WorldLink / Nilepoint Build Documentation

**Introduction**

This document will serve to guide the reader on how to compile and install all components for Worldlink Track and all Nilepoint libraries required for its operation.

**Build Environment**

Worldlink Track and the nilepoint component libraries require both Gradle and Maven to be installed and available on the host that is compiling the components. Gralde and Maven are the de facto standards for dependency and build management in Java development.

Gradle information can be found here: <https://gradle.org/> All components requiring gradle have a binary available called “gradlew” that will allow for installation of the component without gradle being available on the machine, but it is suggested that.

Maven information can be found here: <https://maven.apache.org/>.

The nilepoint data exchange requires grails to be installed in order to compile (or use of the gradle wrapper provided)

It is suggested that you use SDKMan to manage these build environments. They can easily installed / managed for many host environments by following the install directions at <http://sdkman.io/install.html>

It is also advisable to install Android Studio in order to build the nilepoint-mobile application and download / install the required android SDKs.

**Components**

The following components are required to be compiled and installed / accessible in order to compile.

**Android Libraries**

1. nilepoint-persist – This android library contains the base persistence logic and the domain model for the WorldLink track / nilepoint-mobile application. It depends on nilepoint-networking
2. nilepoint-formbuilder- This android library provides a dynamic form builder activity so forms can easily It depends on nilepoint-networking and nilepoint-persist

**Java Libraries**

1. nilepoint-domain-model- This project provides the underlying JPA domain model that allows for a unified object model for all implementations of the nilepoint domain.
2. nilepoint-fh-bridge – This project / component handles the bridging between the nilepoint domain model / and the fh domain model. It also is where all FH WorldLink API calls are defined. It depends on the nilepoint-domain-model
3. nilepoint-networking – This project provides classes that interact with the networking resources to all of the other components. This includes most of the DTN classes (for TCP / UDP support), socket receivers, AMQP classes, as well as serialization classes used in network transmission. It depends on the nilepoint domain-model

**Grails Project**

1. nilepoint-exchange – This is a grails 3 project (Spring Boot-based web framework) that provides the exchange services.

**Android Project**

1. nilepoint-mobile – This android project can be compiled once the Java Libraries and Android Libraries are installed on the system compiling the android project.

**Build Steps**

In order to install the above components clone them in git and build them in the following order. It is advisable to build the maven libraries with the -DskipTests flag as the integration tests may take many minutes to run:

1. nilepoint-domain-model (Maven, mvn -DskipTests install)
2. nilepoint-fh-bridge (Maven, mvn -DskipTests install)
3. nilepoint-networking (Maven, mvn -DskipTests install)
4. nilepoint-persist (Gradle, gradle install)
5. nilepoiint-formbuilder (Gradle, gradle install)
6. nilepoint-mobile (Gradle, gradle build)
7. nilepoint-exchange (Gradle, gradle build)

**Build Results**

Both the java and android libraries will result in jar files that will be installed in your maven / gradle package repositories.

The exchange build process will create a web application archive (WAR). The war can be found in build/libs/nilepoint-exchange-0.2.war of the project root . This WAR can be ran as a stand-alone application as well on any system with java installed by running “java -jar nilepoint-exchange-0.2.war“ or deployed in a standard web container (Tomcat, JBoss, Glassfish)

The nilepoint-mobile build process will result in two APKs, a debug APK and a unsigned release APK. The resulting APK will be found in app/build/outputs/apk/app-release-unsigned.apk and app/build/outputs/apk/app-debug.apk in the project root directory.