# Top-Level Directory

In order to simplify the finding of projects, we are supporting a flat project layout in GIT.

The following is the layout for the CCF Lite repository:

|  |  |
| --- | --- |
| <GIT ROOT>/ | CCF Lite Repo root directory |
| ├── association/ | Device association |
| ├── bridging/ | Bridging |
| ├── discovery/ | Device discovery |
| ├── services/ | Service management |
| ├── security/ | Security API’s to be used by CCF Lite stack on Arduino and Android platforms. |
| ├── controller/  │  │  │  │  │  │  │  │  │ | CCF Lite stack for big devices. It includes the following features:   * Portable controller stack * Android support now * iOS and WinRT are TBD * Library project with a C++ interface * Includes CCF Remoting which interfaces to the CCF stack * Depends on CCF Lite peer project (discovery, association, etc.) |
| ├── android-sdk/  │  │  │  │  │ | CCF Lite SDK for Android. It includes the following features:   * SDK for Android (others to follow) * Test harness for Android (may break this out to separate project) * Consumes controller project |
| └── doc/ | Documentation |

There are changes to the CCF stack to support the CCF Lite remoting.

The following is the layout for the CCF repository that pertain to those changes:

|  |  |
| --- | --- |
| <GIT ROOT>/ | CCF Repo root directory |
| └── portable/ |  |
| ├── sdk/ |  |
| │ └── android/ | SDK additions |
| └── stc/ | Stack changes |

# Project Structure

If you are looking for guidance on project structure consider the following sections.

## Android Projects

Android has its own desired layout. The only thing of note here is that you should ensure that you provide a command line build script that does not require eclipse so that the build machines can handle them.

## C/C++ Projects

If you do not already have a project directory structure, consider using the following structure for C/C++ projects:

|  |  |
| --- | --- |
| project/ |  |
| ├── bin/ | Executable build directory (application or library test executables) |
| ├── lib/ | Library build directory (omitted for projects that build applications) |
| ├── obj/ | GIT ignore this directory. Should be created as part of build process. |
| │ ├── debug/ | Contains the debug version of binary artifacts |
| │ └── release/ | Contains the release version of binary artifacts |
| ├── include/ |  |
| │ └── project/ | Public headers only and should not be dependent on headers in src |
| ├── src/ | C/C++ units plus internal header files |
| ├── doc/ | Doxygen and/or other miscellaneous documentation |
| └── test/ | Test applications (harnesses) |

NOTES:

* This project structure favors Linux/UNIX paradigms.
* Do not commit (gitignore) artifact directory (bin, lib, obj)
* Beware of capitalization in file names. The Windows file system is case insensitive and linux file systems are case sensitive. Changing case and committing will have unpleasant consequences for folks on other OSes.
* Microsoft Visual Studio may require some variation to the above

# GIT

If you are new to git here is the quick set of commands to initialize your project. I keep my repos in ~/Documents/GIT.

* Clone (create a copy of) the repository:

git clone ssh://<user id>@git-amr-2.devtools.intel.com:29418/peg\_ccfl-core

* Change to the new repository directory.
* Create your project directory.
* Copy your code into the project directory.
* Create a .gitignore if necessary
* Add you files

cd <project>

git add \*

* Check your git status

git status

* Commit the code with a descriptive message

git commit -m "First commit"

* Push it back to the repository

git push

Example session:

