Cordova Command-line-interface (CLI) Reference

Syntax

```
cordova <command> [options] -- [platformOpts]
```

Global Command List

These commands are available at all times.

Command	Description
create	Create a project
help	Get help for a command
telemetry	Turn telemetry collection on or off

Project Command List

These commands are supported when the current working directory is a valid Cordova project.

Command	Description
info	Generate project information
requirements	Checks and print out all the installation requirements for platforms specified
platform	Manage project platforms
plugin	Manage project plugins
prepare	Copy files into platform(s) for building
compile	Build platform(s)
clean	Cleanup project from build artifacts

Command	Description
run	Run project (including prepare && compile)
serve	Run project with a local webserver (including prepare)

Common options

These options apply to all cordova-cli commands.

Option	Description
-d or --verbose	Pipe out more verbose output to your shell. You can also subscribe to `log` and `warn` events if you are consuming `cordova-cli` as a node module by calling `cordova.on('log', function() {})` or `cordova.on('warn', function() {})`.
-v or --version	Print out the version of your `cordova-cli` install.
--no-update-notifier	Will disable updates check. Alternatively set `"optOut": true` in `~/.config/configstore/update-notifier-cordova.json` or set`NO_UPDATE_NOTIFIER` environment variable with any value (see details in update-notifier docs).
--nohooks	Suppress executing hooks (taking RegExp hook patterns as parameters)
--no-telemetry	Disable telemetry collection for the current command.

Platform-specific options

Certain commands have options (`platformOpts`) that are specific to a particular platform. They can be provided to the cordova-cli with a '--' separator that stops the command parsing within the cordova-lib module and passes through rest of the options for platforms to parse.

Examples

- This example demonstrates how cordova-cli can be used to create a project with the `camera` plugin and run it for `android` platform. In particular, platform specific options like `--keystore` can be provided:

```
# Create a cordova project
cordova create myApp com.myCompany.myApp myApp
cd myApp
# Add camera plugin to the project and remember that in config.xml
cordova plugin add cordova-plugin-camera --save
# Add camera plugin to the project and remember that in config.xml. Use npm install to fetch.
cordova plugin add cordova-plugin-camera --save --fetch
# Add android platform to the project and remember that in config.xml
cordova platform add android --save
# Add android platform to the project and remember that in config.xml. Use npm install to fetch.
cordova platform add android --save --fetch
# Check to see if your system is configured for building android platform.
cordova requirements android
# Build the android and emit verbose logs.
cordova build android --verbose
# Run the project on the android platform.
cordova run android
# Build for android platform in release mode with specified signing parameters.
cordova build android --release -- --keystore="..\android.keystore" --storePassword=android --alias=mykey
```

cordova create command

Synopsis

Create the directory structure for the Cordova project in the specified path.

Syntax

```
cordova create path [id [name [config]]] [options]
```

Value	Description
path	Directory which should not already exist. Cordova will create this directory. For more details on the directory structure, see below.
id	*Default*: `io.cordova.hellocordova` Reverse domain-style identifier that maps to `id` attirbute of `widget` element in `config.xml`. This can be changed but there may be code generated using this value, such as Java package names. It is recommended that you select an appropriate value.

Value	Description
name	*Default*: `HelloCordova` Application's display title that maps `name` element in `config.xml` file. This can be changed but there may be code generated using this value, such as Java class names. The default value is `HelloCordova`, but it is recommended that you select an appropriate value.
config	JSON string whose key/values will be included in `<path>`/.cordova/config.json

Options

Option	Description
--template	Use a custom template located locally, in NPM, or GitHub.
--copy-from\	--src
--link-to	Symlink to specified `www` directory without creating a copy.

Directory structure

Cordova CLI works with the following directory structure:

```
myapp/
|-- config.xml
|-- hooks/
|-- merges/
| | |-- android/
| | |-- windows/
| | |-- ios/
|-- www/
|-- platforms/
| |-- android/
| |-- windows/
| |-- ios/
|-- plugins/
  |--cordova-plugin-camera/
```

config.xml

Configures your application and allows you to customize the behavior of your project. See also config.xml reference documentation

www/

Contains the project's web artifacts, such as .html, .css and .js files. As a cordova application developer, most of your code and assets will go here. They will be copied on a `cordova prepare` to each platform's www directory. The www source directory is reproduced within each platform's subdirectory, appearing for example in `platforms/ios/www` or `platforms/android/assets/www`. Because the CLI constantly copies over files from the source www folder, you should only edit these files and not the ones located under the platforms subdirectories. If you use version control software, you should add this source www folder, along with the merges folder, to your version control system.

platforms/

Contains all the source code and build scripts for the platforms that you add to your project.

WARNING: When using the CLI to build your application, you should not edit any files in the /platforms/ directory unless you know what you are doing, or if documentation specifies otherwise. The files in this directory are routinely overwritten when preparing applications for building, or when plugins are re-installed.

plugins/

Any added plugins will be extracted or copied into this directory.

hooks/

This directory may contains scripts used to customize cordova-cli commands. Any scripts you add to these directories will be executed before and after the commands corresponding to the directory name. Useful for integrating your own build systems or integrating with version control systems.

Refer to Hooks Guide for more information.

merges/

Platform-specific web assets (HTML, CSS and JavaScript files) are contained within appropriate subfolders in this directory. These are deployed during a `prepare` to the appropriate native directory. Files placed under `merges/` will override matching files in the `www/` folder for the relevant platform. A quick example, assuming a project structure of:

```
merges/
|-- ios/
| -- app.js
|-- android/
| -- android.js
www/
-- app.js
```

After building the Android and iOS projects, the Android application will contain both `app.js` and `android.js`. However, the iOS application will only contain an `app.js`, and it will be the one from `merges/ios/app.js`, overriding the "common" `app.js` located inside `www/`.

Version control

It is recommended not to check in `platforms/` and `plugins/` directories into version control as they are considered a build artifact. Instead, you should save the platform/plugin spec in the `config.xml` and they will be downloaded when on the machine when `cordova prepare` is invoked.

Examples

- Create a Cordova project in `myapp` directory using the specified ID and display name:

```
cordova create myapp com.mycompany.myteam.myapp MyApp
```

- Create a Cordova project with a symlink to an existing `www` directory. This can be useful if you have a custom build process or existing web assets that you want to use in your Cordova app:

```
cordova create myapp --link-to=../www
```

cordova platform command

Synopsis

Manage cordova platforms - allowing you to add, remove, update, list and check for updates. Running commands to add or remove platforms affects the contents of the project's platforms directory.

Syntax

```
cordova {platform | platforms} [
    add <platform-spec> [...] {--save | link=<path> | --fetch } |
    {remove | rm}  platform [...] {--save | --fetch}|
    {list | ls}  |
    check |
    save |
    update ]
```

Sub-command	Option	Description
add`<platform-spec>` [...]		Add specified platforms
	--save	Save `<platform-spec>` into `config.xml` after installing them using `<engine>` tag
	--link=`<path>`	When `<platform-spec>` is a local path, links the platform library directly instead of making a copy of it (support varies by platform; useful for platform development)

Sub-command	Option	Description
	--fetch	Fetches the platform using `npm install` and stores it into the apps `node_modules` directory
remove`<platform>` [...]		Remove specified platforms
	--save	Delete specified platforms from `config.xml` after removing them
	--fetch	Removes the platform using `npm uninstall` and removes it from the apps `node_modules` directory
update`platform` [...]		Update specified platforms
	--save	Updates the version specified in `config.xml`
	--fetch	Fetches the platform using `npm install` and stores it into the apps `node_modules` directory
list		List all installed and available platforms
check		List platforms which can be updated by `cordova-cli platform update`
save		Save `<platform-spec>` of all platforms added to config.xml

Platform-spec

There are a number of ways to specify a platform:

```
<platform-spec> : platform[@version] | path | url[#commit-ish]
```

Value	Description
platform	Platform name e.g. android, ios, windows etc. to be added to the project. Every release of cordova CLI pins a version for each platform. When no version is specified this version is used to add the platform.
version	Major.minor.patch version specifier using semver

Value	Description
path	Path to a directory or tarball containing a platform
url	URL to a git repository or tarball containing a platform
commit-ish	Commit/tag/branch reference. If none is specified, 'master' is used

Supported Platforms

- Android
- iOS
- Windows (8.1, Phone 8.1, UWP - Windows 10)
- Blackberry10
- Ubuntu
- Browser

Deprecated Platforms

- Amazon-fireos (use Android platform instead)
- WP8 (use Windows platform instead)
- Windows 8.0 (use older versions of cordova)
- Firefox OS (use older versions of cordova)

Examples

- Add pinned version of the `android` and `ios` platform and save the downloaded version to `config.xml`:

```
cordova platform add android ios --save
```

- Add pinned version of the `android` and `ios` platform and save the downloaded version to `config.xml`. Install to the project using `npm install` and store it in the apps `node_modules` directory:

```
cordova platform add android ios --save --fetch
```

- Add `android` platform with semver version ^5.0.0 and save it to `config.xml`:

```
cordova platform add android@^5.0.0 --save
```

- Add platform by cloning the specified git repo and checkout to the `4.0.0` tag:

```
cordova platform add https://github.com/myfork/cordova-android.git#4.0.0
```

- Add platform using a local directory named `android`:

- `cordova platform add ../android`

- Add platform using the specified tarball:

- `cordova platform add ../cordova-android.tgz`

- Remove `android` platform from the project and from `config.xml`:

- `cordova platform rm android --save`

- Remove `android` platform from the project and from `config.xml`. Run `npm uninstall` to remove it from the `node_modules` directory.

- `cordova platform rm android --save --fetch`

- List available and installed platforms with version numbers. This is useful to find version numbers when reporting issues:

- `cordova platform ls`

- Save versions of all platforms currently added to the project to `config.xml`.

- `cordova platform save`

cordova plugin command

Synopsis

Manage project plugins

Syntax

```
cordova {plugin | plugins} [

    add <plugin-spec> [..] {--searchpath=<directory> | --noregistry | --link | --save | --browserify | --force | --fetch} |

    {remove | rm} {<pluginid> | <name>} --save --fetch |

    {list | ls} |

    search [<keyword>] |

    save |

]
```

Sub-command	Option	Description
add`<plugin-spec>` [...]		Add specified plugins
	--searchpath`<directory>`	When looking up plugins by ID, look in this directory and each of its subdirectories before hitting the registry. Multiple search paths can be specified. Use ':' as a separator in `*nix` based systems and ';' for Windows.
	--noregistry	Don't search the registry for plugins.
	--link	When installing from a local path, creates a symbolic link instead of copying files. The extent to which files are linked varies by platform. Useful for plugin development.
	--save	Save the `<plugin-spec>` as part of the `plugin` element into `config.xml`.
	--browserify	Compile plugin JS at build time using browserify instead of runtime.
	--force	*Introduced in version 6.1.* Forces copying source files from the plugin even if the same file already exists in the target directory.

Sub-command	Option	Description
	--fetch	Fetches the plugin using `npm install` and stores it into the apps `node_modules` directory
remove `	` [...]	
	--save	Remove the specified plugin from config.xml
	--fetch	Removes the plugin using `npm uninstall` and removes it from the apps `node_modules` directory
list		List currently installed plugins
search`[<keyword>]`[...]		Search http://plugins.cordova.io for plugins matching the keywords
save		Save `<plugin-spec>` of all plugins currently added to the project

Plugin-spec

There are a number of ways to specify a plugin:

```
<plugin-spec> : [@scope/]pluginID[@version]|directory|url[#commit-ish][:subdir]
```

Value	Description
scope	Scope of plugin published as a scoped npm package
plugin	Plugin id (id of plugin in npm registry or in --searchPath)
version	Major.minor.patch version specifier using semver
directory	Directory containing plugin.xml
url	Url to a git repository containing a plugin.xml

Value	Description
commit-ish	Commit/tag/branch reference. If none is specified, 'master' is used
subdir	Sub-directory to find plugin.xml for the specified plugin. (Doesn't work with `--fetch` option)

Algorithm for resolving plugins

When adding a plugin to a project, the CLI will resolve the plugin based on the following criteria (listed in order of precedence):

1. The `plugin-spec` given in the command (e.g. `cordova plugin add pluginID@version`)
2. The `plugin-spec` saved in `config.xml` (i.e. if the plugin was previously added with `--save`)
3. As of Cordova version 6.1, the latest plugin version published to npm that the current project can support (only applies to plugins that list their Cordova dependencies in their `package.json`)
4. The latest plugin version published to npm

Examples

- Add `cordova-plugin-camera` and `cordova-plugin-file` to the project and save it to `config.xml`. Use `../plugins` directory to search for the plugins.

```
cordova plugin add cordova-plugin-camera cordova-plugin-file --save --searchpath ../plugins
```

- Add `cordova-plugin-camera` with semver version ^2.0.0 and save it to `config.xml`:

```
cordova plugin add cordova-plugin-camera@^2.0.0 --save
```

- Add `cordova-plugin-camera` with semver version ^2.0.0 and `npm install` it. It will be stored in the `node_modules` directory:

```
cordova plugin add cordova-plugin-camera@^2.0.0 --fetch
```

- Clone the specified git repo, checkout to tag `2.1.0`, look for plugin.xml in the `plugin` directory, and add it to the project. Save the`plugin-spec` to `config.xml`:

```
cordova plugin add https://github.com/apache/cordova-plugin-camera.git#2.1.0:plugin --save
```

- Add the plugin from the specified local directory:

```
cordova plugin add ../cordova-plugin-camera
```

- Add the plugin from the specified tarball file:

```
cordova plugin add ../cordova-plugin-camera.tgz --save
```

- Remove the plugin from the project and the `config.xml`:

```
cordova plugin rm camera --save
```

- Remove the plugin from the project and `npm uninstall` it. Removes it from the `node_modules` directory:

```
cordova plugin rm camera --fetch
```

- List all plugins installed in the project:

```
cordova plugin ls
```

cordova prepare command

Synopsis

Transforms config.xml metadata to platform-specific manifest files, copies icons & splashscreens, copies plugin files for specified platforms so that the project is ready to build with each native SDK.

Syntax

```
cordova prepare [<platform> [..]]
     [--browserify | --fetch]
```

Options

Option	Description
`<platform> [..]`	Platform name(s) to prepare. If not specified, all platforms are built.
--browserify	Compile plugin JS at build time using browserify instead of runtime.
--fetch	When restoring plugins or platforms, fetch will `npm install` the missing modules.

cordova compile command

Synopsis

`cordova compile` is a subset of the cordova build command. It only performs the compilation step without doing prepare. It's common to invoke `cordova build` instead of this command - however, this stage is useful to allow extending using hooks.

Syntax

```
cordova build [<platform> [...]]
    [--debug|--release]
    [--device|--emulator|--target=<targetName>]
    [--buildConfig=<configfile>]
    [--browserify]
    [-- <platformOpts>]
```

For detailed documentation see cordova build command docs below.

cordova build command

Synopsis

Shortcut for `cordova prepare` + `cordova compile` for all/the specified platforms. Allows you to build the app for the specified platform.

Syntax

```
cordova build [<platform> [...]]
    [--debug|--release]
    [--device|--emulator]
    [--buildConfig=<configfile>]
    [--browserify]
    [-- <platformOpts>]
```

Option	Description
`<platform> [..]`	Platform name(s) to build. If not specified, all platforms are built.
--debug	Perform a debug build. This typically translates to debug mode for the underlying platform being built.
--release	Perform a release build. This typically translates to release mode for the underlying platform being built.
--device	Build it for a device
--emulator	Build it for an emulator. In particular, the platform architecture might be different for a device Vs emulator.

Option	Description
--buildConfig=<configFile>	Default: build.json in cordova root directory. Use the specified build configuration file. build.json file is used to specify paramaters to customize the app build process esecially related to signing the package.
--browserify	Compile plugin JS at build time using browserify instead of runtime
<platformOpts>	To provide platform specific options, you must include them after -- separator. Review platform guide docs for more details.

Examples

- Build for android and windows platform in debug mode for deployment to device:

```
cordova build android windows --debug --device
```

- Build for android platform in release mode and use the specified build configuration:

```
cordova build android --release --buildConfig=..\myBuildConfig.json
```

- Build for android platform in release mode and pass custom platform options to android build process:

```
cordova build android --release -- --keystore="..\android.keystore" --storePassword=android --alias=mykey
```

cordova run command

Synopsis

Prepares, builds, and deploys app on specified platform devices/emulators. If a device is connected it will be used, unless an eligible emulator is already running.

Syntax

```
cordova run [<platform> [...]]
    [--list | --debug | --release]
    [--noprepare] [--nobuild]
    [--device|--emulator|--target=<targetName>]
    [--buildConfig=<configfile>]
    [--browserify]
    [-- <platformOpts>]
```

Option	Description
`<platform> [..]`	Platform name(s) to run. If not specified, all platforms are run.
--list	Lists available targets. Displays both device and emulator deployment targets unless specified
--debug	Deploy a debug build. This is the default behavior unless `--release` is specified.
--release	Deploy a release build
--noprepare	Skip preparing (available in Cordova v6.2 or later)
--nobuild	Skip building
--device	Deploy to a device
--emulator	Deploy to an emulator
--target	Deploy to a specific target emulator/device. Use `--list` to display target options
--buildConfig=`<configFile>`	Default: build.json in cordova root directory. Use the specified build configuration file. `build.json` file is used to specify paramaters to customize the app build process esecially related to signing the package.
--browserify	Compile plugin JS at build time using browserify instead of runtime
`<platformOpts>`	To provide platform specific options, you must include them after `--` separator. Review platform guide docs for more details.

Examples

- Run a release build of current cordova project on `android` platform emulator named `Nexus_5_API_23_x86`. Use the spcified build configuration when running:

```
cordova run android --release --buildConfig=..\myBuildConfig.json --target=Nexus_5_API_23_x86
```

- Run a debug build of current cordova project on `android` platform using a device or emulator (if no device is connected). Skip doing the build:

```
- cordova run android --nobuild
```

- Run a debug build of current cordova project on an `ios` device:

```
- cordova run ios --device
```

- Enumerate names of all the connected devices and available emulators that can be used to run this app:

```
- cordova run ios --list
```

cordova emulate command

Synopsis

Alias for `cordova run --emulator`. Launches the emulator instead of device. See cordova run command docs for more details.

cordova clean command

Synopsis

Cleans the build artifacts for the specified platform, or all platforms by running platform-specific build cleanup.

Syntax

```
cordova clean [<platform> [...]]
```

Example

- Clean `android` platform build artifiacts:

```
- cordova clean android
```

cordova requirements command

Synopsis

Checks and print out all the requirements for platforms specified (or all platforms added to project if none specified). If all requirements for each platform are met, exits with code 0 otherwise exits with non-zero code.

This can be useful when setting up a machine for building a particular platform.

Syntax

```
cordova requirements android
```

cordova info command

Synopsis

Print out useful information helpful for submitting bug reports and getting help. Creates an info.txt file at the base of your project.

Syntax

```
cordova info
```

cordova serve command

Synopsis

Run a local web server for www/ assets using specified `port` or default of 8000. Access projects at: `http://HOST_IP:PORT/PLATFORM/www`

Syntax

```
cordova serve [port]
```

cordova telemetry command

Synopsis

Turns telemetry collection on or off.

Syntax

```
cordova telemetry [STATE]
```

Option	Description
on	Turn telemetry collection on.
off	Turn telemetry collection off.

Details

A timed prompt asking the user to opt-in or out is displayed the first time cordova is run. It lasts for 30 seconds, after which the user is automatically opted-out if he doesn't provide any answer. In CI environments, the `CI` environment variable can be set, which will prevent the prompt from showing up. Telemetry collection can also be turned off on a single command by using the `--no-telemetry` flag.

Examples

```
cordova telemetry on
cordova telemetry off
cordova build --no-telemetry
```

For details, see our privacy notice: https://cordova.apache.org/privacy

cordova help command

Synopsis

Show syntax summary, or the help for a specific command.

Syntax

```
cordova help [command]
cordova [command] -h
cordova -h [command]
```