# Build with Maven

Git Repository:

Git is used as the central repository for all the IIB artefacts. Git Plugin can be used in IIB toolkit to connect to the local git repositories.

IIB-Maven Plugin:

Maven is used to deal with all the dependencies in the Integration projects. The plugin is based around the central concept of a build lifecycle. This plugin has several build lifecycles: The main iib-bar lifecycle automates IIB9 BAR file creation and deployment to a Broker Node. The iib-bar lifecycle will perform most day-to-day operations.

## The iib-bar Build Lifecycle

The iib-bar lifecycle comprises of the following phases:

* clean - performs the standard cleanup and removal of a project's target directory
* initialize - performs any additional file cleanup
* generate-resources - copies jar dependencies into project and downloads any common library dependencies into workspace
* process-resources - validates that each project in a workspace has correct format (project name = directory name)
* compile - creates a broker archive using mqsicreatebar command (recommended) or using package bar.
* test-compile - compiles unit and integration tests in the src/test/java directory.
* process-classes - applies bar overrides using properties files found in resources folders and validates classloader approach
* pre-integration-test - deploys a bar file to a configured broker and execution server
* integration-test - executes integration tests against the just-deployed bar on the configured broker
* verify - evaluates integration test results and stops build in event of failure(s)
* deploy - done in an integration or release environment, copies the final bar to the remote repository for sharing with other developers and projects.

If you call a build phase in this plug-in, it will execute not only that build phase, but also every build phase prior to the called build phase. Just like any other Maven plug-in. Thus, doing

**mvn compile**

will execute every life cycle phase before it (initialize, generate-resources, process-resources), before executing compile.

This plug-in can "mavenize" an entire IIB9 workspace in a few seconds.  IIB9 projects get 'mavenized' by adding a pom.xml file in each project's root directory. And the pom.xml files contain maven dependency references that mirror IIB9 project dependencies.  Also, mavenizing an IIB9 application project includes adding directories such as "src\main\resources" and "src\test\resources".  These directories facilititate bar overrides and bar deployments with property files and broker files respectively. The mavenize goal in the IIB9 Maven plug-in enables these pom.xml files to be automatically generated with a working configuration.

## Application Configuration Parameters

The pom.xml file's configuration parameters are key to executing the iib-bar lifecycle of the IIB9 Maven plugin. These parameters control the behaviour of the iib-bar lifecycle.

* workspace - the location of the IIB9 workspace directory
* initialDeletes - files to delete/clean from a project ; the filenames can be absolute or follow the ant file patterns
* unpackIibDependenciesIntoWorkspace - indicates whether library projects should be downloaded and unpacked into the workspace
* copyDependentJarsLocation - location where dependent jars are downloaded to; defaults to project's base directory
* createOrPackageBar - value of 'create' launches mqsicreatebar whereas 'package' packages bar with uncompiled resources
* pathToMqsiProfileScript - the path to the local mqsiprofile script
* failOnInvalidProperties - indicates whether the build should fail if invalid bar override properties are found in a property file which is not a default property for an non-overridden bar file; During the "process-classes" phase, if a file, like xyz.properties, is found in the src/main/resources directory or the src/test/resources, then the file is used to generate an overridden bar file named xyz.bar.
* useClassloaders - indicates whether class loaders are used; defaults to "false"
* failOnInvalidClassloader - indicates whether build fails if useClassLoader entry differs in overridden property(ies) file(s)
* brokerFileName> - the broker file that containing the connection information to use to deploy a bar file to it
* barFileName - the bar file to deploy to the designated broker.
* completeDeployment - indicates whether the bar file deployment should be a full/complete deployment or a partial deployment
* timeoutSecs - the timeout in seconds to wait on a bar file deployment
* integrationServerName - the name of the integration server to deploy a bar file to
* mqsiPrefixCommands - comma-separated list of commands that will be issued to the underlying operating system before launching the mqsi\* command
* mqsiCreateBarReplacementCommand - an advanced parameter that allows you to replace the generated mqsiCreateBar command arguments with your 'tweaked' arguments
* mqsiCreateBarCompileOnlyReplacementCommand - an advanced parameter that allows you to replace the generated mqsiCreateBar -compileOnly command arguments with your 'tweaked' arguments
* mqsiDeployReplacementCommand - an advanced parameter that allows you to replace the generated mqsideploy command arguments with your 'tweaked' arguments

## Steps to set-up the build environment:

1. Install Maven (Do this once)
2. Update Maven settings (Do this once)

The settings need to be overridden if you need to upload artifacts to a maven distribution repository

1. Install the plugin (Do this once)

~/Handover/iib-maven-plugins/iib-maven-plugin-9.0

1. Install the IBM ConfigManagerProxy.jar in your local repository (Do this once)

In the same command prompt window, install a copy of the IBM ConfigManagerProxy.jar in your local repository with the command:

mvn install:install-file -DgroupId=com.ibm.etools.mft.config -DartifactId=ConfigManagerProxy -Dversion=9.0.300.v20150305-1357 -Dpackaging=jar -Dfile="C:\Program Files\IBM\MQSI\9.0.0.1\classes\ConfigManagerProxy.jar"

1. Install the Plug-In in your local repository (Do this once)
2. In the same command prompt window, install the copy of this plugin in your local repository with the command:

mvn install:install-file -Dfile=iib-maven-plugin-9.0-SNAPSHOT.jar -DgroupId=ch.sbb.maven.plugins -DartifactId=iib-maven-plugin -Dversion=9.0-SNAPSHOT -Dpackaging=jar

1. Add POM files to the workspace (Do this per IIB9 workspace)
2. In the IIB toolkit , create a configuration to run the POM file.

Go to External Tools Configurations: –

Name -> Provide a name to the build

Location-> This must be the Maven installation location pointing to mvn.cmd . For ex., C:\Maven\apache-maven-3.3.9\bin\mvn.cmd

Working Directory -> Point to the application that needs to be built. Note: the POM file must reside here

Arguments -> Command to be provided to Maven for build. For ex., ‘compile’ creates the bar files and does all the overrides based on the \*.properties files.