Ortus Solutions

Tweet

CommandBox 4.5.0-rc (Release Candidate) Ready For Testing

Brad Wood

Dec 10, 2018

We are pleased to announce that **CommandBox 4.5.0-rc** is available for testing. This is a new minor release with over 25 completed tickets. There are a lot of little fixes and a handful of nice features, several of which I'd like to get a few eyeballs on before setting the release in stone. 4.5 will only be in release candidate for a week or two so please test it soon or all bugs will be deferred until next year! :)

Download

You can access the **4.5.0-rc** builds on our download site here:

http://downloads.ortussolutions.com/#/ortussolutions/commandbox/4.5.0-RC/

The auto-generated command docs are here:

https://apidocs.ortussolutions.com/commandbox/4.5.0-RC/index.html

Overview

Here's the stuff you need to know about the 4.5.0-rc release.

Bug Fixes

We've fixed a handful of little bugs here and there:

- Argument parsing issues
- Exit code support
- External modules work again
- Better cleanup of temp folder
- Silence annoying ESAPI warnings in the console

Better Java 9+ support

CommandBox is still on Lucee 5.2.9.31 so Java 9, 10, and 10 work but I still recommend staying on Java 8 until we fully make it to Lucee 5.3 due to some remaining issues. We did squash some annoying warnings in the console that shows up in the new versions of java about illegal reflective access.

No more Oracle JDK-- OpenJDK FTW!

When you download our "JRE Included" bundle, we no longer bundle Oracle JDK due to their change in licensing and the fact that Oracle JDK 8 will have no more free public patches! We're using OpenJDK builds from AdoptOpenJDK. We're still bundling Java 8, but will switch to Java 11 as soon as we can get over to a stable version of Lucee 5.3. **I need help testing the bundled JREs on Mac and Linux specifically!**

Install OpenJDK builds

Speaking of OpenJDK and the rapid-fire 6 month Java release cycles, we've taught CommandBox how to install Java for you now via the nice API that's available from AdoptOpenJDK. There is a new "java" endpoint in CommandBox that will download and install any OpenJDK build from Adopt. They have one of

the largest build farms of OpenJDK builds including all operating systems, x32/x64 bit, JDK/JREs, and even OpenJ9 vs Hostpot builds. If you wanted to install the latest version of OpenJDK 11 into a folder, you can simply type this:

```
install java:openjdk8
```

That's the short version. The Java endpoint lets you dial in the exact version of OpenJDK you want by appending optional text to your install string.

```
<version>_<type>_<arch>_<os>_<jvm-implementation>_<release>
```

Here's an example of a very specific OpenJDK version.

```
install java:OpenJDK8_jre_x64_windows_hotspot_8u181b13
```

CommandBox will automatically default to your current operating system, CPU architecture, as well as favor the JRE (over the JDK) and the hotspot JVM unless you specify otherwise. See the ticket here for more details on all the possible options: COMMANDBOX-911

Start Servers on any Java Version

That's not all! Building on top of the new Java endpoint is the ability for you to declare what version of Java you'd like your server to run on and CommandBox will take care of all the work. downloading the JRE if necessary and modifying your server's settings to use that custom Java version. The format you specify your Java version supports all the same options as the java endpoint, so you can just generically ask for the latest JDK10 build or you can dial in a very specific revision.

You can do a one-off server start

```
server start javaVersion=openJDK11
```

Or you can save the Java version in your **server.json** file

```
server set jvm.javaversion=openJDK11
server start
```

Or you can set a global default version of Java for all servers to use.

```
config set server.defaults.jvm.javaVersion=openJDK11
server start
```

And of course, the out-of-the box default is still to use whatever Java version the CLI itself is running on. Also, you can skip any checks that go out to the internet by using a specific OpenJDK release name and having that release in your local artifacts.

New Java Command Namespace

With all of these new OpenJDK integrations, we've added in a new namespace of commands to help you manage your local OpenJDK installations!

Search through the AdoptOpenJDK API to find all the supported versions they have. The output of the search will give you copy/paste IDs you can use to dial in your Java versions.

```
java search version=openjdk8
java search version=openjdk8 arch=x32
java search version=openjdk11 os=linux type=jdk arch=
```

Install a new version of Java into CommandBox internal registry so it's ready to

be used by your servers.

```
java install openjdk10_windows_x64
```

List out the installed versions of Java that CommandBox is managing. As many servers as you want can share a single Java install.

```
java list
```

A helper command to set the default JRE to be used for servers.

```
java setDefault openjdk10_x64_jdk
```

Java versions will be downloaded on-demand as necessary if they aren't already installed or found in local artifacts.

Shell Environment Variables

This feature has been a long time coming. CommandBox already has a concept of "system settings" which are the combination of Java system properties and OS env vars which you can access with the dollar sign and curly brace shell expansions. Now, each executing command will have its own local set of environment variables it can set and read. These vars will be available to sub commands, but will go away as soon as the top command finishes. This is very handy for recipes. Working with env vars looks like this:

```
set foo=brad
echo $ {foo}
```

The dollar sign expansions will look for an env var first, check the parent command's env vars, if applicable, until it reaches the shell's global env vars,

then it will check java system props, and then operating system env vars last. There's also a new "env" command namespace to allow you programmatic access to your command's vars from the CLI.

```
# Set a var
env set foo=bar
or
set foo=bar

# output a var
env show foo default
or
echo $ {foo:default}

# clear a var
env clear foo
```

Running **env show** with no params returns a struct with all variables. There is also an **env debug** command to help you debug what variables you have set and where. The latest release of CommandBox Dotenv has been updated to read .env files prior to every command execution from the current working directory, taking advantage of these new command-specific env vars, so the variables don't live past the current command execution.

ETA in the Download Progress Bar

Downloading a large file over a slow connection? There's now a new ETA in the progress bar that tells you how long you've got.

Server Directory Browser Defaults to OFF!

This is a backwards incompatible change we made so CommandBox servers would be more secure by default. When you start a server, directory browsing is disabled unless you specifically turn it on. You can get back the default behavior

by setting your global server defaults:

```
config set server.defaults.web.directoryBrowsing=true
```

Removed Stack Trace on Errors

In the unlikely event you encounter an error in the CLI, there is no stack trace by default now to tidy up the screen. You can get the old behavior back with this command:

```
config set verboseErrors=true
```

Honorable Mentions

- The **outdated** and **update** commands are much faster as they use threads to check Forgebox in parallel.
- Multiple Forgebox endpoints are now supported in preparation for Forgebox Enterprise
- Out-of-the-box integration for TestBox 2.9's upcoming code coverage tooling in the **testbox run** command
- Improved server heap handling. Use shortcuts like 1G instead of 1024m and no max heap size by default.