RFC 70 - Bundle Dependency and Class Loading Changes

Final

17 Pages

Abstract

This RFC describes enhancements to the bundle class loader.

0 Document Information

0.1 Table of Contents

All Page Within This Box

0.2 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [1].

`Source code is shown in this typeface`.

0.3 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	21 Nov 2003	First Draft Thomas Watson, IBM, tjwatson@us.ibm.com BJ Hargrave, IBM, hargrave@us.ibm.com
Draft 2	24 Nov 2003	Added comments and issues in the area of loading resources from required bundles. Should a bundle be able to provide a package that it exports/imports? Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 3	24 Jan 2004	Removed most of the solution details. Added sections that describe the concepts of the solution in terms of class spaces. The solution details will be added back after the concepts have been agreed upon. Thomas Watson, IBM, tjwatson@us.ibm.com BJ Hargrave, IBM, hargrave@us.ibm.com
Draft 4	09 Feb 2004	Updates from Austin face to face CPEG meeting. Added back solution details from CPEG discussions. Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 5	24 Feb 2004	Updates from CPEG conference call. Added pitfalls section. Added more usecases. Thomas Watson, IBM, tjwatson@us.ibm.com

Draft 6	02 Mar 2004	Added Bundle Singleton usecase and solution details. Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 7	06 Apr 2004	Updates from Cologne face to face CPEG meeting • Clarified bundle singleton solution and added examples • Modified syntax of Bundle-Version qualifier component • Changed provide-packages attribute to reprovide • Fixed bug in Require-Bundle syntax • Added Registering Services Section • Added diagrams and examples to clarify locating classpath entries section • Added conditions for when a fragment can attach to a resolved/active host Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 8	23 Apr 2004	Minor changes: • Fixed bug in Bundle-Version syntax • Fixed bug in Require-Bundle syntax definitions
Draft 9	12 May 2004	Added changes for version ranges. Improved grammars and made corrections to them to conform to 1.4.1 of R3 spec. BJ Hargrave, IBM, hargrave@us.ibm.com

Draft 10	12 Oct 2004	Moved the following content to RFC 79 • Require-Bundle specification • Class and resource search order • Version range (bundle version) and version selection (bundle selection). Version ranges now apply to both packages and bundles in RFC 79 • Bundle-SymbolicName specification • Lifecycle changes that had to do with bundle symbolic name Removed the following content • Class Space description. This is replaced by the modularity discussion in RFC 79 • Hiding parent class space packages. This is replaced by RFC 79 exporting system classes section • Registering Service Changes. The concept of modularity from RFC 79 allows a service object to come from any exported package even if it is obtained by a require-bundle Added content to restrict when and if a fragment can attach to a host bundle Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 11	21 Jan 2005	Minor updates to clarify fragment starting and stopping behaviour Remove classpath filter features Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 12	26 Jan 2005	Restore grammar for Bundle-Classpath to draft 10 to match other manifest header standard syntax. Accepted all changes. BJ Hargrave, IBM, hargrave@us.ibm.com
Final Review Draft	24 Feb 2005	Change fragment-attachment attribute to a directive Clarify that fragment updates are lazy just like normal bundles and do not take effect until the bundle is refreshed or the framework is restarted.
Final	27 May 2005	No changes BJ Hargrave, IBM, hargrave@us.ibm.com

1 Introduction

The OSGi Framework was originally designed for embedded use and its design has been remarkably robust and stable. However, additional usage scenarios for the framework are now appearing that require the framework be further enhanced.

One area that requires further enhancement is in the bundle class loader. This RFC describes the additional requirements, enhancements and design of the bundle class loader.

2 Application Domain

The original context for the framework was the embedded world (e.g. vehicles, gateways, etc.). We are now seeing the need to deploy the framework and its service oriented architecture in different contexts. While the basic operation of the framework is correct. Additional features are needed to enable the framework to be fully useful on new contexts such as desktops and servers which impose additional requirements in the applications that are to be run within the framework. These changes are also important and useful in the traditional contexts in which the OSGi framework is currently successful.

3 Problem Description

Further enhancements to the framework are necessary to support the new usage models the framework is being asked to fulfill. This RFC specifies proposed solutions to enhance the bundle class loader.

4 Requirements

4.1 Bundle Fragments

Bundle Fragments are very similar to bundles but do not stand alone. At runtime the bundle fragment's content (jars, native code, etc.) is seamlessly integrated with its base bundle. See Eclipse Plugins[5] for a discussion of plugin fragments.

4.1.1 Use Cases

1. A bundle developer separates the national language message catalogs from the base bundle into one or more bundle fragments. The base bundle and some set of the corresponding fragments are installed. The message catalogs in all the installed fragments are now loadable by the base bundle.

2. Platform specific Java code could be delivered in fragments. For example ActiveX code on Windows

3. White box testing and code instrumentation. These both require package visibility.

4.2 Bundle RESOLVED/UNRESOLVED events

When a bundle enters the RESOLVED state from the INSTALLED state the Framework must publish a BundleEvent indicating a bundle has entered the RESOLVED state. When a bundle enters the INSTALLED state from the RESOLVED, ACTIVE, STARTING or STOPPING state then the Framework must publish a BundleEvent indicating a bundle has become UNRESOLVED.

4.2.1 Use Cases

A bundle may provide a runtime that allows other bundles to declare functionality that plugs into the runtime. When a bundle RESOLVED event occurs the runtime bundle searches the RESOLVED bundle for any declarations that can plug into the runtime. This would be done by searching for some meta data file within the RESOLVED bundle using Bundle.getEntry(). This allows the runtime bundle to make the functionality of the RESOLVED bundle available without starting it or requiring the OSGi Framework to create the classloader for the RESOLVED bundle. If a request is made to use the declared functionality the runtime would then load the declared class that provides the functionality from the RESOLVED bundle using Bundle.loadClass().

This allows for many bundles to plugin to the runtime bundle without consuming large startup memory and overhead to start all the installed bundles and creating all of their classloaders until the functionality they declare is requested.

5 Technical Solution

5.1 Overview

The bundle class loader needs to be enhanced to allow for fragment bundles to be attached and lazy creation of class loaders. Additionally there is a need to specify the behavior of resource URLs that are returned by the bundle class loader and by the methods on the Bundle object.

5.2 Bundle Class Loader Changes

5.2.1 Bundle Fragments

Conceptually, a fragment is considered part of a host bundle. A fragment Bundle-ClassPath entries must only be allowed to insert additional classes and resources into a host bundle. The Framework must not allow a fragment to replace any class or resource of a host bundle. Thus a fragment is considered to append its Bundle-ClassPath elements to a host bundle's classpath. A fragment may supply classpath entries that are inserted into the host's specified Bundle-ClassPath. See Locating Classpath Entries 5.2.3.1.

A fragment must not be a host to another fragment. When a host bundle and a fragment bundle are resolved then the Framework must attach the fragment to the host bundle. A host bundle and its fragment bundles must be resolved at the same time. See Attaching Fragment Bundles.

In order for a host bundle to allow fragments to attach, the host bundle must have BundlePermission[<bundle symbolic name>,"host"]. In order to be allowed to attach to a host bundle, a fragment bundle must have BundlePermission[<bundle symbolic name>,"fragment"]. See BundlePermission in RFC 73 for more information.

5.2.1.1 Fragment-Host Manifest Header

The Fragment-Host manifest header allows a fragment bundle to specify the host bundle(s) that the fragment bundle should attach to. The Framework must attach a resolved fragment bundle to the resolved host bundle(s) selected by the specified Fragment-Host manifest header.

The Fragment-Host manifest header must conform to the following syntax:

```
Fragment-Host ::= bundle-description
bundle-description ::= symbolic-name (';' parameter ) *
parameter ::= directive | matching-attribute
directive ::= name ':=' value
matching-attribute ::= name '=' value
name ::= token
value ::= token | ( '"' value-string '"' )
```

The following directives are architected by the framework for Fragment-Host:

- **multiple-hosts** If the value of this parameter is "`true`" then the fragment will attempt attach to all bundles selected by `bundle-version` that can be resolved. If the value is "`false`", then the fragment will only attach to the selected bundle with the greatest version that can be resolved. The default value is "`false`".

The following matching attributes are architected by the framework for Fragment-Host:

- **bundle-version** The value of this parameter is a `version-range` to select the bundle that provides the host bundle. See RFC 79 for the format of the version range. The default value is “0.0.0”.

5.2.1.2 Bundle-SymbolicName Attributes

A bundle host is may specify attributes on the Bundle-SymbolicName manifest header that indicate to the framework if and when a fragment may attach to the host bundle. If these attributes are specified by a fragment bundle manifest then the framework must ignore them.

The following directives are recognized by the framework for Bundle-SymbolicName:

- **fragment-attachment** - a string taking one of the values of ”always”, “resolve-time” or “never”. The default value is “always”
    - **always** – indicates that fragments are allowed to attach to the host bundle at any time (while the host is resolved or during the process of resolving the host bundle).
    - **resolve-time** – indicates that fragments are allowed to attach to the host bundle only during the process of resolving the host bundle.
    - **never** – indicates that no fragments are allowed to attach to the host bundle at any time.

5.2.1.3 Attaching Fragment Bundles

When a fragment bundle becomes resolved the Framework must attach the fragment bundle to the selected host bundle(s). When a fragment bundle is attached to a host bundle it logically becomes part of the host bundle. All classes and resources within the fragment bundle must be loaded using the class loader of a host bundle. The fragment bundles of a host bundle must be attached to the host bundle in the order that the fragment bundles are installed. That is ascending bundle id order. If an error occurs during the attachment of a fragment bundle then the fragment bundle must not be attached to the host. A fragment bundle must enter the resolved state only if it has been successfully attached to one or more host bundles.

A framework implementation may support attaching fragments to an already resolved bundle but are not required to. Attaching a fragment bundle to an already resolved host bundle is possible if and only if the following conditions are true:

1. The fragment is logically at the end of the list of attached fragment bundles of the host bundle. This would normally be the case given that fragment bundles are attached in ascending bundle id order.
2. The fragment’s Import-Package and Export-Package entries do not add additional packages to the host.
3. The fragment’s Require-Bundle entries do not add additional required bundles to the host.

If any of these conditions are not true then the fragment must not be attached to the host until the Framework is restarted or the host bundle is refreshed.

When an attached fragment bundle is updated, the content of the previous fragment must remain attached to the host bundle. The new content of the updated fragment must not be allowed to attach to the host bundle until the Framework is restarted or the host bundle is refreshed.

When attaching a fragment bundle to a host bundle the Framework must perform the following steps.

1. Append the Bundle-Classpath entries for the fragment bundle to the Bundle-Classpath of the host bundle.
2. Append the Import-Package entries for the fragment bundle that do not conflict with an Import-Package entry of the host to the Import-Package entries of the host bundle. A fragment Import-Package entry

conflicts with a host Import-Package entry only if it has the same package name and any of its directives or matching attributes are different. If a conflict is found, the fragment is not attached to the host bundle.

3. Append the Require-Bundle entries of the fragment bundle that do not conflict with a Require-Bundle entry of the host to the Require-Bundle entries of the host bundle. A fragment Require-Bundle entry conflicts with a host Require-Bundle entry only if it has the same bundle symbolic name but a different version range. If a conflict is found, the fragment is not attached to the host bundle.
4. Append the Export-Package entries of a fragment bundle that do not conflict with an Export-Package entry of the host to the Export-Package entries of the host bundle. A fragment Export-Package entry conflicts with a host Export-Package entry only if it has the same package name. If a conflict is found, the export from the fragment is ignored.

When a host bundle becomes unresolved then all attached fragment bundles must be detached from the host bundle. When a fragment bundle becomes unresolved the Framework must detach it from the host and reresolve the host bundle and reattach the remaining attached fragments.

5.2.2 Bundle Class Loaders

Each bundle installed in the Framework that is not a fragment bundle must not have a class loader until after it is resolved. Frameworks must only create one class loader per each bundle that is not a fragment. The Framework must not create a class loader for fragment bundles. Fragment bundles resources and classes are loaded by their host bundle’s class loader (See Bundle Fragments).

One class loader per bundle allows all resources within a bundle to have package level access to all other resources in the bundle within the same package. This class loader provides each bundle with its own name-space, to avoid name conflicts, and allows resource sharing with other bundles. The bundle's class loader must find classes and resources in the bundle by searching the bundle's classpath as specified by the Bundle-Classpath header in the bundle's manifest and any attached fragment bundles (see Bundle Classpath and Bundle Fragments).

5.2.3 Bundle Class Path

The Bundle-Classpath manifest header declares the intrabundle classpath. This declaration allows a bundle to declare its internal classpath using one or more directory or JAR file entries that are contained within the bundle or any attached fragments. For example, a bundle could contain servlet.jar and cocoon.jar as entries. Both entries need to be part of the bundle’s classpath. The Bundle-Classpath manifest header specifies these embedded JAR files.

The Bundle-Classpath manifest header is a list of comma-separated classpath entries. A classpath entry consists of a list of semi-colon-separated bundle entry names and optionally, a semicolon (‘;’) followed by parameters. A bundle entry name is either the path of a directory or JAR file entry contained in the bundle or any attached fragments. The directory of dot (’.’ or ‘/’) represents the entire bundle.

Classpath dependencies must be resolved as follows:

- If a Bundle-Classpath header is not declared, the default value of dot (’.’) is used, which specifies the entire bundle.
- If a Bundle-Classpath manifest header is declared and dot (’.’ or ‘/’) is not an element of the classpath, then the bundle itself must not be searched. In this case, only the specified classpath entries must be searched.

The Bundle-Classpath manifest header must conform to the following syntax:

```
Bundle-Classpath ::= classpath-entries ( ’,’ classpath-entries )*

classpath-entries ::= classpath (‘;’ classpath)* (‘;’ parameter)*
```

```
      classpath ::= jar-path | (’”’jar-path’”’)

      parameter ::= attribute ’=’ value
      attribute ::= token
      value ::= token | quoted-string
```

The `jar-path` value may reference a jar file or directory within the bundle. The path is relative to the root of the bundle.

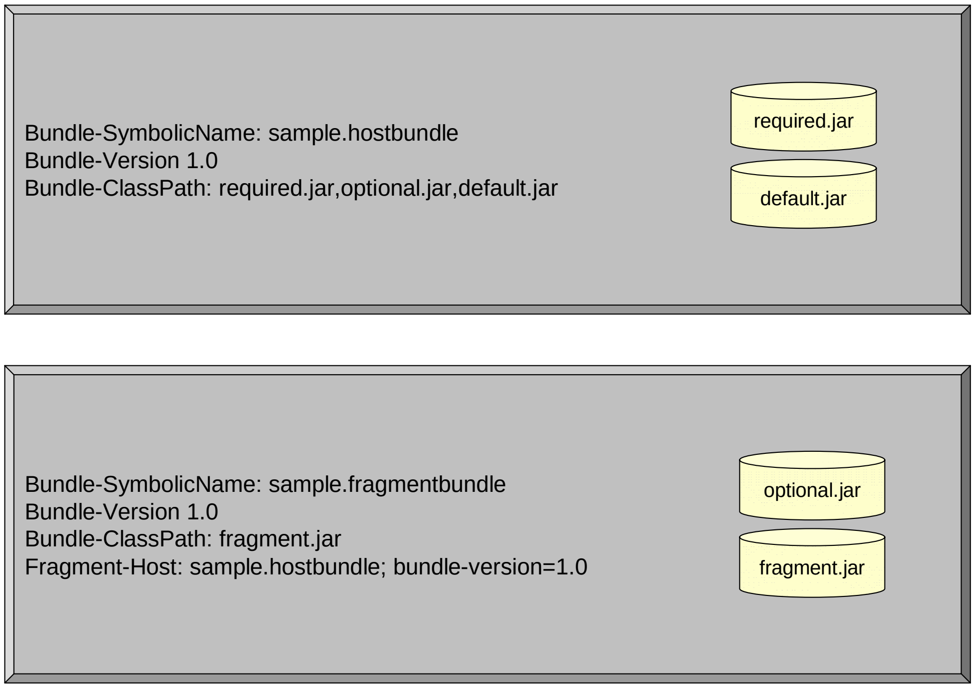
No Bundle-Classpath parameters are recognized by the Framework. The framework must ignore any unrecognized parameters.

For example, the following declaration in a bundle’s manifest will allow classes and resources to be accessed from the bundle and also all classes and resources defined in lib/servlet.jar:

```
      Bundle-Classpath: ., lib/servlet.jar
```

5.2.3.1 Locating Classpath Entries

When locating a classpath entry in a bundle the Framework must attempt to locate the classpath entry relative to the root of the bundle. If a classpath entry cannot be located in the bundle, then the Framework must attempt to locate the classpath entry in each attached fragment bundle. The attached fragment bundles are searched in ascending bundle id order. This allows a fragment to supply classpath entries that are inserted into the host’s specified Bundle-ClassPath. The following example illustrates this:



```
Bundle-SymbolicName: sample.hostbundle
Bundle-Version 1.0
Bundle-ClassPath: required.jar,optional.jar,default.jar
```

```
Bundle-SymbolicName: sample.fragmentbundle
Bundle-Version 1.0
Bundle-ClassPath: fragment.jar
Fragment-Host: sample.hostbundle; bundle-version=1.0
```

In this example the host bundle with the symbolic name of sample.hostbundle has a Bundle-ClassPath with three entries (required.jar, optional.jar, and default.jar). The required.jar classpath entry may contain the classes and resources that must be present for the bundle to function. The optional.jar classpath entry may contain classes and resources that if present the bundle will use. The default.jar classpath entry may contain classes and resources that the bundle will use if the optional.jar is not available, but the classes and resources from default.jar may get overridden by classes and resources in the required.jar or optional.jar classpath entries. The bundle sample.hostbundle only has the required.jar and default.jar entries packaged with it. This allows a fragment bundle to be installed that can supply the optional.jar classpath entry for the host.

The fragment bundle with the symbolic name of sample.fragmentbundle has a Bundle-ClassPath with one entry (fragment.jar). When the bundle sample.hostbundle is resolved and the fragment bundle sample.fragmentbundle is attached then the bundle classpath for the bundle sample.hostbundle will be required.jar, optional.jar, default.jar, fragment.jar.

The Framework must ignore a classpath entry in the Bundle-Classpath header if the entry cannot be located. However, the Framework should publish a Framework Event of type INFO with an appropriate message for each entry that cannot be located. The Framework should also publish a Framework Event of type INFO if the Bundle-Classpath evaluates to the null set.

5.3 Bundle Lifecycle Changes

Fragment bundles lifecycle operations are handled differently than host bundles by the Framework.

5.3.1 Starting Bundles

Fragment bundles must not specify a Bundle Activator. If the Bundle.start method is called on a Bundle object for a fragment, then the framework must throw a BundleException indicating that a fragment bundle cannot be started.

5.3.2 Stopping Bundles

Fragment bundles must not specify a Bundle Activator. If the Bundle.stop method is called on a Bundle object for a fragment, then the framework must throw a BundleException indicating that a fragment bundle cannot be started.

5.4 Resolved and Unresolved Bundle Events

Once the bundle is resolved by the framework, the framework must publish a Bundle Event of type RESOLVED. A bundle may become unresolved at some future time. This could occur as a result of uninstalling the bundle, refreshing its package or some other reason. When a bundle becomes unresolved, the framework must publish a Bundle Event of type UNRESOLVED.

When a set of bundles are refreshed using the PackageAdmin API then each bundle in the set must have an UNRESOLVED BundleEvent published. The UNRESOLVED BundleEvent must be published after all the bundles in the set have been stopped and before any of the bundles in the set are re-started.

5.5 Bundle Resource URLs

In order to have access to a resource in a bundle, appropriate permissions are required. A bundle must always be given permission by the Framework to access the resources contained within itself, as well as permission to access any resources in imported packages.

When a URL object to a resource within a bundle is created, the caller is checked for AdminPermission to access the resource if the resource is located in another bundle. If the caller does not have the necessary permission, the resource is not accessible and SecurityException is thrown. If the caller has the necessary permission, then the

URL object to the resource is successfully created. Once the URL object is created, no further permission checks are performed when the contents of the resource are accessed.

A resource in a bundle may also be accessed by using the Bundle.getResource or Bundle.getEntry methods. The Bundle.getResource method calls getResource on the bundle's classloader to perform the search. The caller of Bundle.getResource must have AdminPermission[<bundle id>,"resource"]. The Bundle.getEntry method does not use the bundle's classloader to locate the resource. Thus Bundle.getEntry can be called on an unresolved bundle. Bundle.getEntry only searches within the bundle itself and does not use standard bundle classloader search rules. The caller of Bundle.getEntry must have AdminPermission[<bundle id>,"resource"].

The URL objects that reference bundle resources or bundle entries must use a scheme defined by the Framework implementation. The external form of these URL objects must be defined by the Framework implementation. The URLs using the protocol scheme defined by the Framework for bundle resources and bundle entries must be hierarchical (See RFC 2396: Uniform Resource Identifiers URI: Generic Syntax [6]) and able to be used as a context for constructing another URL. The java.net.URLStreamHandler used for bundle resource and bundle entry URLs must be available to the java.net.URL class to setup a URL that uses the protocol scheme defined by the Framework.

The URL.getPath method for a URL that uses the bundle resource or bundle entry protocol must return an absolute path (a path that starts with '/') to a resource or entry in a bundle. For example, the URL returned from Bundle.getEntry("myimages/test.gif") must have a path of "/myimages/test.gif".

The java.net.URLStreamHandlers for bundle resource and entry URL objects must perform a permissions check to allow access to the URL. When a URL is constructed using the following java.net.URL constructors the parseURL method of the URLStreamHandler is called:

- URL(String spec)
- URL(URL context, String spec)
- URL(URL context, String spec, URLStreamHandler handler)

When one of these constructors is called for a URL that uses the bundle resource or entry protocol scheme then the parseURL method of the URLStreamHandler must check the caller for the necessary permissions. If the caller does not have the necessary permissions then a SecurityException must be thrown. If the caller has the necessary permissions, then the URL object is setup to access the bundle resource and the authority of the URL is set to a Framework defined value. When the content of the URL is accessed a check is done to see if the authority of the URL has been set to the Framework defined value. If it is set then the caller is allow access, if not then the caller is checked for the necessary permissions.

The following java.net.URL constructors do not call parseURL to setup a URL to use a specific protocol.

- URL(String protocol, String host, int port, String file)
- URL(String protocol, String host, int port, String file, URLStreamHandler handler)
- URL(String protocol, String host, String file)

When one of these constructors is called the authority of the constructed URL is not set. When the content of one of these URLs that use the bundle resource or entry protocol scheme is accessed then the caller is checked for the necessary permissions because the authority of the URL is not set to the Framework defined value.

6 Considered Alternatives

6.1 Alternative Fragment Solution

A fragment solution where each fragment bundle has its own classloader has drawbacks such as a fragment does not get package level access to the host.

6.2 Alternative Loading of provided-package Resources

When finding classes or resources in required bundles if the class or resource is not found in any providing bundle, then the Framework must fail the request.

The client of a provided package becomes fragile to the content of the provider.

For example:

Let's take bundles A and B, where A requires B and B provides package 'b'. A can then see classes from B and have in its private name space packages like 'a'. In this case everything works fine. A see classes from B and its local classes.

Now let's suppose that B decides to provide package 'b' and 'a'. Then A will no longer see its internal classes because the lookup will assume that the provider for all 'a' qualified classes comes from B and will therefore not consult the content of A.

To summarize, not checking for local classes when the provided class lookup fails makes requiring more brittle to provider changes.

6.3 Ability to Export a required package

Allow a bundle to export a package that is provided by a bundle that is required. For example, if bundle com.foo.a requires com.foo.b and com.foo.b provides the package com.foo.bpkg then the bundle com.foo.a may export the package com.foo.bpkg. The following manifest files illustrate this.

```
Bundle-SymbolicName: com.foo.a
Require-Bundle: com.foo.b
Export-Package: com.foo.bpkg

Bundle-SymbolicName: com.foo.b
Provide-Package: com.foo.bpkg
```

The following are arguments for not allowing this type of behavior.

- When a bundle requires another bundle there is no way of know exactly what packages the requiring bundle will get. If the required bundle is updated it may stop providing the package that the requiring bundle is trying to export. In the opposite situation where a bundle wants to provide a package that it imports/exports, the package name is clearly specified and must be present in order for the importing bundle to provide the package

- The package specification of the exported package would not be clear. Provided packages have no specification version. The bundle version may be used but this would probably be misleading in most cases.

We decided to not allow for this behavior and to force the resource or class loading to be limited to only the classloader of the bundle that is exporting the package.

6.4 Multiple Bundles contributing to the same NAMED Class Space

- If we allow multiple bundles to contribute to the same NAMED class space then how do we specify the search order of the bundles that contributed to the class space when attempting to locate a class or resource?

- What happens when two bundles contribute the same package to the same NAMED class space? Do we allow package splitting across the different bundles contributing to the same NAMED class space?. What happens if one bundle imports a package and then contributes it to a NAMED class space and then another bundle contributes the same package but does not import it?

6.5 Version-match attribute

The version-match attribute and its values have been replaced by a version range syntax for the bundle-version attribute.

The version-match attribute supported the following match rules: qualifier, micro, minor, major or greater than or equal.

- Two versions are considered to match on **qualifier** if their major, minor, micro and qualifier components are equal.

- A version is considered to match on **micro** with another version if its major, minor and micro components are equal to the other version’s major, minor and micro components and its qualifier is greater than or equal to the other version’s qualifier (using String.compareTo).

- A version is considered to match on **minor** with another version if its major and minor components are equal to the other version’s major and minor components, and its micro level is greater than or equal to the other version’s micro level. If the micro levels are equal, a version is considered to match on **minor** with another version if its qualifier is greater than or equal to the other version’s qualifier (using String.compareTo).

- A version identifier is considered to match on **major** with another version identifier if its major component is equal to the other version’s major component, and its minor component is greater than or equal to the other version’s minor component. If the minor components are equal, a version is considered to be match on **major** with another version if its micro level is greater than or equal to the other version’s micro level. If the micro levels are equal, a version is considered to match on **major** with another version if its qualifier is greater than or equal to the other version’s qualifier (using String.compareTo).

- A version is considered to match on **greater than or equal to** with another version identifier if its major component is greater than the other version’s major component, or the major components are equal and its minor component is greater than the other version’s minor component, or the major and minor components are equal and its micro component is greater than the other version’s micro component, or the major, minor and micro components are equal and it’s qualifier component is greater than or equal to the other version’s qualifier component (using String.compareTo).

7 Security Considerations

- The current design of fragments will allow a fragment to import a package that already exists in the host but is not exported. This will in effect give the fragment the ability to replace a package in the host with an imported package.
- The new BundlePermission class is used to assign bundles the permission to provide packages, require bundles, and attach as fragment bundles to host bundles.

8 Document Support

8.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

[3]. The Java 2 Package Versioning Specification http://java.sun.com/j2se/1.4/docs/guide/versioning/index.html

[4]. Package Version Identification, Java 2 JDK 1.4 Documentation, http://java.sun.com/j2se/1.4.1/docs/guide/versioning/index.html

[5]. "What the heck is a plug-in anyway?", Debbie Wilson and Jeff McAffer, IBM, http://dev.eclipse.org/viewcvs/index.cgi/~checkout~/platform-core-home/documents/plugins/article.html?rev=HEAD&content-type=text/html

[6]. RFC 2396: Uniform Resource Identifiers URI: Generic Syntax, http://www.ietf.org/rfc/rfc2396.txt

[7]. Interval Notation, http://www.math.ohio-state.edu/courses/math104/interval.pdf

8.2 Author's Address

Name	Thomas Watson
Company	IBM
Address	11501 Burnet Rd, Austin, TX 78758 USA
Voice	+1 512 838 4533
e-mail	tjwatson@us.ibm.com

Name	BJ Hargrave
Company	IBM
Address	11501 Burnet Rd, Austin, TX 78758 USA
Voice	+1 512 838 8838
e-mail	hargrave@us.ibm.com

8.3 Acronyms and Abbreviations

8.4 End of Document