## **QLC-6 — Command Line Reporting with xcodebuild & xcresulttool**

### **Objective**

Learn how to extract test results and code coverage data from an Xcode project using Apple’s native command-line tools. This enables inspection and automation of results outside of the Xcode IDE.

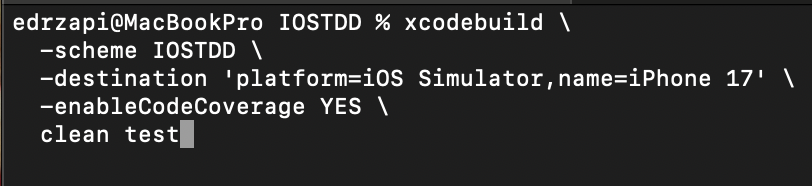
### **Scenario**

You’re working in a team and want to:

* Verify your test suite runs correctly from the command line.
* Extract structured test results.
* Analyze code coverage metrics programmatically.
* Prepare for CI/CD integration with test and coverage reporting.

### **Step 1 — Run Tests with xcodebuild**

Use xcodebuild to run your test suite with code coverage enabled. This will generate a .xcresult bundle containing test and coverage output.



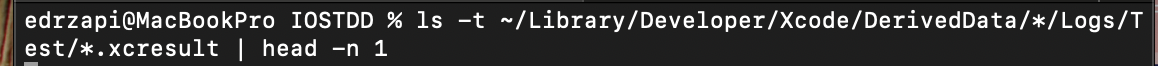
If you experience any problems with signing, try:

* CODE\_SIGN\_IDENTITY="" \
* CODE\_SIGNING\_REQUIRED=NO \
* CODE\_SIGNING\_ALLOWED=NO \

### **Step 2 — Explore Results with xcresulttool**

The .xcresult bundle contains all metadata related to your test run, including test outcomes, diagnostics, and code coverage information. You can extract and inspect this data using xcresulttool, which outputs structured JSON suitable for analysis or transformation.

The following command will locate the most recent .xcresult bundle. You can use the open command to view it in Xcode’s Report Navigator:



While this simply launches the report viewer in Xcode, the key point is that we now have access to this data via the command line. The next step is to capture this information in a shareable format, such as an HTML report, that can be distributed or integrated into automated workflows.

### **Step 3 — Inspect Coverage Info**

Use xccov to view coverage results from the .xcresult bundle. You can inspect coverage:

* At the project or target level
  + xcrun xccov view --report /Users/Library/Developer/Xcode/DerivedData/IOSTDD-etc.xcresult
* Per file
  + xcrun xccov view --file IOSTDD.framework/TaskService.swift /Users/Library/Developer/Xcode/DerivedData/IOSTDD-etc.xcresult
* Per function or line
  + xcrun xccov view --file IOSTDD.framework/TaskService.swift --json /Users/Library/Developer/Xcode/DerivedData/IOSTDD-etc.xcresult

This allows you to verify exactly what code your tests are exercising.

**QLC-6.1 — Group Activity: Shippable Reports with llvm-cov**

### **Objective**

Use llvm-cov to transform raw code coverage data into **readable, sharable reports**, and reflect on gaps in test coverage. This supports better team feedback, PR reviews, and CI transparency.

### **Scenario**

You've generated .xcresult and .profraw/.profdata files using xcodebuild. Now, it's time to **convert this into HTML output**, analyze what was covered, and reflect on what wasn’t.

In small groups, you'll:

* **Run tests to generate coverage data**
* **Produce HTML reports using llvm-cov**
* **Identify and discuss gaps in coverage**
* **Compare JSON vs HTML output**
* **Propose strategies for improving coverage in key areas**

### **Pre-requisites**

Make sure:

* Your project builds and tests with xcodebuild
* You include the flag -enableCodeCoverage YES
* You know your .xcresult location and .profdata path
* You’ve verified llvm works:
  + xcrun llvm-cov --version

### **What to Produce**

Each group should:

* Generate an **HTML report** using llvm-cov show --format=html
* Compare the **HTML** and **JSON** outputs for readability and tooling potential
* Identify **at least one coverage gap** in your current test suite
* Discuss:
  + Why wasn’t that code covered?
  + How would you write a test to close the gap?
  + Would this report help in a real PR/code review?

### **Outcome**

Groups will present:

* A sample HTML report (or screenshots)
* Their chosen gap and how they’d test it
* Comparison notes: HTML vs JSON
* Ideas for CI/CD integration (bonus)

**Hints:**:

xcrun llvm-cov show \

[PATH\_TO\_IOSTDD\_BINARY] \

-instr-profile=[PATH\_TO\_COVERAGE.PROFDATA] \

-format=html \

-output-dir=coverage-html