

# Ant in the 'hood

Ant users will be happy to learn that CTL has excellent support for Ant. Ant is a first class citizen. Nearly the entire CTL Java API is exposed as Ant tasks and types. You will need these CTL tasks to enable Ant automation to be useful for distributed automation jobs. All [context data](#) provided by CTL is exposed as Ant properties files. This means you can bring your Ant skills to bear and work within an Ant friendly environment and accomplish making flexible, maintainable and operational deployment and control process.

The figure above highlights the Ant support in the CTL software framework. CTL adopts the "batteries included" metaphor incorporating a full Ant distribution (currently 1.7), CTL-specific and third party task libraries ([ant-contrib](#), [antxtras](#)) and a wide variety of supporting Java libraries. You can also add your own Ant library dependencies by copying them into the CTL's `ANT_HOME/lib` directory.

## Ant commands

The CTL `type.xml` module definition file is used to define new commands. To define a command using Ant tasks declare `command-type="AntCommand"`. For the body of your command, define it inside a set of `<implementation></implementation>` tags. Ant commands are, of course, defined using Ant tasks.

An example is shown below:

```
<command name="hello" description="say hello with Ant."
  command-type="AntCommand" is-static="true">
  <implementation>
    <!--
      ** your command implmentation
      ** tasks go here.
    -->
    <echo message="hi there"/>
  </implementation>
</command>
```

You can use any Ant tasks. For convenience, the `ant-contrib` tasks are already declared.

## CTL's Controller Ant task

You are no doubt familiar with Ant tasks like [Ant](#) and [AntCall](#) to call another build file or another target in the same build file. These mechanisms allow you to break your builds into

pieces so they are smaller, simpler and modular. CTL was designed with that kind of modularity in mind too and provides a similar mechanism, enabling one command to call another command.

CTL provides an Ant task called [Controller](#) that gives you access to CTL's command dispatcher. Using this task you can invoke commands in local or remote modules.

The controller task includes a set of other tags letting you specify command line options and set any additional properties for the data context of the receiving command.

```
<controller>
  <execute>
    <context depot="${context.depot}"/>
    <command name="commandName" module="moduleName"/>
    <arg line="-opt1 arg1 -opt2 arg2"/>
    <property name="a" value="a value"/>
    <property name="b" value="b value"/>
    <property name="c" value="c value"/>
  </execute>
</controller>
```

### Where to go from here?

Be sure to visit the [Ant by example cookbook](#) for a series of practical examples for typical use cases. Familiarize yourself with CTL's Ant tasks and types.

[Next: CTL in 5 minutes #](#)