

# Artifact-Coordinate Expression Transformation

## Expression Transformation in Artifact Coordinate Values within the POM

### Relevant JIRA Issues

#### Main

- [MNG-4167](#)
- [MNG-4140](#)
- [MNG-3057](#)

#### Similar

- [MNG-2971](#)
- [MNG-2446](#)
- [MNG-2412](#)

### Use cases and correct behavior

#### coordinate expression use cases

##### jar packaging

The main concern here is the way transitive dependencies will be resolved. dependency POMs will be interpolated during the consumer's build, which could result in invalid artifact references, or at least changed references from how the jar dependency was built.

**NOTE:** This is a similar scenario to artifacts that don't specify their version in a locked-down range...i.e. they are suggestions, not absolute requirements.

##### POM packaging / parent POM dependencies

The main concern here is maintaining dynamic artifact coordinates in dependencyManagement, plugins, etc. so that property references can be overridden by child POMs. For example:

- specifying `<mavenVersion>` property for a suite of maven-related deps in dependencyManagement, which specify: `<version>${mavenVersion}</version>`
- the parent POM may supply a basic default of: `<mavenVersion>2.0.9</mavenVersion>`
- If a child POM overrides with: `<mavenVersion>2.1.0</mavenVersion>`, it should be able to make use of all the maven artifacts specified in the dependencyManagement of the parent with the newly-overridden version.

The pom packaging also shares the same concerns as jar packaging WRT transitive dependencies

##### envvar and other user-specific expressions

These will evaluate to the local value. For instance:

- `os.name`
- `java.version`
- `user.dir`

- user.home
- etc.

If a POM uses these in expressions for artifact coordinates, it may result in artifact references that only resolve on certain environments. For example, if built using JDK 1.6 and no corresponding artifact with '1.6' in its coordinate exists, but one does for '1.5' and '1.4', then the build will fail on that 1.6 environment.

## properties in profiles

These may change the artifact coordinate(s) according to which profile is active, or whether the default properties (in the POM main section) are used

When loading POMs from the repository:

- settings profiles are NOT activated here
- profiles.xml profiles are NOT activated here
- profiles specified in -P cli options are NOT activated here
- only those that trigger based on the following criteria will be activated:
  - system properties
  - cli-specified properties
  - other non-property activators

## plugin requirements for POM information

### release plugin

- invoked directly from cli, outside of any lifecycle
  - invokes different lifecycle builds on the project, but currently uses separate java processes to do so
- MUST modify the original pom.xml file as it exists on disk, WITHOUT ANY OTHER MANIPULATIONS
- must NOT have new files added that are then referenced from project.file
  - this means that transforming the original POM file and writing it to a new location, which is then set on project.file, WILL NOT WORK

### enforcer plugin

- normally bound to the lifecycle
- MAY require access to unaltered, original POM in order to execute rules

### gpg plugin

- normally bound to the lifecycle
- requires access to the POM file that will eventually be installed or deployed. This file MUST NOT be changed after GPG runs.

### shade plugin

- normally bound to the lifecycle
- requires access to project.originalModel, which MUST reflect the information in the POM file that will be installed or deployed.
  - this is necessary for the shade plugin to be able to generate a dependency-reduced POM.

## Implementation strategies

### 2.0.10

Expressions in artifact coordinates are ignored. Users have plenty of rope with which to hang themselves

## 2.1.0

This was the first attempt to clean up the coordinate values in POMs before installing/deploying. Obviously, we didn't really understand the scope of the problem at this point.

- attempted to resolve artifact versions to concrete terms during install/deploy process
- implemented as an ArtifactTransformation, just like snapshot handling.
  - incidentally, snapshot handling violates the requirements for the shade plugin AND the gpg plugin.

### Problems:

- also modifies plugin configurations where the element '<version>' is used
- fails to account for activated profiles that may supply/change interpolation values
- only accounts for artifact versions, not artifactId, groupId, classifier, or type
- modifies the POM after it has been signed by GPG, making the signature worthless
- modifies the POM without reflecting the new information in the originalModel for the shade plugin to use
  - transformation happens too late for shade plugin anyway, though

This breaks legitimate use cases for expressions in artifact coordinates, like those detailed in the 'pom' and 'jar' packaging scenarios, above

## 2.2.0-current

In the latest attempt to resolve artifact coordinate expressions, the solution from 2.1.0 has been:

- generalized to look at all artifact fields (groupId, artifactId, version, classifier, type)
- moved into DefaultMavenProjectBuilder, to be run just before a project is returned to the build process
  - this makes the transformed artifact coordinate information available in POM-file form to all plugins in the build process, such as GPG
  - it also means that the POM transformation happens AT ALL TIMES

### Problems:

- release plugin tries to add the transformed version of the POM as a new file to SCM, since that's the POM file referenced from the project instance
- shade plugin still cannot gain access to transformed information, since it's not reflected in project.originalModel
  - since the transformed information isn't original, this may not be appropriate anyway, though...

This breaks legitimate use cases for expressions in artifact coordinates, like those detailed in the 'pom' and 'jar' packaging scenarios, above

## 2.2.0-final

For this release, we're probably going to have to reverse course and remove all POM transformation code. We need a more comprehensive design review, and much more planning on how to introduce this sort of feature without breaking the use cases above

- legitimate/safe coordinate expressions should be supported
- any transformation must be reflected in all locations that plugins look for the information
  - either that, or the plugins must migrate to any new api we put in place to support coordinate transformation