

# Command options

## Table of contents

1 Command options via GetOpts.....	2
------------------------------------	---

DRAFT: May need corrections!

An important requirement to defining commands that will operate in different contexts, is to avoid hard coding. The use of command options will provide a flexible way to parameterize your procedures.

## 1. Command options via GetOpts

The CTL [get-opts](#) task provides a command line option parsing facility. This task accepts one or more `<opt>` tags, each defining a new option. Options can be defined to take an argument or work like a boolean flag. Options that take arguments can also have a default value assigned if the option is not provided by the user. The values of these options can be referenced from within your implementations as a property value.

The example below describes a command that takes an option called `-message`. The value of the `-message` argument is used by the `echo` task:

```
<command name="hello" description="say hello with Ant."
  command-type="AntCommand" is-static="true">
  <implementation>
    <echo message="{opts.message}"/>
  </implementation>
  <!--
    ** Define an option called "-message".
    ** The command line will be parsed and set the
    ** value to the property named opts.message
    -->
  <opts>
    <opt parameter="message" description="option name"
required="false"
      property="opts.message" type="string" default="hello"/>
  </opts>
</command>
```

Running the command without argument will return the default string "hello":

```
ctl -m poly -c hello
hello
```

Now run the command with an argument:

```
ctl -m poly -c hello -- -message hola
hola
```

### Attributes and options

The ProjectBuilder's build-type command will generate a file called `type.properties` that is automatically read in before the tasks within the implementation tags.

```
<!--
  ** define an attribute named "message" to use as a default.
  -->
<attributes>
  <attribute-default name="message" value="guten tag"/>
</attributes>
<commands>
  <command name="hello" description="say hello with Ant."
    command-type="AntCommand" is-static="true">
    <implementation>
      <echo message="{opts.message}"/>
    </implementation>
    <opts>
      <!--
        ** the "defaultproperty" (if set) takes precedence
        over the literal value, "hello"
      -->
      <opt parameter="message" description="option name"
required="false"
        property="opts.message" type="string" default="hello"
        defaultproperty="entity.attribute.message"/>
    </opts>
  </command>
</commands>
```

```
ctl -m poly -c hello
guten tag
```

Of course, the command line has overall precedence:

```
ctl -m poly -c hello -- -message hi
hi
```