Maven Core Cleanup

- General goals
- General backward compatibility notes
- Empower Maven Core
  - Ingest MAT into Maven
  - Questions/Problems
- Sort out RepositorySystem/MavenRepositorySystem
- Transition period (with Maven Artifact Transfer)
  - Questions/Problems
- Cut off step (from Maven Artifact Transfer)
  - Questions/Problems
- Drop maven-compat module
  - Questions/Problems

This page discusses Maven Core (or better, "Maven as whole") cleanup, as currently we are stuck on a dead end, where (mostly) legacy prevents us to move forward (innovate, progress).

Notes: This is a DRAFT. Also, this document is initially composed only as "food for though"/RFC, so please consider all sentences are like beginning with "IMHO....".

General goals

Currently we agreed to up plugins to depend on Maven 3.1.1 (released 2013-09-17) as "baseline". This means that we skip the plagued Maven 3.0 line (and org.sonatype resolver package). Implication of this is that main role of MAT (to bridge between org.sonatype and org.eclipse packages is now not needed anymore.

Our general goals (not in execution order) are following:

- get rid of maven-compat (Maven2 support) module in Maven project
- get rid of maven-artifact-transfer (MAT) "artificial" dependency ("artificial" as it provides two things: bridges between org.sonatype and org.eclipse Aether packages. This is not needed anymore, as our baseline is Maven 3.1.1 for LTS. Also, this artifact provides functionality that Maven Core should provide in the first place)
- decide fate of very much related two important components in Maven: maven-core:RepositorySystem (is an interface, implementation of this interface is in maven-compat) and maven-core:MavenRepositorySystem (is a budged class with lot of statics).
- plugins SHOULD NOT use any core dependency (meaning Resolver, Wagon) directly, they should use Maven Core provided services instead (and Maven Core should cover all the required functionalities).

General backward compatibility notes

This section is about Maven and "our" Maven Plugins (residing in org.apache.maven group ID, maybe also org.codehaus.mojo).

Maven 3.x line supported Maven 3 and Maven 2 compatible plugins. Maven 4.x in same sense, will support Maven 4 and Maven 3 compatible plugins, but not Maven 2 compatible plugins.

Empower Maven Core

As first step, we should provide all functionalities to Maven Core that lacks today (and are provided by MAT as a hack) but "shielding" underlying implementations from leaks (to not repeat the org.sonatype/org.eclipse fiasco). In short, plugins SHOULD NOT use Aether directly (Aether managed by Maven). Plugin could still "roll their own container" and do whatever they want if they have to, but goal is to extend Maven Core to cover most of the needs of plugins (artifact: resolve/install/deploy, dependencies: collect/resolve, project: install/deploy). This could be implemented in several ways, but given of spread of MAT, less resources would be needed (so to say, later transition would be simplest), if we'd reuse MAP API.

Ingest MAT into Maven

Ingest and not "merge" as this requires package changes (to place things where they belongs), but also due the fact that MAT itself introduced breaking package changes in version 0.10.0 of it.

Questions/Problems

- where should MAT API go?
- what package layout?
- what should happen to filters, that are NOT in core but in maven-common-artifact-filters? (while we do not want to expose Resolver)

Sort out RepositorySystem/MavenRepositorySystem

These two components should most probably merged into one and properly split (iface + impl).

Questions/Problems

- we may want to do only move RepositorySystem implementation from maven-compat for start and leave them as-is if there is a lot of usage.
• maybe "redirect" most of logic to ingested MAT API?

Transition period (with Maven Artifact Transfer)

In existing plugins (or everything) that consumes MAP we should up MAT dependency version to latest released (0.13.1 as of now) to get rid of package changes happened in 0.10.0.

Once previous step is done ("Empower Maven Core"), in current master of MAT (refactored into provider modules, unreleased still) we should introduce new "maven 4" provider that would simply delegate calls to "empowered Maven Core".

Questions/Problems

• this step depends on previous step (as new MAT provider needs ingested MAT API in Maven Core)

Cut off step (from Maven Artifact Transfer)

Once Maven Core is "empowered" and stable (4.0.0 release?), plugins using MAT could be transitioned to use Maven API for things they pulled in MAT for and drop MAT from their dependencies. At the end, no plugin should use MAT and MAT will cease to exists.

This implies that these plugins will become "Maven4 compatible" and cut off the long history trail they all pulled (this is similar if not same step as we did with 3.x lines of core plugins).

Questions/Problems

• we must cut off history that lead us to the point where we are today
• for "legacy" support we have 3.8.x line, and as discussed we can even do 3.9.x line (upping Java baseline) as Java7 as baseline in 2021. is insane.

Drop maven-compat module

Once this happened, and RepositorySystem/MavenRepositorySystem is cleaned up, we should be able to drop Maven2 support from Maven4.

Questions/Problems

• this step depends on all steps above