



# OSGi Release Versioning Strategies

Robert Dunne (Paremus)



# Antediluvian versioning

In the beginning: Versioning not used at build or runtime

Next: Maven, Ivy introduce build time versioning

Module to Module



# Enter OSGi

Define and export API packages - think in terms of APIs

Versions and Version ranges checked at runtime

Need a versioning policy

Use tools that support package level dependencies at build time.

**Bnd, Sigil, PDE, Bundlor?**

Deploy using a resolver



# Versioning Policy

Pick a sensible one: Version = major.minor.micro.qualifier

major = breaking change

minor = backwards compatible

micro = bug fix or other essentially invisible change

qualifier = extra info

No way to express this policy programatically



# Development version ordering

These are all in version-range [3,4)

3.0.0.SNAPSHOT

3.0.0.RC1

3.0.0.RC2

3.0.0

Lexical order != Logical order

3.0.0, 3.0.0.RC1, 3.0.0.RC2, 3.0.0.SNAPSHOT



# Improved lexical ordering

3.0.0.SNAPSHOT

3.0.0.V4567-RC1

3.0.0.V4678-RC2

3.0.0.V4679-RELEASE

In order - right this time (still partly lexical fudge)

3.0.0.SNAPSHOT, 3.0.0.V4567-RC1, 3.0.0.V4678-RC2, 3.0.0.V4679-RELEASE



# Build Status Attributes

Package-Export: name, version, **attributes (some mandatory)**

3.0.0.V4567-RC1 (**build-status=RC1**)

Mandatory?



# Improved control

Resolver Policy - restrict particular installs based on attributes and other info.

OBR, Nimble(Newton), P2(Eclipse)

Package Hooks? - add attributes at install time

Repository Hierarchy?



# Conclusion and Desiderata

OSGi means we have to take Versioning Policy seriously

Help clients by providing attributes and ensuring the correct lexical ordering

Use tools and resolvers that make versioning and other aspects of modularization easier to manage

Want to express package versioning policy in code (attributes again?)

Package Hooks please