

Running ad-hoc commands via ctl-exec

Table of contents

1 Command Line.....	2
1.1 Files and Environment Variables.....	2
1.2 Default project.....	2
1.3 Options.....	3
1.4 Listings.....	4

This document assumes you have the CTL software installed on a single node that has an SSH configuration supporting key-based authentication.

Other documents that are useful to new CTL users:

- [Download CTL](#) - Get and install the CTL software.
- [Project Setup](#) - Setup a CTL project.
- [Running overview](#) - Running ctl-exec versus ctl.

Note:

Ensure the CTL executables are in your PATH and CTL_BASE and CTL_HOME environment variables are also set.

1. Command Line

If you've installed CTL as described in the [Installing CTL](#) section, running the ctl-exec command-line is simple: just type `ctl-exec`.

The `ctl-exec` command is used to execute commands to registered nodes. The following sections describe how to list nodes and invoke commands via `ctl-exec`.

For a detailed discussion of the command line consult the [CTL-EXEC command reference](#).

1.1. Files and Environment Variables

Wrapper scripts exist for Unix and Windows to run the `ctl-exec` launcher. These launcher scripts read an initialization profile. For Unix OSes, the `$CTL_BASE/etc/profile` file is read. On Windows, `%CTL_BASE%\etc\profile.bat` is called. These files are created by the `ctl-setup` command.

The wrapper scripts depend on the following environment variables:

- `CTL_HOME` - full path to the CTL software install base
- `CTL_BASE` - full path to the CTL repository base

If either is not set, the `ctl-exec` command will return an error.

1.2. Default project

CTL commands are always performed within a project. For the sake of convenience you can omit specifying the project parameter if there is just one project. By convention, CTL will create a default project (named "default") where you will initially begin loading modules and running commands.

If there is more than one project, the `ctl-exec` command will give an error saying "no project specified". You will then need to supply the `-p project` parameter.

1.3. Options

CTL has a variety of options that can be displayed via `-help`. Typing `ctl-exec -help` shows a complete listing of all the `ctl-exec` options and some examples.

The general option pattern follows this form:

```
ctl-exec [options] [-- [command-options]]
```

For example, the following command shows several `ctl-exec` options.

```
ctl-exec -I 'web.*' -- ps
```

When no arguments are specified, `ctl-exec` falls into listing mode. See [below](#).

Remember to use the `--` (double dash) to separate the `ctl-exec` options from those of the command you wish to execute.

1.3.1. Node Dispatching Options

The `ctl-exec` shell command includes a feature called [node dispatch](#) that allows you to call commands across a set of hosts without having to specify specific host names. Internally, CTL looks up the nodes that have the module (or object if that is what you specified), and then dispatches that command to those matching nodes. Options provide a way to filter the lookup results.

Also, using either of the two following options enables `nodedispatch`.

- `-I, --nodes`: Run the command on all nodes matching the regular expression.
- `-X, --xnodes`: Do not run the command on any node matching the regular expression. If no `-I` flag is specified, all nodes are looked up and then those matching the `-X` expression are filtered out.

See the document [Node Filtering Options](#) for further discussion of node filtering.

Two other options of interest for "nodedispatch" are:

- `-C, --threadcount`: For all the matching `nodedispatch` results, execute the commands in the specified number of threads.
- `-K`: If specified, `ctl-exec` will continue if any errors occur.

Supporting node dispatching, is a configuration file called [nodes.properties](#), that declares to

your node information. See the [server setup](#) section for information about setting up a repository to centrally maintain and distribute the `nodes.properties` file.

Here are some `ctl-exec` examples using node dispatch.

1. Check if the web servers are listening on port 80:

```
ctl-exec --nodes "web.*" -- netstat -an |grep 80
```

2. Run `who` command on all the dev machines except don't run it on localhost:

```
ctl-exec --nodes ".*.dev.acme.com" --xnode localhost -- who
```

3. Run the `pgrep` command across all the machines in dev using 10 threads:

```
ctl-exec -C 10 -I ".*.dev.acme.com" -X localhost -- pgrep httpd
```

1.4. Listings

The `ctl-exec` command supports a listing feature allowing you to list all the projects and their contents: modules their commands, and possibly their objects.

As mentioned earlier the install will have created a project called "default" along with a set of modules.

The example below shows modules in the default project:

```
$ ctl-exec  
strongbad dev@development centos
```

The example above gave an overview of the modules and commands in that project but if you want to know more about the commands for a given module add the `-m modulename` arguments:

```
$ ctl-exec -I os-family=Linux  
centos
```

[Next: Running ctl #](#)