's) requirements.</p> <p>You can add other executables, if required.</p>
Build File	<p>The name of your existing build file (e.g. <code>build.xml</code>).</p> <p>You can include variables (see Using Global or Build-specific Variables).</p>
Target	<p>The Ant target that you want this Bamboo task to execute (e.g. <code>test</code>).</p> <p>You can use <code>'-D'</code> to define one or more JVM parameters (e.g.: <code>-Djava.awt.headless="true"</code>). You must use double quotes around the parameter value; single quotes are considered as part of the actual value.</p> <p>Multiple Ant targets can be specified with a space-delimited list.</p> <p>You can also include variables (see Using Global or Build-specific Variables).</p>

Build JDK

The JDKs that are available to perform the task. The JDK that you select will become one of the task's (and so, the job's) requirements. You [can add other JDKs](#), if required.

Environment Variables

(Optional) Additional system environment variables that you want to pass to your build. Note that existing environment variables are automatically available to the executable. You can also include Bamboo global or build-specific variables (see [Using Global or Build-specific Variables](#)).


Multiple variables should be separated with spaces. Parameters with spaces must be quoted (e.g ANT_OPTS="-Xms200m -Xmx700m").

Working Sub Directory

(Optional) An alternative subdirectory, relative to the job's root directory, where Bamboo will run the executable. The root directory contains everything checked out from the job's configured source repository. If you leave this field blank, Bamboo will look for build files in the root directory. This option is useful if your task has a build script in a subdirectory and the executable needs to be run from within that subdirectory.

The build will produce test results

Select to specify the directory, relative to the root directory, where test results will be created. You can use Ant-style patterns such as `**/test-reports/*.xml`. Bamboo requires test results to be in JUnit XML format.

 For jobs that use CVS, the root directory is `<bamboo-home>/xml-data/build-dir/JOB_KEY/<cvcs-module>`.

4. Click **Save**.

Ant configuration

[How to use the Ant task](#)

Task description

☐ Disable this task

Executable

Ant ▼ [Add new executable](#)

Build file

Target*

clean test

The target you want to execute. You can also define system properties such as `-Djava.Awt.Headless=true`.

Build JDK*

JDK 1.7 ▼ [Add new JDK](#)

Which JDK do you need to use for the build? the `JAVA_HOME` will be added as an environment variable.

Environment variables

Extra environment variables. e.g. `JAVA_OPTS="-Xmx256m -Xms128m"`. You can add multiple parameters separated by a space.

Working sub directory

Specify an alternative sub-directory as working directory for the task.

Where should Bamboo look for the test result files?

☒ The build will produce test results.

If checked, the build will fail if no tests are found. Test output must be in [JUnit](#) XML format.

Specify custom results directories

**/test-reports/*.xml

Where does the build place generated test results?

this is a comma separated list of test result directories. You can also use Ant style patterns such as `**/test-reports/*.xml`

[Save](#) [Cancel](#)

