

Programming Perspective

An experienced Ant programmer is accustomed to editing their build xml files, adding targets and tasks to automate their build life cycle. While Ant provides many core and optional tasks to carry out low level procedures for the deployment part of the application life cycle, one eventually finds the limitations of Ant by itself and seeks external tools and facilities to support the overall process.

The CTL framework focusses on distributed management aspects required to handle the deployment process, as well as, allow experienced Ant users a means to define automation in Ant. But how does one integrate their existing Ant build code to the CTL deployment and management framework? What perspective should one have when thinking about how to program their automation using Ant and CTL? There is no one absolute answer but there are three alternatives which should be considered.

1. Ant calls CTL

The first obvious starting point is to call CTL from your Ant build procedure. In this scenario, you can simply call CTL commands via Ant's `exec` task, the same way you would run CTL commands via the shell.

The figure below shows an Ant project called "mybuild" that uses the `exec` task to call a command defined in CTL. The command defined in CTL can itself be implemented in Ant.

When considering calling CTL commands from your Ant project, recall there are two styles of commands you can use:

```
<!-- dispatch a arbitrary shell script to all deployhosts: -->
<exec executable="ctl-exec">
  <arg line="-I tags=deployhosts -s myscript.sh"/>
</exec>

... or ...

<!-- dispatch deploy command to all deployhosts: -->
<exec executable="ctl">
  <arg line="-I tags=deployhosts -m mymodule -c deploy"/>
</exec>
```

This approach makes sense when you want to have the build process kick off a separately defined deploy process.

2. CTL calls Ant

You might already have a working procedure defined in an Ant build file. You can invoke external build files from CTL.

The figure below describes a CTL defined command called "callAnt" which uses the ant Ant task to invoke the "run" target in an external build file called "mybuild":

In this example, the "run" target uses the exec task to call a start.sh script passing to it a command line argument. As the code fragment shows below you can pass parameters as named properties via the property sub element:

```
<!-- call an external Ant build file: -->
<ant antfile="mybuild.xml" target="run">
  <property name="param1" value="vall1"/>
</exec>
```

This approach works best when an existing build file lives outside of the CTL framework yet must be called by some step managed by a CTL command.

3. CTL solo

The final programming choice is to implement your procedure completely inside a CTL command definition. In this scenario, an experienced Ant user prefers to implement their function using Ant tasks of their choice.

Specify "AntCommand" as the command-type and then insert the Ant code and logic in between <implementation></implementation> tags.

In this example, the exec task is used to invoke the "start.sh" script.

```
<!-- call the start script -->
<exec executable="start.sh">
  <arg line="-arg1 ${opts.param1}" />
</exec>
```

You will also notice that the "run" command definition also includes the definition of a command line option called "param1". The argument to the "-arg1" option is passed in as the opts.param1 property reference.

Defining a CTL command using Ant code inside the command definition makes your procedure's implementation completely standalone. This choice is often preferred when you are comfortable scripting in Ant, are familiar with the Ant task library and want to leverage [CTL's built in support for Ant](#).