Centralized Dependency Management

Build a central place for maven dependency management of all 3rd party libraries for all magnolia modules (bundled, unbundled, community, enterprise, etc).

Purpose

The Problem

We manage most dependencies in magnolia main & ui. We import main and/or ui in other modules to reuse the dependency management.
This turned out to be not the best idea because of the following reasons:

- We have the policy to alway depend on the oldest main version possible. Which means we tend to depend always on outdated 3rd-party libs in modules.
- We want to reduce the footprint of main and because of we managed some dependencies in other modules like UI specific libs in UI, REST libraries in rest, etc. This leads to potential divergent versions (like we actually had with the jackson library)

Goals

The most agreed on solution is to have a specific project/pom where we will just define the dependency management. This project will then be imported in all modules.

Use Cases

> AKA: User Stories
> List concrete ways in which this feature will be used.

Proposal

Concept

Special BOM project which resides at the same place like the super-poms.

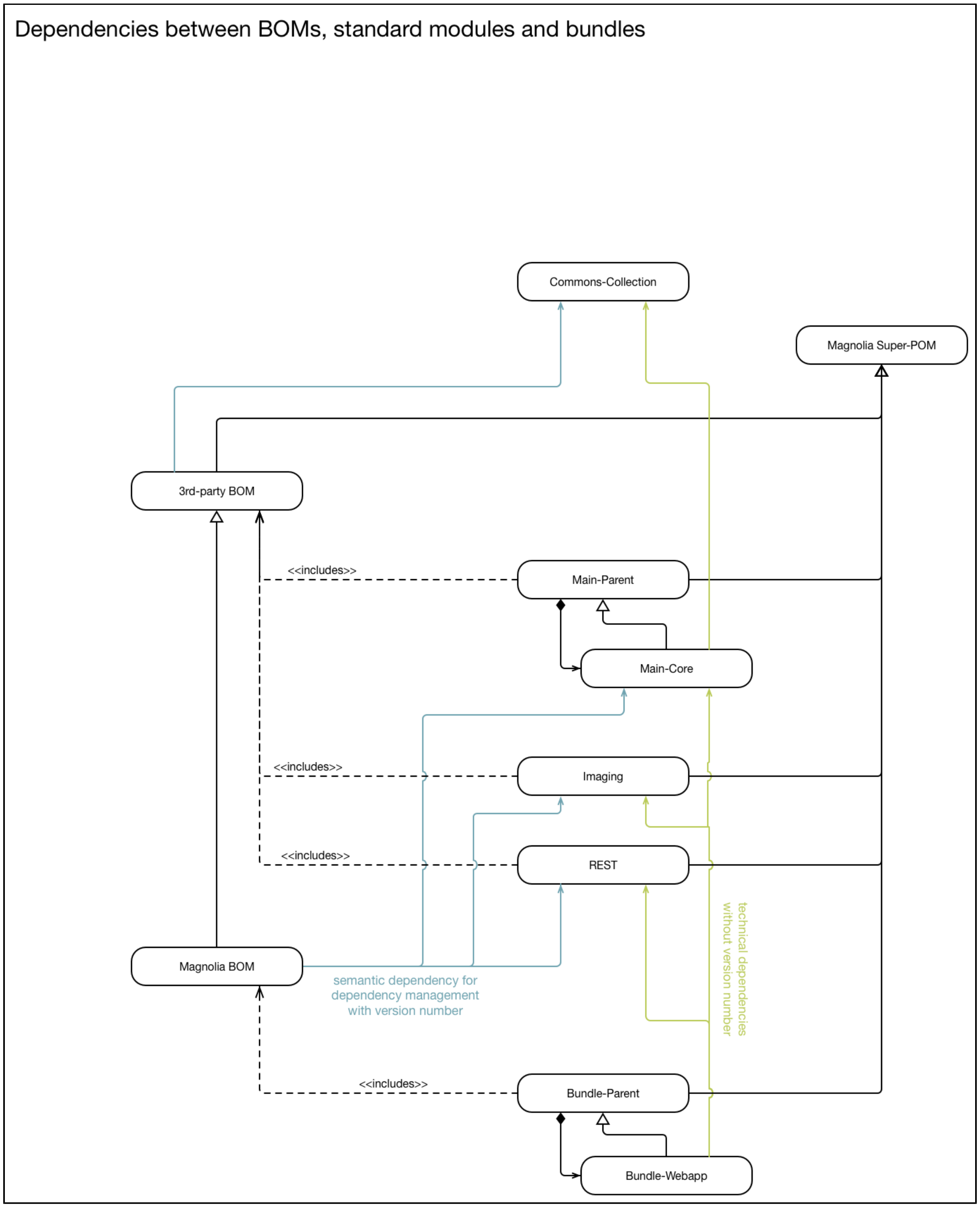
Would look like:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd ">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>info.magnolia.maven.poms</groupId>
    <artifactId>magnolia-parent-pom-community</artifactId>
    <version>33</version>
  </parent>

  <groupId>info.magnolia.boms</groupId>
  <artifactId>magnolia-external-dependencies</artifactId>
  <packaging>pom</packaging>
  <name>Magnolia 3rd-party dependencies</name>
  <version>5.6-SNAPSHOT</version>

  <!--
    Bill of material for every magnolia module to be used.
  -->

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <scmTagPrefix>magnolia-bom</scmTagPrefix>
  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.0.1</version>
        <scope>provided</scope>
      </dependency>
      <dependency>
        <groupId>javax.jcr</groupId>
        <artifactId>jcr</artifactId>
        <version>2.0</version>
      </dependency>
      <dependency>
        <groupId>org.apache.jackrabbit</groupId>
        <artifactId>jackrabbit-core</artifactId>
        <version>2.12.4</version>
        <exclusions>
          <exclusion>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
          </exclusion>
          <!-- maybe this will have to be un-excluded if we remove our explicit usage of it in magnolia-core -->
          <exclusion>
            <groupId>xerces</groupId>
            <artifactId>xercesImpl</artifactId>
          </exclusion>
        </exclusions>
      </dependency>
      <dependency>
        <groupId>org.apache.jackrabbit</groupId>
        <artifactId>jackrabbit-jcr-commons</artifactId>
        <version>2.12.4</version>
      </dependency>

      ...
    </dependencies>
  </dependencyManagement>

</project>
```

And be used like this:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd ">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>info.magnolia.maven.poms</groupId>
    <artifactId>magnolia-parent-pom-community</artifactId>
    <version>33</version>
  </parent>
  <groupId>info.magnolia.bundle</groupId>
  <artifactId>magnolia-bundle-parent</artifactId>
  <version>5.6-SNAPSHOT</version>


...


  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>info.magnolia.boms</groupId>
        <artifactId>magnolia-external-dependencies</artifactId>
        <version>5.6-SNAPSHOT</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>

...

</project>
```

Simplified overview of dependencies between super-pom, bom, modules and bundles:



Decisions

We made the following design decisions:

	Design Decision	Reasoning

1	We want to define the dependency management in a separate BOM project and not in one of the super-poms	We want to try to keep parent poms in sync over all release streams. Both 5.4 and 5.5 should use the most current magnolia-parent-pom (currently 33). We can basically do that for build plugins but we can't do that for 3rd party libraries.
2	We bump BOM Version together with Magnolia Release even if it has no changes	Its a simple POM project/file and causes no overhead even if we do a lot of version bumps. One reason for creating BOMs is to get a grip on the version hell in magnolia. If the BOM version now diverges from the Magnolia Release version its more complex and harder to follow for us and customers.
3	The 3rd-party BOM resides in a separate repository in the build project.	This way its easier to maintain. Easier to work with the different BOMs for the release streams (5.4.x, 5.5.x, 5.6.x ...) Reasons for packing it with the Super POMs were: • Not yet another repository to maintain • Would force to make release process of poms project accessible for everybody
4	The Magnolia BOM resides in the same repository as the 3rd-party BOM	Those two projects have the exact same release cycle It has actually the same release cycle like ce & ee & ce-packs & ee-packs. But because those are in separate repositories the Magnolia BOM can not reside with them.
5	The Magnolia BOM becomes the driver for a Magnolia Release	Because the Magnolia BOM includes versions from bundled and unbundled modules it can become the single source of truth for the releases instead of the JIRA changelog filter.
6	We backport the 3rd-party BOM to the 5.5 release stream starting with 5.5.7.	We have several modules which are atm backwards compatible with 5.5 with their current release. Additional one of them REST even manages dependencies. Because REST was one of the main reasons why we want to have a BOM (Version divergens of jackson) in the first place, we have to introduce BOMs for 5.5.x as well. The versions of 3rd-party libraries is atm we made that decision (Sept. 11, 2017) identical and because the BOM is transparent to end users, the backport should have no impact on customers.

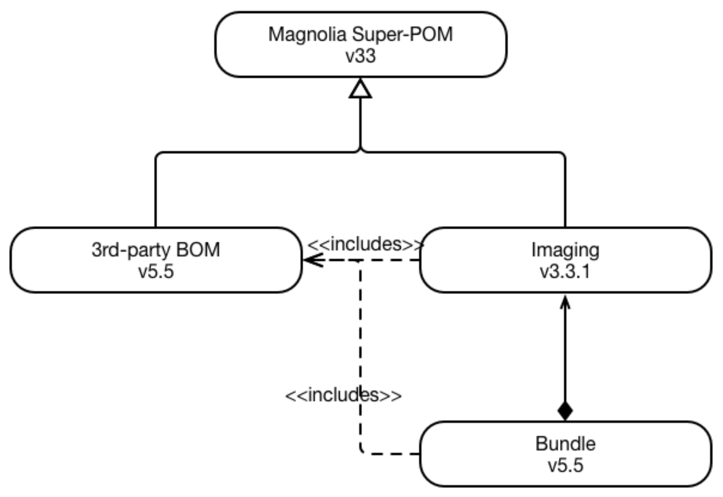
Current places where we maintain 3rd party library versions

- platform/main (magnolia-project)
    - used ... everywhere
- platform/ui (magnolia-ui-project)
    - used .. most places
- module/rest (magnolia-rest-parent)
    - used in cache
- platform/ce (for test libraries)

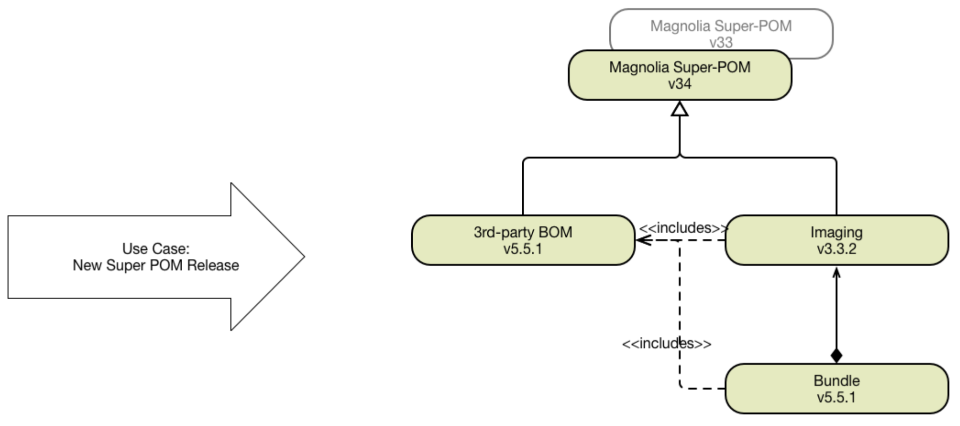
Scenarios

The following scenarios high-light how specific use cases would look like with the new central dependency management in place.

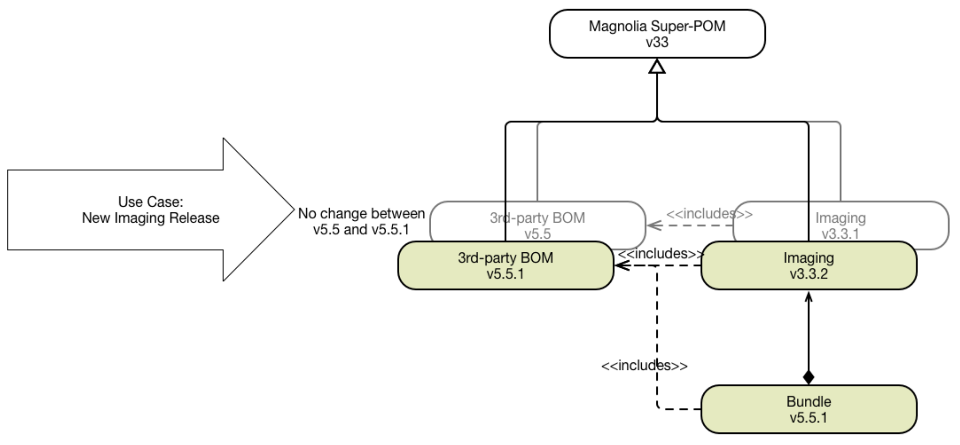
Simplified software structure

 Magnolia Super-POM v33 3rd-party BOM v5.5 <<includes>> Imaging v3.3.1 <<includes>> Bundle v5.5	For the following scenarios we use this simplified setup.

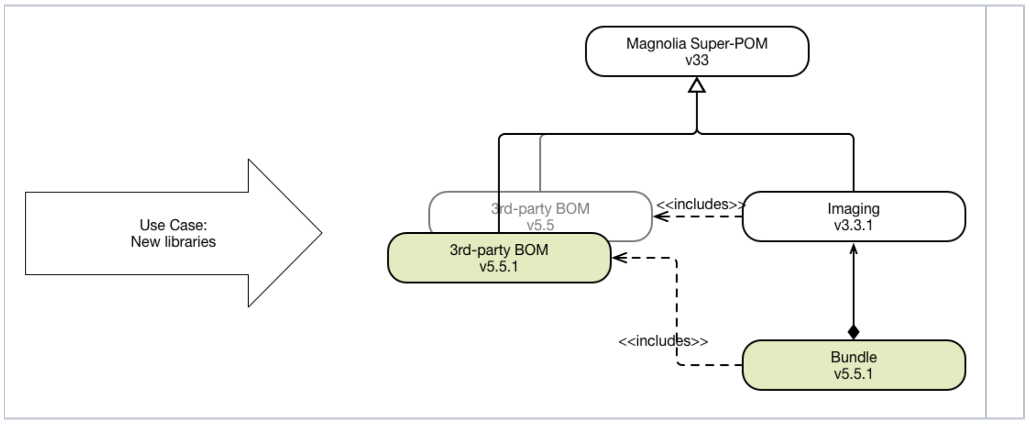
Release new version of the super POMs



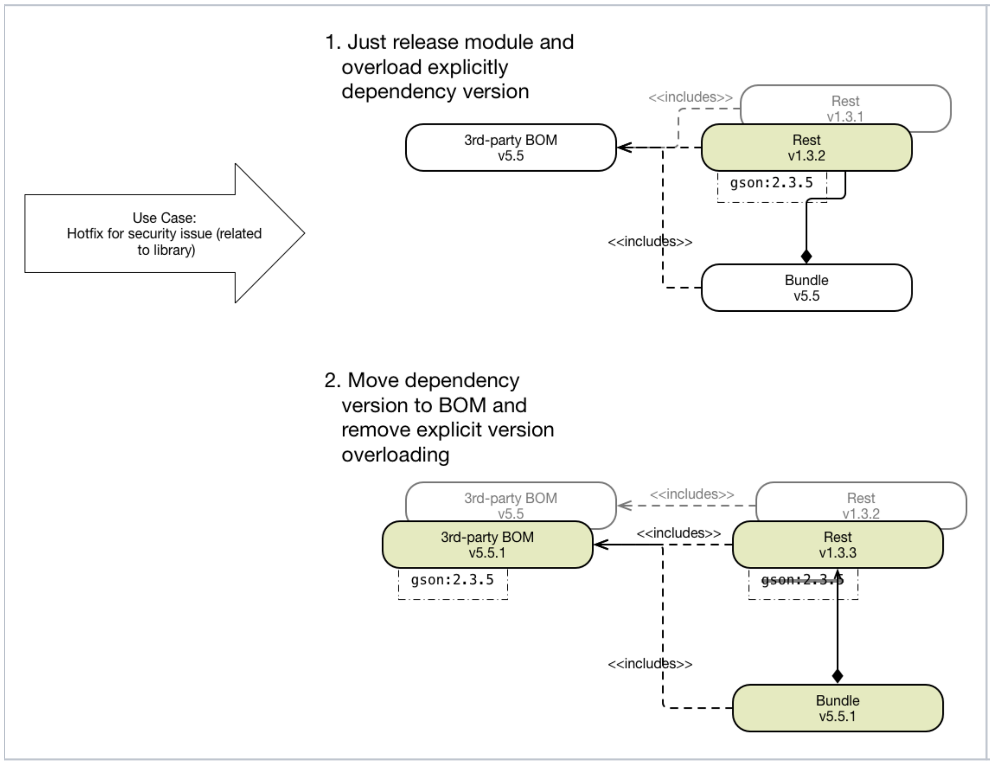
Release of new magnolia module version



Library update without changes in modules



Security Release which includes 3rd party lib



If we want to make a module hotfix release, because of a security issue for example, we would temporarily manage the dependency in that specific module.

Later we move it to the 3rd-party BOM and remove the explicit management in the module. We should have tooling to validate that no versions are managed in individual modules.

Issue with patched version: customer potentially still imports bom 5.5 in his bundle and overrides version from rest.

Implementation

Following steps:

1. Move main dependencies without changes to `magnolia-external-dependencies`
2. Use `magnolia-external-dependencies` in main
3. Move ui dependencies without changes to `magnolia-external-dependencies`
4. Use `magnolia-external-dependencies` in UI
5. ... repeat those steps for any other place dependencies are managed atm like Rest or CE
6. Reorganize dependencies in `magnolia-external-dependencies`

Thoughts

We should probably do the same for our own artifacts as well like indirectly requested by MAGNOLIA-6343

Open Questions

Question	Answer	Reasoning
~~For master branch or 5.5, 5.4 as well?~~	~~Only next major~~	~~clear cut~~
	~~Next maintenance~~	~~easier for us~~
Which dependencies do we manage? • For all dependencies or only some (if second: what is the rule to decide and how will the rest be managed) • Which scopes do we manage?	All dependencies, all scopes	Simplicity & atm, no reason why we can't
What's the release cycle of this pom?	Same as Magnolia Bundle Releases, Yes this includes *empty* releases (see above) *Jan 12/04/18: Contrary to previous agreement and line of thinking, we found out (by accident) that* ***it is not necessary to keep version numbers in sync****. One can simply rely on Magnolia bundle version and import dependency management section of the pom as shown in documentation.*	Simplicity
Does it make sense to test different depMan sections (via profiles) to check compatibility?	---	
Validate whether modules still work with base line (when specifying dep. to e.g. 5.4.x)	*Let's burn that bridge when we get there*	
How does the standard release work like related to the changes from this concept?! Do we need/want some kind of release aggregator? (BOM)	3rd-party BOM becomes first in line in release process.	
How do we version the BOM?	It should be somehow in sync with the main magnolia versions	
Do we wanna be always on the latest version of the BOM? (similar question goes to all dependencies)	Is tied to the general question of *always on current*. For the moment we keep the same policy as with dep on main.	
How can we find unused dependencies? (for cleanup)	---	

Specific for Magnolia BOM

- Can we have ee (non accessible) dependencies managed in the BOM? Can community users still use it?
- Do we have dependencies we have to manage to the bundles?
    - And if so (like webapps) how do we handle them?

Glossary

Name	Definition
Magnolia Release	The full stack release or the bundle release.
Super-POM	Actually a set of parent poms.
BOM	Bill of material. A single place to define material of in this case dependencies we can use.
3rd-party BOM	BOM for all external dependencies.
Magnolia BOM	BOM for **all** magnolia modules

Etcetera

> Possible Improvements

> Discarded Proposals

> Discussion

> References