Controlling Compatibility

Why Declare Compatible Versions

Structure Java API will change with time, and it is a good practice to ensure that your plugin uses the correct version of the API.

Structure API Versions page explains how version numbers change based on how compatibility is affected. Say, you develop your code using Structure API version 16.2.0 – your code will work with any version of the API starting from 16.2.0 and up to, but not including version 17.0.0.

So what happens if your code is run on JIRA with Structure that provides an incompatible API? It may break, or it may work. The exact answer depends on which parts of the API you use and what are the differences. But if the code breaks, it may not break outright – it may seem to work at first, until it tries to use a method that's not there, for example.

To make your code fail fast, you can declare dependency on a specific range of versions of the Structure API. In that case, if the version of the API is different, your plugin will fail to load and the user will immediately know that there's a problem.

Importing Specific Range of API Versions

You can declare dependency on the specific range of the API versions via OSGi bundle instructions added to your `pom.xml` or `atlassian-plugin.xml`. Figure out the compatible OSGi versions range from the API versions table and modify your `pom.xml` to contain the following:

```
<plugin>
  <groupId>com.atlassian.maven.plugins</groupId>
  <artifactId>maven-jira-plugin</artifactId>
  ...
  <configuration>
    <instructions>
      <Import-Package>
        com.almworks.jira.structure.api*;version="[16,17)",
        com.almworks.integers*;version="0",
        org.jetbrains.annotations;version="0"
      </Import-Package>
    </instructions>
  </configuration>
</plugin>
```

Here we are declaring the acceptable range of versions for the Structure classes, taken from the example above. We don't much care about the versions of Integers and Annotations libraries, so `version="0"` will match any version of those packages.

✓ You may have other `Import-Package` instructions to declare dependency rules for other packages, and you may have other instructions besides `Import-Package` as well. See the API Usage Samples for a more complete example.

Next: Making Structure Dependency Optional