Extending Maven 2.0 Dependencies

Synopsis

The current version scheme for Maven 2.0 of

```
<major>.<minor>.<revision>([-<qualifier>]|[-<build>])
```

is suited for versions up to and including a project's release, but additional identifiers are recommend for vendor identifiers and post-release versions.

Needs

Vendor Tag

Oftentimes to use an open source project within a corporate application requires that the corporation has access to the source code of a given project release and that the library is reproducible from this source code. Many development organizations have the compliance requirement that these libraries can be identified as being packaged by that organization or a known integrator.

Existing packaging mechanisms such as RPM provide the capability of a vendor tag.

We propose that Maven dependencies be able to be specified based on the identification of a vendor.

Post release

In addition to the vendor tagging requirement, a library can often go through multiple patches before a new release is published by the original project team. Even in the case where the release lifecycle is short, it is inevitable that production issues will occur between releases that need to be addressed, and a library incorporating a fix needs to be issued. These fixes can be aggregated and the source rebuilt by the user of the library or third party support provider. The resulting library then needs identification as being of a certain patch level or 'service pack'.

To make this concrete in terms of a Maven user let us look at some dependency cases on the Apache Commons Logging project:

- (exactly 3.1)
- (3.1 built by us)
- (3.1 built by us and sp2)
- (3.1 built by us and at least sp2)

Current Design

- Currently, the `<major>.<minor>.<revision>` component of a version must be numeric; if non-numeric characters are used to annotate this (e.g. `3.1.0myvendor-01`), the whole string is viewed as a qualifier.
- `<qualifier>` and `<build>` are exclusive. Hence one can't specify `[3.1.0-myvendor-01]`.
- `<qualifier>`'s main purpose is for pre-releases (i.e. alpha, beta, rc). From a release timeline, the qualifier is lexographically compared, with no qualifier having the greatest value. So 3.1.0-alpha < 3.1.0-beta < 3.1.0-rc < 3.1.0-sp1 < 3.1.0. The desired behavior would be that 3.1.0-sp1 > 3.1.0

Alternative Designs Considered

Three sub-optimal alternatives to handle the vendor tag requirement in the existing version scheme have been considered

1. Alter groupId or artifactId to include the vendor's tag. An artifactId of commons-logging becomes commons-logging-myvendor. This has the effect of bubbling the version dependency up.
2. Restrict the Maven remote repository only to those which contain artifacts built by a vendor. With this scheme no metadata about the vendor is possible in the local repository. It's a very tenuous dependency at best
3. Exactly specify the vendor tag and release (e.g. {{[3.1.0-myvendor-sp1])}}. Effectively this treats the whole string as a qualifier and does work for that specific library, but one loses all ordering relationships. You can't specify `[3.1.0-myvendor-sp1,3.1.1)` with this scheme.

For handling post release versions an alternative would be

1. Use the build number to represent a patch release or service pack. commons-logging-3.1.0-03.jar would in this case be presumed to be different than commons-logging-3.1.0-01.jar and contain the additional fixes and patches. While this could work, it's not explicit that the build number correlates to additional patches.

Proposal

In adding these addition requirements, it is important to keep in mind the existing version scheme which is good for casual use of libraries from an existing Maven repository like iBiblio. Thus we don't want to break backward compatibility.

For handling the vendor tag requirement we propose a vendor qualifier tag to the existing version:

```
<major>.<minor>.<revision>[:vendortag]([-<qualifier>]|[-
<build>])
```

This tag if present in a version dependency would be the most binding. A specification of `[3.1.0:myvendor-SNAPSHOT]` would require that the snapshot used must have that 'myvendor' identifier.

In conjunction with this vendor qualifier tag, it may be acceptable to use the `[-<build>]` to distinguish follow-on patch releases, essentially correlating 'service packs' to a specific build number. With the presence of a vendor qualifier, this allows the vendor to define what each build number represents.

Putting this altogether, we can specify the version dependencies for the examples as follows:

(exactly 3.1) = `[3.1]`
(3.1 built by us) = `[3.1:us]`
(3.1 built by us and sp2) = `[3.1:us-02]` (assuming sp2 correlates to build 2)
(3.1 built by us and at least sp2) = `[3.1:us-02,3.1.1:us)`