# Making Structure Dependency Optional

If you are integrating your plugin with Structure, or when you generally write code that uses Structure API but also should work when Structure Plugin is not present, you need to declare that dependencies are optional and isolate dependencies in the code.

## 1. Declare Optional Dependency

Since your plugin must first be loaded as an OSGi bundle, it should declare dependencies from the Structure API packages as optional.

Modify `<Import-Package>` declaration in your `pom.xml` or `atlassian-plugin.xml` and add `resoltion:=optional` classifier. (Add Import-Package to control API compatibility if you don't have this declaration yet.)

```
<Import-Package>
  com.almworks.jira.structure*;version="[16,17)";resolution:=optional,
  com.almworks.integers*;version="0";resolution:=optional,
  org.jetbrains.annotations;version="0";resolution:=optional
</Import-Package>
```

## 2. Isolate Dependencies in the Code

So once you have declared the optional resolution of the Structure API classes, your bundle will load - but if your code tries to access a class from the Structure API, you'll get a `NoClassDefFoundError`. To avoid that, you need to isolate the dependency on Structure API classes - typically in some wrapper classes.

> ✓ This is also a point to make design decisions. So your code can use Structure when it's present, and can work independently when Structure is not there. Are there any abstractions that address both of these situation? What are the concepts that are realized through Structure API and through some other means when Structure is not avialable?

Here's a sample wrapper for the Structure API that provides `ForestAccessor` wrapper (whatever it does) when Structure is available and `null` otherwise.

```java
public class StructureAccessor {
  public static boolean isStructurePresent() {
    if (!ComponentAccessor.getPluginAccessor().isPluginEnabled("com.almworks.jira.structure")) {
      return false;
    }
    try {
      Class.forName("com.almworks.jira.structure.api.StructureComponents");
    } catch (Exception e) {
      return false;
    }
    return true;
  }

  public static ForestAccessor getForest(long structureId) {
    if (!isStructurePresent()) return null;
    StructureComponents structureComponents;
    try {
      structureComponents = ComponentAccessor.getOSGiComponentInstanceOfType(StructureComponents.class);
    } catch (Exception e) {
      return null;
    }


    try {
      return new ForestAccessor(structureComponents.getForestService().getForestSource(ForestSpec.structure
(structureId)));
    } catch (StructureException e) {
      return null;
    }
  }
}
```