RFC 71 - Framework API Changes

Final

29 Pages

Abstract

This RFC describes API changes to support enhancements to the framework.

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 Jennifer Fogell, IBM, jfogell@us.ibm.com Thomas Watson, IBM, tjwatson@us.ibm.com BJ Hargrave, IBM, hargrave@us.ibm.com
Draft 2	24 Nov 2003	Added the Constant OPTIONAL_ATTRIBUTE. Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 3	02 Mar 2004	Updates from Austin face to face CPEG meeting. Moved most additions to Bundle and BundleContext APIs to the PackageAdmin service. Thomas Watson, IBM, tjwatson@us.ibm.com

All Page Within This Box

Draft 4	07 Apr 2004	Added FrameworkEvent type of WARNING Changed PackageAdmin.isFragment to PackageAdmin.getBundleType Added PackageAdmin.resolveBundles method Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 5	22 Apr 2004	Added selection-filter and require-packages constants Added Boolean return type to the PackageAdmin.resolveBundles method
Draft 6	12 May 2004	Moved BundlePermission to RFC 81 Replaced text with latest javadoc. BJ Hargrave, IBM, hargrave@us.ibm.com
Draft 7	9 June 2004	Added back Version class w/o matching methods. BJ Hargrave, IBM, hargrave@us.ibm.com
Draft 8	16 Nov 2004	Added the following methods: • PackageAdmin.getBundle(Class) • PackageAdmin.getRequiredBundles(String) • PackageAdmin.getAllExportedPackages(String) • Bundle.getResources(String) Added a new RequiredBundle class that replaces the ProvidingBundle class. Updated the Constants class to include new constants from RFC 79 and updates from RFC 70 Thomas Watson, IBM, tjwatson@us.ibm.com
Draft 9	2 Dec 2004	Added javadoc for fragments on Bundle interface.
Final	27 May 2005	No Changes BJ Hargrave, 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.

In order to support additional usage scenarios for the framework, various API changes are necessary. This RFC describes the additional requirements, enhancements and design of the API changes.

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 is a collection of proposed framework API changes.

4 Requirements

Various. See RFP 44[3] and 48[4] as well as RFC 70[5].

4.1 Use Cases

4.1.1 Bundle Events

A bundle does not become meaningful or useful in the framework (as far as providing resources through exporting/providing packages) until it become resolved. Some containers running on top of OSGi need event notification of when bundles become resolved so that they can know that a new bundle has become "useful" and may be "deployed" into the container.

4.1.2 Bundle.isFragment/Bundle.getHost

Fragments appear as regular bundles, and other bundles may want to tell them apart, and find out the relationships without having to process all existing bundles.

4.1.3 Bundle.getFragments

1) A bundle needs to access all known host bundles and their fragments.

2) A bundle needs to obtain a location for a file contributed by some fragment not known beforehand.

3) A bundle needs to compute the classpath for a set of bundles (and its fragments) when running another Java application (Ant/Tomcat).

5 Technical Solution

This RFC collects a set of framework API changes together from other related RFCs.

5.1 Bundle changes

These new methods come from RFC 70.

5.1.1 getHeaders

```
public java.util.Dictionary getHeaders(java.lang.String localeString)
```

Returns this bundle's Manifest headers and values localized to the specifed locale.

All Page Within This Box

This method performs the same function as `Bundle.getHeaders()` except the manifest header values are localized to the specified locale. If a Manifest header value starts with "%", it will be localized with the localization properties file for the specified locale. If `null` is specified as the locale string, the header values will be localized using the default locale. If the empty string ("") is specified as the locale string, the header values will not be localized but any leading "%"s will be stripped off of the header values.

This method will continue to return Manifest header information while this bundle is in the `UNINSTALLED` state, however the header values will only be localized to the default locale.

Returns:
A `Dictionary` object containing this bundle's Manifest headers and values.
Throws:
`java.lang.SecurityException` - If the caller does not have the `AdminPermission`, and the Java Runtime Environment supports permissions.
Since:
1.3
See Also:
`getHeaders()`, `Constants.BUNDLE_LOCALIZATION`

5.1.2 getSymbolicName

```
public java.lang.String getSymbolicName()
```

Returns the symbolic name of this bundle as specified by its `Bundle-SymbolicName` manifest header. The name must be unique, it is recommended to use a reverse domain name naming convention like that used for java packages. If the bundle does not have a specified symbolic name then `null` is returned.

This method will continue to return this bundle's symbolic name while this bundle is in the `UNINSTALLED` state.

Returns:
The symbolic name of this bundle.
Since:
1.3

5.1.3 loadClass

```
public java.lang.Class loadClass(java.lang.String name)
                         throws java.lang.ClassNotFoundException
```

Loads the specified class using this bundle's classloader.

If this bundle's state is `INSTALLED`, this method will attempt to resolve the bundle before attempting to load the class.

If the bundle cannot be resolved, a Framework event of type `FrameworkEvent.ERROR` is broadcast containing a `BundleException` with details of the reason the bundle could not be resolved. This method must then throw a `ClassNotFoundException`.

If the bundle is a fragment bundle then this method must throw a `ClassNotFoundException`.

If this bundle's state is `UNINSTALLED`, then an `IllegalStateException` is thrown.

Parameters:
name - The name of the class to load.
Returns:
The Class object for the requested class.
Throws:
java.lang.ClassNotFoundException - If no such class can be found or if this bundle is a fragment bundle or if the caller does not have the AdminPermission, and the Java Runtime Environment supports permissions.
java.lang.IllegalStateException - If this bundle has been uninstalled.
Since:
1.3

5.1.4 getResource

public java.net.URL **getResource**(java.lang.String name)

Find the specified resource in this bundle. This bundle's class loader is called to search for the named resource. If this bundle's state is INSTALLED, then only this bundle will be searched for the specified resource. Imported packages cannot be searched when a bundle has not been resolved. If this bundle is a fragment bundle then null is returned.
Parameters:
name - The name of the resource. See java.lang.ClassLoader.getResource for a description of the format of a resource name.
Returns:
a URL to the named resource, or null if the resource could not be found or if this bundle is a fragment bundle or if the caller does not have the AdminPermission, and the Java Runtime Environment supports permissions.
Throws:
java.lang.IllegalStateException - If this bundle has been uninstalled.
Since:
1.1

5.1.5 getResources

public java.util.Enumeration **getResources**(java.lang.String name)

Find the specified resources in this bundle. This bundle's class loader is called to search for the named resource. If this bundle's state is INSTALLED, then only this bundle will be searched for the specified resource. Imported packages cannot be searched when a bundle has not been resolved. If this bundle is a fragment bundle then null is returned.
Parameters:
name - The name of the resource. See java.lang.ClassLoader.getResources for a description of the format of a resource name.
Returns:
an Enumeration of URLs to the named resources, or null if the resource could not be found or if this bundle is a fragment bundle or if the caller does not have the AdminPermission, and the Java Runtime Environment supports permissions.
Throws:
java.lang.IllegalStateException - If this bundle has been uninstalled.
Since:

1.3

5.1.6 getEntryPaths

`public java.util.Enumeration` **`getEntryPaths`**`(java.lang.String path)`

Returns enumeration of all the paths to entries within the bundle whose longest sub-path matches the supplied path argument. The bundle's classloader is not used to search for entries. Only the contents of the bundle is searched. A specified path of "/" indicates the root of the bundle.

Returned paths indicating subdirectory paths end with a "/". The returned paths are all relative to the root of the bundle and have a leading "/".

This method returns null if no entries could be found that match the specified path or if the caller does not have `AdminPermission` and the Java Runtime Environment supports permissions.

Parameters:
`path` - the path name to get the entry path names for.
Returns:
An Enumeration of the entry paths that are contained in the specified path or null if no entries could be found that match the specified path or if the caller does not have `AdminPermission` and the Java Runtime Environment supports permissions..
Throws:
`java.lang.IllegalStateException` - If this bundle has been uninstalled.
Since:
1.3

5.1.7 getEntry

`public java.net.URL` **`getEntry`**`(java.lang.String name)`

Returns a URL to the specified entry in this bundle. The bundle's classloader is not used to search for the specified entry. Only the contents of the bundle is searched for the specified entry. A specified path of "/" indicates the root of the bundle.

This method returns a URL to the specified entry, or `null` if the entry could not be found or if the caller does not have the `AdminPermission` and the Java Runtime Environment supports permissions.

Parameters:
`name` - The name of the entry. See `java.lang.ClassLoader.getResource` for a description of the format of a resource name.
Returns:
A URL to the specified entry, or `null` if the entry could not be found or if the caller does not have the `AdminPermission` and the Java Runtime Environment supports permissions.
Throws:
`java.lang.IllegalStateException` - If this bundle has been uninstalled.
Since:
1.3

5.2 BundleEvent changes

These new event types come from RFC 70 .

5.2.1 RESOLVED

`public static final int` **`RESOLVED`**

The bundle has been resolved.

The value of `RESOLVED` is 0x00000020.

Since:
1.3
See Also:
`Bundle.RESOLVED`, Constant Field Values

5.2.2 UNRESOLVED

`public static final int` **`UNRESOLVED`**
The bundle has been unresolved.

The value of `UNRESOLVED` is 0x00000040.

Since:
1.3
See Also:
`Bundle.INSTALLED`, Constant Field Values

5.2.3 Other Specification Changes For New Event Types

Section 9.6.1 Bundle Events Mapping (from R3 spec) will need to be updated for these new event types.

5.3 BundleException changes

This exception is updated to conform to the general purpose exception chaining mechanism.

5.3.1 getCause

`public java.lang.Throwable` **`getCause`**`()`
Returns the cause of this exception or `null` if no cause was specified when this exception was created.
Overrides:
`getCause` in class `java.lang.Throwable`
Returns:
The cause of this exception or `null` if no cause was specified.
Since:
1.3

5.3.2 initCause

`public java.lang.Throwable` **`initCause`**`(java.lang.Throwable cause)`
The cause of this exception can only be set when constructed.
Overrides:
`initCause` in class `java.lang.Throwable`
Throws:
`java.lang.IllegalStateException` - This method will always throw an `IllegalStateException` since the cause of this exception can only be set when constructed.
Since:
1.3

All Page Within This Box

5.4 Constants changes

These new constants come from RFCs 70, 72, 74 and 79.

5.4.1 BUNDLE_SYMBOLICNAME

`public static final java.lang.String` **`BUNDLE_SYMBOLICNAME`**

Manifest header (named "Bundle-SymbolicName") identifying the bundle's symbolic name.

The attribute value may be retrieved from the `Dictionary` object returned by the `Bundle.getHeaders` method.

Since:
1.3
See Also:
Constant Field Values

5.4.2 SINGLETON_ATTRIBUTE

`public static final java.lang.String` **`SINGLETON_ATTRIBUTE`**

Manifest header attribute (named "singleton") identifying whether a bundle is a singleton. The default value is `false`.

The attribute value is encoded in the Bundle-SymbolicName manifest header like:

```
Bundle-SymbolicName: com.acme.module.test; singleton=true
```

Since:
1.3
See Also:
Constant Field Values

5.4.3 FRAGMENT_ATTACHMENT_ALWAYS

`public static final java.lang.String` **`FRAGMENT_ATTACHMENT_ALWAYS`**

Manifest header attribute value (named "always") identifying a fragment attachment type of always. A fragment attachment type of 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).

The attribute value is encoded in the Bundle-SymbolicName manifest header like:

```
Bundle-SymbolicName: com.acme.module.test; fragment-attachment="always"
```

Since:
1.3
See Also:
`FRAGMENT_ATTACHMENT_ATTRIBUTE`, Constant Field Values

5.4.4 FRAGMENT_ATTACHMENT_RESOLVETIME

`public static final java.lang.String` **`FRAGMENT_ATTACHMENT_RESOLVETIME`**

All Page Within This Box

Manifest header attribute value (named "resolve-time") identifying a fragment attachment type of resolve-time. A fragment attachment type of resolve-time indicates that fragments are allowed to attach to the host bundle only during the process of resolving the host bundle.

The attribute value is encoded in the Bundle-SymbolicName manifest header like:

```
Bundle-SymbolicName: com.acme.module.test; fragment-attachment="resolve-time"
```

Since:
1.3
See Also:
FRAGMENT_ATTACHMENT_ATTRIBUTE, Constant Field Values

5.4.5 FRAGMENT_ATTACHMENT_NEVER

`public static final java.lang.String` **`FRAGMENT_ATTACHMENT_NEVER`**

Manifest header attribute value (named "never") identifying a fragment attachment type of never. A fragment attachment type of never indicates that no fragments are allowed to attach to the host bundle at any time.

The attribute value is encoded in the Bundle-SymbolicName manifest header like:

```
Bundle-SymbolicName: com.acme.module.test; fragment-attachment="never"
```

Since:
1.3
See Also:
FRAGMENT_ATTACHMENT_ATTRIBUTE, Constant Field Values

5.4.6 BUNDLE_LOCALIZATION

`public static final java.lang.String` **`BUNDLE_LOCALIZATION`**

Manifest header (named "Bundle-Localization") identifying the base name of the bundle's localization file.

The attribute value may be retrieved from the `Dictionary` object returned by the `Bundle.getHeaders` method.

Since:
1.3
See Also:
Constant Field Values

5.4.7 BUNDLE_LOCALIZATION_DEFAULT_BASENAME

`public static final java.lang.String` **`BUNDLE_LOCALIZATION_DEFAULT_BASENAME`**

Default value for the Bundle-Localization manifest header.

Since:
1.3
See Also:
BUNDLE_LOCALIZATION, Constant Field Values

All Page Within This Box

5.4.8 REQUIRE_BUNDLE

`public static final java.lang.String` **`REQUIRE_BUNDLE`**

Manifest header (named "Require-Bundle") identifying the symbolic names of other bundles required by the bundle.

The attribute value may be retrieved from the `Dictionary` object returned by the `Bundle.getHeaders` method.

Since:
1.3
See Also:
Constant Field Values

5.4.9 BUNDLE_VERSION_ATTRIBUTE

`public static final java.lang.String` **`BUNDLE_VERSION_ATTRIBUTE`**

Manifest header attribute (named "bundle-version") identifying a range of versions for a bundle specified in the Require-Bundle or Fragment-Host manifest headers. The default value is `0.0.0`.

The attribute value is encoded in the Require-Bundle manifest header like:

```
Require-Bundle: com.acme.module.test; bundle-version="1.1"
Require-Bundle: com.acme.module.test; bundle-version="[1.0,2.0)"
```

The bundle-version attribute value uses a mathematical interval notation to specify a range of bundle versions. A bundle-version attribute value specified as a single version means a version range that includes any bundle version greater than or equal to the specified version.

Since:
1.3
See Also:
Constant Field Values

5.4.10 FRAGMENT_HOST

`public static final java.lang.String` **`FRAGMENT_HOST`**

Manifest header (named "Fragment-Host") identifying the symbolic name of another bundle for which that the bundle is a fragment.

The attribute value may be retrieved from the `Dictionary` object returned by the `Bundle.getHeaders` method.

Since:
1.3
See Also:
Constant Field Values

5.4.11 MULTIPLE_HOSTS_DIRECTIVE

`public static final java.lang.String` **`MULTIPLE_HOSTS_DIRECTIVE`**

Manifest header directive (named "multiple-hosts") identifying if the fragment should attach to each bundle selected by the Fragment-Host manifest header. The default value is `false`.

The directive value is encoded in the Fragment-Host manifest header like:

```
Fragment-Host: com.acme.module.test; multiple-hosts="false"
```

Since:
1.3
See Also:
Constant Field Values

5.4.12 SELECTION_FILTER_ATTRIBUTE

`public static final java.lang.String` **`SELECTION_FILTER_ATTRIBUTE`**

Manifest header attribute (named "selection-filter") is used for selection by filtering based upon system properties.

The attribute value is encoded in manifest headers like:

```
Bundle-NativeCode: libgtk.so; selection-filter="(ws=gtk)"; ...
Bundle-ClassPath: base.jar, gtk.jar; selection-filter="(ws=gtk)", ...
```

Since:
1.3
See Also:
Constant Field Values

5.4.13 BUNDLE_MANIFESTVERSION

`public static final java.lang.String` **`BUNDLE_MANIFESTVERSION`**

Manifest header (named "Bundle-ManifestVersion") identifying the bundle manifest version. A bundle manifest may express the version of the syntax in which it is written by specifying a bundle manifest version. Bundles exploiting OSGi R4, or later, syntax must specify a bundle manifest version.

The bundle manifest version defined by OSGi R4 or, more specifically, by V1.3 of the OSGi Framework Specification is "2".

The attribute value may be retrieved from the `Dictionary` object returned by the `Bundle.getHeaders` method.

Since:
1.3
See Also:
Constant Field Values

5.4.14 VERSION_ATTRIBUTE

`public static final java.lang.String` **`VERSION_ATTRIBUTE`**

Manifest header attribute (named "version") identifying the version of a package specified in the Export-Package or Import-Package manifest header.

All Page Within This Box

The attribute value is encoded in the Export-Package or Import-Package manifest header like:

```
Import-Package: org.osgi.framework; version="1.1"
```

Since:
1.3
See Also:
Constant Field Values

5.4.15 BUNDLE_SYMBOLICNAME_ATTRIBUTE

`public static final java.lang.String` **`BUNDLE_SYMBOLICNAME_ATTRIBUTE`**

Manifest header attribute (named "bundle-symbolic-name") identifying the symbolic name of a bundle which exports a package specified in the Import-Package manifest header.

The attribute value is encoded in the Import-Package manifest header like:

```
Import-Package: org.osgi.framework; bundle-symbolic-
name="com.acme.module.test"
```

Since:
1.3
See Also:
Constant Field Values

5.4.16 RESOLUTION_DIRECTIVE

`public static final java.lang.String` **`RESOLUTION_DIRECTIVE`**

Manifest header directive (named "resolution") identifying the resolution type in the Import-Package or Require-Bundle manifest header.

The directive value is encoded in the Import-Package or Require-Bundle manifest header like:

```
Import-Package: org.osgi.framework; resolution:="optional"
Require-Bundle: com.acme.module.test; resolution:="optional"
```

Since:
1.3
See Also:
`RESOLUTION_MANDATORY`, `RESOLUTION_OPTIONAL`, Constant Field Values

5.4.17 RESOLUTION_MANDATORY

`public static final java.lang.String` **`RESOLUTION_MANDATORY`**

Manifest header directive value (named "mandatory") identifying a mandatory resolution type. A mandatory resolution type indicates that the import package or require bundle must be resolved when the bundle is resolved. If such an import or require bundle cannot be resolved, the module fails to resolve.

The directive value is encoded in the Import-Package or Require-Bundle manifest header like:

```
Import-Package: org.osgi.framework; resolution:="manditory"
```

```
Require-Bundle: com.acme.module.test; resolution:="manditory"
```

Since:
1.3
See Also:
RESOLUTION_DIRECTIVE, Constant Field Values

5.4.18 RESOLUTION_OPTIONAL

`public static final java.lang.String` **`RESOLUTION_OPTIONAL`**

Manifest header directive value (named "optional") identifying an optional resolution type. An optional resolution type indicates that the import or require bundle is optional and the bundle may be resolved without the import or require bundle being resolved. If the import or require bundle is not resolved when the bundle is resolved, the import or require bundle may not be resolved before the bundle is refreshed.

The directive value is encoded in the Import-Package or Require-Bundle manifest header like:

```
Import-Package: org.osgi.framework; resolution:="optional"
Require-Bundle: com.acme.module.test; resolution:="optional"
```

Since:
1.3
See Also:
RESOLUTION_DIRECTIVE, Constant Field Values

5.4.19 GROUPING_DIRECTIVE

`public static final java.lang.String` **`GROUPING_DIRECTIVE`**

Manifest header directive (named "grouping") identifying the package grouping specified in the Import-Package or Export-Package manifest header.

The directive value is encoded in the Import-Package or Export-Package manifest header like:

```
Export-Package: org.osgi.framework; grouping:="coregroup"
```

Since:
1.3
See Also:
Constant Field Values

5.4.20 INCLUDE_DIRECTIVE

`public static final java.lang.String` **`INCLUDE_DIRECTIVE`**

Manifest header directive (named "include") identifying a list of classes and/or resources of the specified package which must be allowed to be exported in the Export-Package manifest header.

The directive value is encoded in the Export-Package manifest header like:

```
Export-Package: org.osgi.framework; include:="MyStuff*"
```

Since:
1.3

All Page Within This Box

See Also:
Constant Field Values

5.4.21 EXCLUDE_DIRECTIVE

`public static final java.lang.String` **`EXCLUDE_DIRECTIVE`**

Manifest header directive (named "exclude") identifying a list of classes and/or resources of the specified package which must not be allowed to be exported in the Export-Package manifest header.

The directive value is encoded in the Export-Package manifest header like:

```
Export-Package: org.osgi.framework; exclude:="MyStuff*"
```

Since:
1.3
See Also:
Constant Field Values

5.4.22 MANDATORY_DIRECTIVE

`public static final java.lang.String` **`MANDATORY_DIRECTIVE`**

Manifest header directive (named "mandatory") identifying names of matching attributes which must be specified by matching Import-Package statements in the Export-Package manifest header.

The directive value is encoded in the Export-Package manifest header like:

```
Export-Package: org.osgi.framework; mandatory:="bundle-symbolic-name"
```

Since:
1.3
See Also:
Constant Field Values

5.4.23 VISIBILITY_DIRECTIVE

`public static final java.lang.String` **`VISIBILITY_DIRECTIVE`**

Manifest header directive (named "visibility") identifying the visibility of a reqiured bundle in the Require-Bundle manifest header.

The directive value is encoded in the Require-Bundle manifest header like:

```
Require-Bundle: com.acme.module.test; visibility:="reexport"
```

Since:
1.3
See Also:
`VISIBILITY_PRIVATE`, `VISIBILITY_REEXPORT`, Constant Field Values

5.4.24 VISIBILITY_PRIVATE

`public static final java.lang.String` **`VISIBILITY_PRIVATE`**

All Page Within This Box

Manifest header directive value (named "private") identifying a private visibility type. A private visibility type indicates that any packages that are exported by the required bundle are not made visible on the export signature of the requiring bundle.

The directive value is encoded in the Require-Bundle manifest header like:

```
Require-Bundle: com.acme.module.test; visibility:="private"
```

Since:
1.3
See Also:
VISIBILITY_DIRECTIVE, Constant Field Values

5.4.25 VISIBILITY_REEXPORT

`public static final java.lang.String` **`VISIBILITY_REEXPORT`**

Manifest header directive value (named "reexport") identifying a reexport visibility type. A reexport visibility type indicates any packages that are exported by the required bundle are re-exported by the requiring bundle. Any arbitrary arbitrary matching attributes with which they were exported by the required bundle are deleted.

The directive value is encoded in the Require-Bundle manifest header like:

```
Require-Bundle: com.acme.module.test; visibility:="reexport"
```

Since:
1.3
See Also:
VISIBILITY_DIRECTIVE, Constant Field Values

5.4.26 REEXPORT_PACKAGE

`public static final java.lang.String` **`REEXPORT_PACKAGE`**

Manifest header (named "Reexport-Package") identifying the names of the packages that the bundle offers to the Framework for reexport.

The attribute value may be retrieved from the `Dictionary` object returned by the `Bundle.getHeaders` method.

Since:
1.3
See Also:
Constant Field Values

5.5 FrameworkEvent Changes

The R3 spec has many places where it is mandatory or suggested that a FrameworkEvent of type ERROR be published. Sometimes the condition is not an error but just need to be brought to someone's attention.

5.5.1 WARNING

`public static final int` **`WARNING`**

A warning has occurred.

There was a warning associated with a bundle.

The value of `WARNING` is 0x00000010.

Since:
1.3
See Also:
Constant Field Values

5.5.2 INFO

`public static final int` **`INFO`**

An informational event has occurred.

There was an informational event associated with a bundle.

The value of `INFO` is 0x00000020.

Since:
1.3
See Also:
Constant Field Values

5.5.3 Other Specification Changes For New Event Types

R3 Specification section	Subject	Change
4.4.9 (Replaced by RFC 70)	Bundle-Classpath missing entries	Change to INFO
4.19.1	Framework startup	Leave as ERROR
4.19.2	Framework shutdown	Leave as ERROR
4.23.3.17	Bundle.uninstall	Leave as ERROR
4.23.3.18	Bundle.update	Leave as ERROR
4.23.5.11	BundleContext.getService	Leave as ERROR
5.5.3.3	PackageAdmin.refreshPackages	Leave as ERROR
6.2.2	Changing active start level	Leave as ERROR
6.2.7	Exceptions in BundleActivator	Leave as ERROR
9.6.3	Framework Events Mapping	Need to change to support new FrameworkEvent types.

All Page Within This Box

5.6 PackageAdmin Changes

These changes are to support the new features in RFC 70 and RFC 79.

5.6.1 getExportedPackage

This method has the same signature but the semantics have changed because now we can have multiple versions of the same package exported in the Framework.

`public ExportedPackage getExportedPackage(java.lang.String name)`

Gets the `ExportedPackage` object with the specified package name. All exported packages will be checked for the specified name. The exported package with the highest version will be returned.

In an environment where the exhaustive list of packages on the system classpath is not known in advance, this method attempts to see if the named package is on the system classpath. This means that this method may discover an `ExportedPackage` object that was not present in the list returned by a prior call to `getExportedPackages()`.

Parameters:
`name` - The name of the exported package to be returned.
Returns:
The exported package with the specified name, or `null` if no exported packages with that name exists.

5.6.2 getAllExportedPackages

NOTE: we cannot overload the getExportedPackages here because it is common for clients to pass null to the existing PackageAdmin.getExportedPackages(Bundle) method. If we overload getExportedPackages then clients passing null will get ambiguous method reference errors.

`public ExportedPackage[] getAllExportedPackages(java.lang.String name)`

Gets all the `ExportedPackage` objects with the specified package name. All exported packages will be checked for the specified name.

In an environment where the exhaustive list of packages on the system classpath is not known in advance, this method attempts to see if the named package is on the system classpath. This means that this method may discover an `ExportedPackage` object that was not present in the list returned by a prior call to `getExportedPackages()`.

Parameters:
`name` - The name of the exported packages to be returned.
Returns:
An array of the exported packages with the specified name, or `null` if no exported packages with that name exists.
Since:
1.2

5.6.3 resolveBundles

`public boolean resolveBundles(Bundle[] bundles)`

Resolve the specified bundles. The Framework must attempt to resolve the specified bundles that are unresolved. Additional bundles that are not included in the specified bundles may be resolved as a result of calling this method. A permissible implementation of this method is to attempt to resolve all unresolved bundles installed in the framework.

All Page Within This Box

If `null` is specified then the Framework will attempt to resolve all unresolved bundles. This method must not cause any bundle to be refreshed, stopped, or started. This method will not return until the operation has completed.

Parameters:
`bundles` - The bundles to resolve or `null` to resolve all unresolved bundles installed in the Framework.
Returns:
`true` if all specified bundles are resolved;
Since:
1.2

5.6.4 getRequiredBundles

```
public RequiredBundle[] getRequiredBundles(java.lang.String symbolicName)
```

Returns an array of RequiredBundles with the specified symbolic name. If the symbolic name argument is `null` then all RequiredBundles are returned.
Parameters:
`symbolicName` - The symbolic name of the RequiredBundle or `null` for all RequiredBundles in the Framework.
Returns:
An array of RequiredBundles with the specified symbolic name or `null` if no RequiredBundles exist with that symbolic name.
Since:
1.2

5.6.5 getBundles

```
public Bundle[] getBundles(java.lang.String symbolicName,
                           java.lang.String versionRange)
```

Returns the bundles with the specified symbolic name within the specified version range. If no bundles are installed that have the specified symbolic name, then `null` is returned. If a version range is specified, then only the bundles that have the specified symbolic name and belong to the specified version range are returned. The returned bundles are ordered by version in descending version order so that the first element of the array contains the bundle with the highest version.
Parameters:
`symbolicName` - The symbolic name of the desired bundles.
`versionRange` - The version range of the desired bundles, or `null` if all versions are desired.
Returns:
An array of bundles with the specified name belonging to the specified version range ordered in descending version order, or `null` if no bundles are found.
Since:
1.2
See Also:
`Constants.BUNDLE_VERSION_ATTRIBUTE`

5.6.6 getFragments

```
public Bundle[] getFragments(Bundle bundle)
```

Returns an array of attached fragment bundles for the specified bundle. If the specified bundle is a fragment then `null` is returned. If no fragments are attached to the specified bundle then `null` is returned.

All Page Within This Box

Parameters:
`bundle` - The bundle whose attached fragment bundles are to be returned.
Returns:
An array of fragment bundles or `null` if the bundle does not have any attached fragment bundles.
Since:
1.2

5.6.7 getHosts

`public Bundle[] getHosts(Bundle bundle)`

Returns an array of host bundles to which the specified fragment bundle is attached or `null` if the specified bundle is not attached to a host or is not a fragment bundle.
Parameters:
`bundle` - The bundle whose host bundles are to be returned.
Returns:
An array of host bundles or `null` if the bundle does not have any host bundles.
Since:
1.2

5.6.8 getBundle

`public Bundle getBundle(java.lang.Class clazz)`

Returns the bundle for which the specified class is loaded from. The classloader of the bundle returned must have been used to load the specified class. If the class was not loaded by a bundle classloader then `null` is returned.
Parameters:
`clazz` - the class object to get a bundle for
Returns:
the bundle from which the specified class is loaded or `null` if the class was not loaded by a bundle classloader
Since:
1.2

5.6.9 BUNDLE_TYPE_FRAGMENT

`public static final int BUNDLE_TYPE_FRAGMENT`

The bundle is a fragment bundle.

The value of `BUNDLE_TYPE_FRAGMENT` is 0x00000001.

Since:
1.2
See Also:
Constant Field Values

5.6.10 getBundleType

`public int getBundleType(Bundle bundle)`

Returns the special type of the specified bundle. The bundle type values are:

- `BUNDLE_TYPE_FRAGMENT`

All Page Within This Box

A bundle may be more than one type at a time. A type code is used to identify the bundle type for future extendability.

If a bundle is not one or more of the defined types then 0x00000000 is returned.

Returns:
The special type of the bundle.
Since:
1.2

5.7 New RequiredBundle Interface

public interface **RequiredBundle**

A required bundle. Instances implementing this interface are created by the Package Admin service.

The information about a `RequiredBundle` provided by this object is valid only until the next time `PackageAdmin.refreshPackages()` called. If a `RequiredBundle` object becomes stale (that is, the bundle it references has been updated or removed as a result of calling `PackageAdmin.refreshPackages()`), its `getSymbolicName()` and `getVersion()` continue to return their old values, `isRemovalPending()` returns true, and `getBundle()` and `getRequiringBundles()` return `null`.

Since:
1.2

Method Summary

Bundle	**getBundle**() Returns the bundle which defines this RequiredBundle.
Bundle[]	**getRequiringBundles**() Returns the resolved bundles that currently require this bundle.
java.lang.String	**getSymbolicName**() Returns the symbolic name of the bundle.
java.lang.String	**getVersion**() Returns the version of the bundle.
boolean	**isRemovalPending**() Returns `true` if the bundle has been updated or uninstalled.

Method Detail

5.7.1 getBundle

`public Bundle` **`getBundle`**`()`

Returns the bundle which defines this RequiredBundle.
Returns:
The bundle, or `null` if this `RequiredBundle` object has become stale.

5.7.2 getRequiringBundles

`public Bundle[] getRequiringBundles()`

Returns the resolved bundles that currently require this bundle. If this `RequiredBundle` object is required and re-exported by another bundle then all the requiring bundles of the re-exporting bundle are included in the returned array.

Returns:

An array of resolved bundles currently requiring this bundle, or `null` if this `RequiredBundle` object has become stale.

5.7.3 getSymbolicName

`public java.lang.String getSymbolicName()`

Returns the symbolic name of the bundle.

Returns:

The symbolic name of the bundle.

5.7.4 getVersion

`public java.lang.String getVersion()`

Returns the version of the bundle.

Returns:

The version of the bundle.

5.7.5 isRemovalPending

`public boolean isRemovalPending()`

Returns `true` if the bundle has been updated or uninstalled.

Returns:

`true` if the bundle has been updated or uninstalled, or if the `RequiredBundle` object has become stale; `false` otherwise.

5.8 ExportedPackage Changes

5.8.1 getImportingBundles

This method has the same signature but the semantics have changed because now exporters do not automatically import their exported package and requiring bundles automatically import the exports from a required bundle.

`public Bundle[] getImportingBundles()`

Returns the resolved bundles that are currently importing the package associated with this `ExportedPackage` object.

Bundles which require the exporting bundle associated with this `ExportedPackage` object are considered to be importing bundles and are included in the returned array. See `RequiredBundle.getRequiringBundles()`

Returns:

All Page Within This Box

The array of resolved bundles currently importing the package associated with this `ExportedPackage` object, or `null` if this `ExportedPackage` object has become stale.

5.8.2

6 Considered Alternatives

6.1 BundlePermission

The BundlePermission class has been moved to RFC 73.

-

6.2 New Version Class

public final class **Version**
extends java.lang.Object
implements java.lang.Comparable

Version identifier. Version instances are immutable.

6.2.1 emptyVersion

```
public static Version emptyVersion
```

The empty version "0.0.0". Equivalent to calling new Version(0,0,0).

6.2.2 Version

```
public Version(int major,
               int minor,
               int micro)
```

Creates a version identifier from its components.

Parameters:

`major` - major component of the version identifier
`minor` - minor component of the version identifier
`micro` - micro update component of the version identifier

6.2.3 Version

```
public Version(int major,
               int minor,
               int micro,
               java.lang.String qualifier)
```

All Page Within This Box

Creates a version identifier from its components.

Parameters:

`major` - major component of the version identifier
`minor` - minor component of the version identifier
`micro` - micro update component of the version identifier
`qualifier` - qualifier component of the version identifier

6.2.4 Version

`public` **`Version`**`(java.lang.String version)`

Creates a version identifier from the given string.

Parameters:

`version` - string representation of the version identifier.

6.2.5 equals

`public boolean` **`equals`**`(java.lang.Object object)`

Compare version identifiers for equality. Identifiers are equal if all of their components are equal.

Overrides:

`equals` in class `java.lang.Object`

Parameters:

`object` - an object to compare

Returns:

whehter or not the two objects are equal

6.2.6 hashCode

`public int` **`hashCode`**`()`

Returns a hash code value for the object.

Overrides:

`hashCode` in class `java.lang.Object`

Returns:

an integer which is a hash code value for this object.

6.2.7 getMajor

`public int` **`getMajor`**`()`

Returns the major component of this version identifier.

Returns:

the major version

6.2.8 getMinor

`public int` **`getMinor`**`()`

Returns the minor component of this version identifier.

Returns:

the minor version

All Page Within This Box

6.2.9 getMicro

`public int` **`getMicro`**`()`

Returns the micro level component of this version identifier.

Returns:

the micro level

6.2.10 getQualifier

`public java.lang.String` **`getQualifier`**`()`

Returns the qualifier component of this version identifier.

Returns:

the qualifier

6.2.11 toString

`public java.lang.String` **`toString`**`()`

Returns the string representation of this version identifier.

Overrides:

`toString` in class `java.lang.Object`

Returns:

the string representation of this version identifier

6.2.12 compareTo

`public int` **`compareTo`**`(java.lang.Object o)`

Compares this Version object with the specified Version object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Specified by:

`compareTo` in interface `java.lang.Comparable`

Parameters:

o - the Version object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified Version object.

Throws:

`java.lang.ClassCastException` - if the specified object's type is not Version.

---

7 Security Considerations

All Page Within This Box

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]. RFP 44 Framework: Next Generation, http://membercvs.osgi.org/rfps/rfp-0044-FrameworkNG.doc

[4]. RFP 48 Security Enhancements, http://membercvs.osgi.org/rfps/rfp-0048-Security%20Enhancements.doc

[5]. RFC 70 Bundle Classloading Changes, http://membercvs.osgi.org/rfcs/rfc0070

8.2 Author's Address

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

All Page Within This Box