DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

DATE OF DISTRIBUTION: 6 April 2010

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the “Distribution”) in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree not to participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. “OSGi Name Space” shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED “AS IS,” AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS. NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable, worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future (“Future Specification”), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license...
under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.
Preface

This document is Early Draft 1 of the OSGi Service Platform Release 4 Core Version 4.3 specifications. As an early draft, it contains non-final specification work and it is not organized in the format normally associated with final release OSGi specifications. This document contains copies of OSGi design documents which either propose to modify existing published OSGi specifications from the OSGi Service Platform Release 4 Version 4.2 specification documents or propose new specifications to potentially be incorporated in the final OSGi Service Platform Release 4 Core Specification Version 4.3 documents.

Since this early draft is not a complete specification document, the reader is expected to be familiar with OSGi Technology and the currently published OSGi Service Platform Release 4 Version 4.2 specification documents. The reader should refer to http://www.osgi.org/About/Technology for more information on the OSGi Technology. There the reader can find a description of the OSGi Technology, as well as links to whitepapers and the OSGi Service Platform Release 4 Version 4.2 specification documents, which are all available for download.

Pursuant to the Distribution and Feedback License above, the OSGi expert groups welcome your feedback on this early draft. Feedback can be provided by opening a bug at https://www.osgi.org/bugzilla/enter_bug.cgi?product=OSGi%20Specification.

BJ Hargrave
Chief Technical Officer
OSGi Alliance
RFC 0138 Multiple Frameworks In One JVM

Draft

58 Pages

Abstract

This document explores the complexity of running more than one OSGi framework in a single JVM process, and proposes a set of requirements to support that environment.
0 Document Information

0.1 Table of Contents

0 Document Information..................................................................................................................2
   0.1 Table of Contents................................................................................................. 2
   0.2 Terminology and Document Conventions................................................................. 4
   0.3 Revision History......................................................................................................... 4

1 Introduction..............................................................................................................................7

2 Application Domain..................................................................................................................8
   2.1 Behavior of multiple frameworks in one JVM.............................................................. 8
   2.2 Terminology + Abbreviations....................................................................................... 9

3 Problem Description................................................................................................................10
   3.1 Using multiple frameworks for isolation..................................................................... 10
      3.1.1 Use case 1: Runtime/Application isolation............................................................ 10
      3.1.2 Use case 2: Construction of composites................................................................. 11
      3.1.3 Use case 3: Application scoping/isolation............................................................... 11
      3.1.4 Use case 4: Tight Control Over Imported Capabilities From Peers....................... 11

4 Requirements..........................................................................................................................12

5 Technical Solution..................................................................................................................13
   5.1 Composite Bundle........................................................................................................ 15
      5.1.1 Bundle methods................................................................................................... 16
      5.1.2 Composite bundle meta-data................................................................................ 17
   5.2 Installing Composite Bundles........................................................................................ 19
      5.2.1 Installing Composite meta-data........................................................................... 19
      5.2.2 Constituent bundles.............................................................................................. 19
   5.3 Sharing Policy................................................................................................................ 20
      5.3.1 Nested Composites................................................................................................ 21
      5.3.2 Resolve Time Sharing Policy.................................................................................. 21
      5.3.3 Service Sharing Policy.......................................................................................... 29
      5.3.4 Transparency of sharing........................................................................................ 30
      5.3.5 Priority of shared capabilities.............................................................................. 31
      5.3.6 Sibling Composite Constraints.......................................................................... 31
   5.4 Lifecycle of composite bundles....................................................................................... 33
   5.5 Impacts on the system bundle..........................................................................................34
      5.5.1 Core Services......................................................................................................... 34
      5.5.2 Framework Extension Bundles............................................................................. 37
   5.6 Configuration Properties..................................................................................................37
      5.6.1 BundleContext Properties.................................................................................... 37

Copyright © IBM Corporation 2010 All Rights Reserved
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 9.1.

Source code is shown in this typeface.

0.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>AUG 12 2008</td>
<td>Initial Draft (boiler plate)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Erin Schnabel, IBM  <a href="mailto:schnabel@us.ibm.com">schnabel@us.ibm.com</a></td>
</tr>
<tr>
<td>Additional text</td>
<td>SEP 05 2008</td>
<td>First pass at description of solution</td>
</tr>
<tr>
<td>Piscataway F2F</td>
<td>SEP 18 2008</td>
<td>Clarification of parent-child resource sharing (description of resources,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>relationships between the framework, etc.).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Glyn Normington, SpringSource,  <a href="mailto:glyn.normington@springsource.com">glyn.normington@springsource.com</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Erin Schnabel, IBM,  <a href="mailto:schnabel@us.ibm.com">schnabel@us.ibm.com</a></td>
</tr>
<tr>
<td>Service changes</td>
<td>SEP 26 2008</td>
<td>Separation of concerns: FrameworkService for starting a child framework, a FrameworkBridgeService for sharing resources between frameworks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Erin Schnabel, IBM,  <a href="mailto:schnabel@us.ibm.com">schnabel@us.ibm.com</a></td>
</tr>
<tr>
<td>Service changes</td>
<td>OCT 20 2008</td>
<td>FrameworkFactory: createChildBundle, createFrameworkLink; FrameworkLink instead of Bridge;</td>
</tr>
<tr>
<td>(from RI)</td>
<td></td>
<td>LinkDescription instead of VisibilityFilter. Addition of javadoc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Erin Schnabel, IBM,  <a href="mailto:schnabel@us.ibm.com">schnabel@us.ibm.com</a></td>
</tr>
<tr>
<td>API updates</td>
<td>NOV 04 2008</td>
<td>Clean up for public draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bug 829: using collections rather than arrays (bug 827), changing SystemBundle to Framework,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>method names on FrameworkFactory should start with “new”, clarify properties used for registration of link service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bug 828: description of service dynamics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Erin Schnabel, IBM,  <a href="mailto:schnabel@us.ibm.com">schnabel@us.ibm.com</a></td>
</tr>
<tr>
<td>API revision</td>
<td>NOV 11 2008</td>
<td>Reversion to strict parent-child relationship. Simplify to a single factory method: a child framework is always created with two link bundles, regardless of whether or not resources are shared.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Erin Schnabel, IBM,  <a href="mailto:schnabel@us.ibm.com">schnabel@us.ibm.com</a></td>
</tr>
<tr>
<td>Revision</td>
<td>Date</td>
<td>Comments</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| API revision  | DEC 10 2008| LinkBundle -> CompositeBundle  
bug 781: Changes to make FrameworkUtil.createFilter, AdminPermission, BundleSignerCondition framework agnostic.  
bug 863: AllPermission required to create a nested/child framework; framework implementation must ensure correct permissions on composite bundles.  
bug 881: New method on Bundle interface to get a bundle's signer DN chain.  
bug 905: BundleException if non-CompositeBundle already installed w/ provided location string; clarification of allowed manifest headers for composite bundles  
bug 906: first pass at concrete use cases for creating child frameworks |
| API finalization | DEC 30 2008| Section 3.1 for isolation use cases (bug 906)  
API Finalization: CompositeBundle, SurrogateBundle, CompositeBundleFactory  
bug 813: Composite and surrogate bundle start level  
Erin Schnabel, IBM, schnabel@us.ibm.com |
| Additional revisions | JAN 12 2009| bug 765: headers returned by the system bundle in getHeaders  
bug 814: sharing system bundle packages (section 5.3)  
bug 904: behavior of attributes and directives (section 5.3)  
bug 906: three concrete use cases for multiple frameworks in a VM (section 3.1)  
bug 909: singleton interface for multiplexing (section 5.4.1 + javadoc)  
Erin Schnabel, IBM, schnabel@us.ibm.com  
Tom Watson, IBM, tjwatson@us.ibm.com |
| post-CPEG     | JAN 13 2009| Mark multiplexer API “Provisional” as well  
Constants for method names added to multiplexer API  
Remove “framework” from multiplexer package name  
Add static helper that provides basic implementation of reflection to check for osgi support and to maintain/traverse the double-linked list.  
|
| Fixing bugs   | 07/23/09   | Fixed bug 1014 in section 5.3.2  
<p>|</p>
<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent Sharing Design</td>
<td>09/24/09</td>
<td>Redesign of solution:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Remove access to Framework object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Improve lifecycle of composite framework</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Design does not assume real nested framework for implementation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Transparent sharing policy for direct wiring of shared resources.</td>
</tr>
<tr>
<td></td>
<td>10/05/09</td>
<td>Added comments from David Bosschaert</td>
</tr>
<tr>
<td></td>
<td>10/29/09</td>
<td>• Clarified that BundleEvents and Bundle objects are not shared across composite boundaries.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarified that a composite framework beginning start-level is set while the Composite bundle is in the STARTING state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• When with the simple null option for composite loadClass, get/findEntry getResource methods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specified what constitutes an invalid composite manifest and that install must fail if the composite manifest is invalid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed getFrameworkUUID() method. This will belong on the BundleContext/Framework interface eventually</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarified that each composite framework will have a system bundle and that system bundle must have a bundle id of 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarified the visibility rules for capabilities exported out of child composites</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarified the sharing rules between multiple nested frameworks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarified that the bundle require policy does not automatically import all exported packages from a required bundle for importing by constituents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added wild card matching for package sharing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarified singleton resolution in composites.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Clarified that the composite system bundle context must remain valid as long as the composite bundle is installed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Defined the behavior of service hooks in composites.</td>
</tr>
</tbody>
</table>

Tom Watson, IBM, tjwatson@us.ibm.com
1 Introduction

The OSGi platform is increasingly being used as a basis for composing complex runtime environments. Middleware providers are using OSGi implementations as the foundation of their runtimes, while application developers are looking to use OSGi standards and artifacts to build enterprise applications. As these two uses converge, questions arise over how to isolate applications from the runtime, or even certain applications from other applications.
This RFP has its origins in the CPEG conference call on 11 April, 2008. In that call a new RFC (RFC 136: “Scopes and OSGi”) was discussed for the first time, and RFC 132 (“Command Line Interface”) was reviewed. RFC 136 is a response to RFP 100 (“Visibility Scoping of Exported Packages”), and outlines a proposal for defining visibility scopes within a framework. RFC 132 defines a common mechanism for booting both standalone and embedded OSGi frameworks. Nested frameworks (or even peer frameworks) in the same JVM were mentioned during the discussion of both RFCs: nested or peer frameworks provide an implicit scoping mechanism with additional isolation characteristics (RFC 136/RFP 100), and would be booted using those mechanisms defined in RFC 132.

Initial investigation of using RFC 132 launching as a means for launching a nested framework for a composite bundle has proved to introduce some interesting lifecycle issues. A rework if this RFC has been done to try and hide the composite framework as an implementation detail of a composite bundle.

---

### 2 Application Domain

As noted above, having multiple frameworks in a single JVM has been mentioned tangentially in relationship to several RFPs and RFCs. Some elements of the OSGi specification already apply to and would support multi-framework environments:

- A parent framework can provide classes to a nested framework using an existing property to specify what the nested framework's system bundle must export (org.osgi.framework.system.packages). This property does not currently define the packages a parent framework provides. Instead it tells the framework what packages from the class loader that loaded the framework should be exported by the system bundle.

- RFC 132 defines a mechanism for setting properties for a particular framework instance. Without that mechanism, the BundleContext interface only provides a mechanism to get properties, not to set them. Note that this issue is not as important in the composite solution because you only provide configuration parameters to launch the root (parent) framework using RFC 132 to launch.

- RFC 132 also provides a mechanism for controlling the state of embedded frameworks (start, stop), and defines rules for managing a framework's lifecycle. For Composite bundles the lifecycle of the composite framework should be an implementation detail. A composite bundle's lifecycle is related to the constituent bundles and the composite framework. This relationship must be clearly defined.

#### 2.1 Behavior of multiple frameworks in one JVM

A framework instance is a discrete unit: it has its own class loading policy, and is responsible for resolving class loading constraints for all bundles that it "owns" (all bundles installed for that framework).

If there are multiple frameworks in a JVM:

- There may be more than one instance of a singleton bundle active in the JVM: each framework manages its installed bundles independently from the other frameworks— one framework should not have to know that a singleton bundle has been instantiated by another framework in the same JVM.
➢ The Java mechanisms for extending the URL class allow handler factories to be specified once per JVM process. As bundles can have a much shorter life cycle than the JVM process as a whole, the URL Handlers Service Specification requires frameworks to register `URLStreamHandlerFactory` and `ContentHandlerFactory` objects to act as proxies for handlers that are registered with the framework as services. If multiple frameworks are present in a JVM, some mediation would have to occur to ensure that the single `URLStreamHandlerFactory` and `ContentHandlerFactory` can multiplex between available frameworks.

➢ There is only one **Security Manager** reachable via the `java.lang.System` class. The Conditional Permission Admin (CPA) service interacts directly with the configured Security Manager to perform permission checks. The impact of any mechanism to adapt the Security Manager to a multi-framework environment on CPA condition evaluation would have to be well understood. At a base level, the discrete nature of each framework should be maintained: each framework should be able to establish, maintain, and evaluate its conditions, permissions and bundle protection domains independently and without knowledge of any other framework. Postponed conditions cannot be supported without multiplexing the Security Manager.

➢ Issues with class loader parenting and use of the context class loader are not new, nor are they particular to a multi-framework environment.

### 2.2 Terminology + Abbreviations

➢ **Embedded framework**: An OSGi framework started from within an existing Java process.

➢ **Nested framework**: An embedded OSGi framework started by an OSGi bundle. The lifecycle of a nested framework is bounded by the lifecycle of the launching bundle (in other words, it is the launching bundle's responsibility to stop and clean up the framework when it is stopped).

➢ **Hosting framework**: An OSGi framework that contains other framework instances. Since nested frameworks (in general) are started by arbitrary bundles, the hosting framework may not be aware of nested instances.

➢ **Child framework**: A special case of nested framework-- an OSGi framework started by an OSGi framework. The lifecycle of a child framework is bounded by the lifecycle of the parent framework.

➢ **Parent framework**: A special case of hosting framework-- an OSGi framework that owns and manages child framework instances.

➢ **Peer framework**: An embedded framework at the same level of nesting as another embedded framework in a multi-framework environment: e.g. two nested frameworks started by the same bundle, or two child frameworks started by the same parent.

➢ **Multi-framework environment**: a JVM containing more than one framework, regardless of how those frameworks are related to each other.

➢ **Composite Bundle**: A bundle which is composed of a set of bundles call constituents.

➢ **Constituent Bundle**: A bundle which composes the content of a composite bundle.

➢ **Composite Framework**: A special case of child framework – an OSGi framework associated with a composite bundle and is bound to the lifecycle of that composite bundle. Constituent bundles are installed in a composite framework.
3 Problem Description

Multiple frameworks cannot easily be present in the same JVM due to some of the constraints mentioned above:

- There isn't a uniform, framework-agnostic way to deal with JVM Singletons like the registered URLStream and ContentHandler factories for URLs. Some framework implementations (Eclipse Equinox, Apache Felix) have invented multiplexing handler factories that a) only handle frameworks of from the same vendor, b) use reflection to ensure the right handler factories are being used, and c) rely on their custom handler factories to mediate calls to factory methods between frameworks. There are synchronization and resource management issues with this approach (what happens when a framework is restarted, etc).

- There isn't a clear definition of how packages and/or services would or could be shared between multiple frameworks in the same JVM, or how managing the lifecycle of one framework would impact other frameworks that are using resources provided by that framework.

Standardizing a set of rules and expected behavior will allow multiple frameworks, potentially provided by different vendors, to behave and interact consistently when run within the same JVM.

3.1 Using multiple frameworks for isolation

As mentioned briefly in the introduction, using embedded frameworks provides an implicit scoping mechanism with strong isolation characteristics: without the creation of special mechanisms to share resources, each embedded framework provides an isolated scope for OSGi artifacts and resources (separate service registries, distinct class spaces, independent event notifications, etc.).

3.1.1 Use case 1: Runtime/Application isolation

There is a requirement for this kind of isolation in the enterprise environment: many enterprise middleware providers are using OSGi as a foundation for composable middleware runtimes. At the same time, enterprise application developers want to use OSGi to develop and manage the lifecycle of their applications. There are several issues caused by the middleware runtime and OSGi applications sharing the same OSGi framework:

- Accidental creation of API: Middleware providers often create infrastructure services meant only to support management and provisioning of the runtime itself. Because the API for these services are shared (and published to the OSGi registry), there isn't a good way to demarcate “internal” API from “external” API, short of extensive use of security (which has performance implications), or reliance on brittle package or bundle decorations, attribute matching, or framework-specific bundle/package resolution mechanisms.

- Conflicts over use of third party libraries: Some third party libraries may be required by both the middleware runtime and enterprise OSGi applications. Without some isolation between the runtime and applications, the runtime’s use of third party libraries (and versioning requirements, etc.) could significantly impact and restrict the libraries available for use by OSGi applications.

This presents a use case for a nested framework approach: assume Framework A is an enterprise-level middleware runtime that is OSGi-based and composed from a set of proprietary bundles. The middleware provider could create a nested/child Framework B to host an arbitrary number of deployed applications, and should ideally be able to share with it a well-defined list of packages and services that are then available for use.
by the applications running in that inner framework. Aside from those explicitly shared resources, the two frameworks remain completely independent.

The child framework in this use case is long-lived, and the creator of the child framework (the parent middleware runtime) is not aware at creation time of the bundles that will be used by the child framework: as applications are installed or removed, the list of bundles installed in the child framework will change.

3.1.2 Use case 2: Construction of composites

Another use case for nested frameworks is the definition and use of “Composites”: the composite would declaratively specify the packages and services it would provide (as well as the packages and services it requires), but would otherwise provide isolation between the user of the composite, and the components providing the composite's implementation.

For example, a declarative composite definition could look something like this:

```java
composite {
    Install-Bundle: foo; bundle-version=[1.0.0,1.2.0),
    bar; bundle-version=[1.0.0,1.0.0]
    Import-Package: waz; version=2.0.0
    Export-Package: baz; version=1.0.1
}
```

This composite could be implemented using a child framework: package waz would be shared with the child framework (just as with the first use case), but here, the child will also be exporting package baz back to the parent. From the point of view of other bundles in the parent, the composite should look like any other bundle that imports or exports packages or that uses or provides services. The parent bundles should have no awareness of what bundles the child is using to provide package baz. Bundles in the child framework, similarly, have no view of the bundles used by the parent framework to provide package waz.

Unlike the first use case, the composite is intended to provide packages and services to the hosting framework, and the full set of bundles that provide the composite implementation are known at declaration time.

3.1.3 Use case 3: Application scoping/isolation

Applications need to be isolated from each other. It should be possible to install multiple applications and be assured that they will not interact in unexpected ways (i.e. it is necessary to selectively prevent packages and services being resolved across applications). In the same way a runtime may want to have private infrastructure services and libraries, applications may also have services and libraries intended to be used/resolved internally; they may also depend on different/conflicting third-party library versions and therefore must execute in isolated frameworks in order for each application to resolve to the libraries it requires. The runtime/hosting environment may also provide packages and services to the isolated applications.

3.1.4 Use case 4: Tight Control Over Imported Capabilities From Peers

This is similar to use case 3 where an application needs to be isolated from other applications (as well as the middleware) but additional control is needed by the middleware to ensure an application can be configured to use a particular shared capability installed outside of the scope of the application.

In this scenario some application X must be isolated from other applications and the middleware. This can be implemented by using a nested framework from use case 1 or a composite from use case 2. In this case application X has a dependency on library Y from vendor A and on library Z from vendor B. The middleware has several other applications that depend on library Y from vendor C and library Z from vendor D. The middleware needs the ability to install both versions of library Y (from vendors A and B) and both versions of Z (from vendors A and D).
C and D) in such a way that they can be isolated from each other and used by one or more isolated applications. The following diagram illustrates this scenario:

In this scenario there are 4 applications X, P, Q and R and each is configured to use a different combination of vendors for libraries Y and Z. The use case requires for maximum amount of sharing for the different library implementations used by the various applications. One solution could be to place each library required by an application within the individual application scope but this reduces the amount of sharing greatly. Another solution would be to place all applications that use the same library version into a hierarchy where the applications parent could contain the common library version but this limits the ability to configure an application to arbitrarily select different vendors for the libraries they require. All applications sharing a common hierarchy would need to be configured to use the same library vendors. For example, this may not be possible for any of the applications above because they all are configured to use a different combination of library vendors for libraries Y and Z.

### 4 Requirements

1. The solution MUST define the parent class loader of child frameworks. REMOVE – Not sure why this is a must. It forces the implementation to create a physically different framework for the composite bundle.

2. The solution MUST describe how the lifecycle of composite frameworks are managed and how parent relationships are impacted by framework state changes.

3. The solution MUST allow frameworks to manage the lifecycle of their installed bundles independently from the lifecycle of bundles in other frameworks.
4. The solution MUST define how framework services, like the Conditional Permission Admin service, interact with the JVM singleton Security Manager.

5. The solution MUST provide a mechanism for parent frameworks to provide resources (e.g. packages and services, though potentially other resources as well) to composite frameworks, including defining how state changes to resource providers in the parent framework impact resource consumers in the composite framework.

6. The solution MAY provide a mechanism for composite frameworks to provide resources (packages and services) to its parent, including defining how state changes to resource providers in a child framework impact resource consumers in the parent framework.

---

## 5 Technical Solution

**NOTE:** Prior versions of this RFC assumed the use of nested frameworks to provide isolation. This RFC has been rewritten to remove this assumption. The use of a nested framework is an implementation detail to provide the necessary isolation. But it is not the ONLY way we can envision this solution to be implemented. There are disadvantages to using a real nested framework that prevent certain scenarios and also introduce tricky lifecycle issues.

This RFC introduces the composite bundle to provide the necessary isolation for a group of bundles. At a high level, a composite is a bundle for which the content is composed of meta-data describing the composite, its sharing policy and a set of bundles called constituent bundles. Constituent bundles are isolated from other bundles which are not constituents of the same composite bundle. Composite bundles provide isolation for constituent bundles in the following ways:

- **Class space:** By default constraints specified by constituent bundles (e.g. Import-Package, Require-Bundle, Fragment-Host) can only be resolved against capabilities provided by other constituents within the same composite bundle (Export-Package, Bundle-SymbolicName/Bundle-Version). A composite sharing policy may be used to share class space capabilities across composite boundaries.

- **Service Registry:** By default constituent bundles only have access to services registered by other constituents within the same composite bundle. This includes service events and service references. A composite sharing policy may be used to share services across composite boundaries.

- **Bundles:** Constituent bundles only have access to other Bundle objects that represent constituents within the same composite bundle. This includes bundle events and core API like BundleContext.getBundles, PackageAdmin and StartLevel. There is no sharing policy for exposing Bundle objects or BundleEvents across composite boundaries.

A parent framework refers to the OSGi framework where a composite bundle is installed. Composites provide isolation for its constituent bundles but there are many cases where capabilities (packages and services) need to be shared between bundles installed in the parent framework and the constituents within a composite bundle. Bundles installed in the parent framework will not have access to the capabilities provided by the constituent
bundles unless the composite sharing policy declares the capabilities as exported out of the composite bundle. Similarly, constituent bundles will not have access to any capabilities provided by the bundles installed in the parent framework unless the composite sharing policy declares the capabilities as imported into the composite bundle.

The figure above illustrates a composite bundle composed of three constituent bundles (A1, A2, A3). The parent framework contains bundles B, C and the composite bundle. The constituent bundles are able to participate in sharing capabilities (packages, services etc.) within the isolated class and service registry space of the composite. The constituents cannot directly interact with the capabilities provided by bundle B or C (or any other bundles) in the parent.

Through a sharing policy, the composite is in control of what capabilities from bundles installed in the parent framework are exposed (imported) into the composite to make them available for constituent bundles. In the figure above the composite's sharing policy allows one of the packages exported by bundle B to be available to constituent bundles and constituent bundle A2 is wired to the exported package from bundle B. Similarly the composite's sharing policy allows the service registered by bundle C to be available to constituent bundles and
constituent bundle A1 has acquired the service registered by bundle C. Details on how a composite's sharing policy controls what capabilities are imported into a composite will be covered later.

Through a sharing policy, the composite is in control of what capabilities from constituent bundles are exposed (exported) out of the composite to make available for bundles installed in the parent framework. In the figure above the composite's sharing policy allows the package exported by constituent bundle A3 to be available to bundles installed in the parent framework and bundle B is wired to the exported package from bundle A3. Similarly the composite's sharing policy allows the service registered by bundle A1 to be available to bundles installed in the parent framework and bundle C has acquired the service registered by bundle A1. Details on how a composite's sharing policy controls what capabilities are exported out of a composite will be covered later.

The lifecycle of constituent bundles are tied to the lifecycle of the composite bundle. For example, constituent bundles are not allowed to be activated until the composite bundle is activated. When a composite bundle is stopped then all constituent bundles are stopped. When a composite bundle is uninstalled then all constituent bundles are uninstalled.

The remainder of this document will go into the details of composite bundles:

- Installing composite bundles and defining the composites sharing policy
- Managing the constituents of a composite bundle
- The lifecycle of a composite bundle and a composite framework
- Impacts on the system bundle and core framework services

## 5.1 Composite Bundle

A composite bundle provides isolation for its constituent bundles. Conceptually the inside of a composite bundle is a framework instance with a set of bundles installed which can collaborate with each other in their own isolated framework. This framework is called the composite framework. One possible implementation is to use a nested framework instance using RFC 132 to launch and control the composite framework (using the org.osgi.framework.launch.Framework API) but that is an implementation detail that is not mandated by this specification. The composite framework is never exposed in the API and all lifecycle operations of the composite framework are done through the composite bundle.

A parent framework is a framework with at least one composite bundle installed. A composite bundle can also be nested in a composite framework making the composite framework a parent framework as well. The root framework is the initial framework which is not a composite framework. Conceptually parent frameworks, composite frameworks and installed constituent bundles are represented with a tree. All bundles reachable by the root framework is referred to as the global tree of bundles.
From a constituent bundle’s perspective they are a normal bundle running in a framework instance. This composite framework instance is represented with its own system bundle just like a real framework is represented by a system bundle. The composite framework has its own service registry, class space and set of bundles. The composite sharing policy is used to either import capabilities from bundles installed in the parent framework into the composite framework or export capabilities from constituent bundles out to the parent framework.

A new interface, called CompositeBundle, is introduced to represent a composite bundle in the parent framework. The composite bundle extends the Bundle interface and adds the following methods:

- **public BundleContext getSystemBundleContext()** - Returns the system bundle context for the composite framework. This method must return a valid BundleContext as long as the CompositeBundle is installed. Once a composite bundle is uninstalled this method must return null. The composite system bundle context can be used to install and manage the constituent bundles.

- **public void update(Map compositeManifest)** – Updates the composite bundles meta-data with the specified manifest.

### 5.1.1 Bundle methods

A composite bundle has divergent behavior for the following Bundle methods:

- **start** – When a composite bundle transitions to the STARTING state all of its constituent bundles are enabled for activation.

- **stop** – When a composite bundle transitions from ACTIVE to STOPPING all of its constituent bundles are disabled for activation and are stopped.

- **uninstall** – During a composite uninstall process all of its constituent bundles must be uninstalled. When a composite bundle’s state transitions from RESOLVED->INSTALLED->UNINSTALLED the constituent
bundles are uninstalled after a composite bundle's state enters the INSTALLED and before it enters the
UNINSTALLED state.

- update methods – Must throw a BundleException with a nested UnsupportedOperationException.
- loadClass – Always throw ClassNotFoundException
- getResource – return null.
- getResources – return null.
- getEntry -- return null.
- getEntryPaths – return null.
- findEntries – return null.

5.1.2 Composite bundle meta-data

There are two categories of composite meta-data

- Composite Identity – provides a unique identity for the composite
- Composite Sharing Policy – specifies a policy for sharing resources/capabilities across composite boundaries.

5.1.2.1 Composite Identity

- Bundle-SymbolicName – the symbolic name of the composite. The directive composite:=true must be specified to identify this bundle as a composite
- Bundle-Version – optionally a composite may specify a bundle version.

5.1.2.2 Composite Sharing Policy

- Composite-PackageImportPolicy – Uses the same syntax as Import-Package. Specifies a list of package constraints to import into the composite. Any exported package from a bundle installed in the parent framework which satisfies one of the specified package constraints is available to satisfy Import-Package constraints from constituent bundles. The Import-Package resolution directive is ignored. Additional directives are supported for specifying a requirement on a peer composite.

- Composite-PackageExportPolicy – Uses the same syntax as Import-Package. Specifies a list of package constraints to export out of a composite. Any exported package from a constituent bundle in the composite which satisfies one of the specified package constraints is available to satisfy Import-Package constraints from bundles installed in the parent framework. The Import-Package resolution directive is ignored.

- Composite-BundleRequirePolicy – Uses the same syntax as Require-Bundle. Specifies a list of require bundle constraints to import into the composite. Any bundle installed in the parent framework which satisfies one of the specified require bundle constraints is available to satisfy Require-Bundle constraints from constituent bundles. Note that this sharing policy only effects the resolution of Require-Bundle constraints. Bundles installed in the parent that match this policy are still not accessible to
constituent bundles through bundle events or BundleContext methods. The Require-Bundle resolution and visibility directives are ignored.

- **Composite-BundleProvidePolicy** – Uses the same syntax as Require-Bundle. Specifies a list of bundle constraints to provide out of a composite. Any constituent bundle installed in the composite which satisfies one of the specified bundle constraints is available to satisfy Require-Bundle constraints from bundles installed in the parent framework. Note that this sharing policy only effects the resolution of Require-Bundle constraints. Bundles installed in the composite that match this policy are still not accessible to bundles installed in the parent framework through bundle events or BundleContext methods. The Require-Bundle resolution and visibility directives are ignored.

- **Composite-ServiceImportPolicy** – Specifies a list of service filters that controls services to import into the composite. Any services registered by bundles installed in the parent framework that match the specified service filter is available to constituent bundles.

- **Composite-ServiceExportPolicy** – Specifies a list of service filters that controls services to export out of the composite. Any services registered by constituent bundles that match the specified service filter is available to bundles installed in the parent framework.

The service sharing policy must conform to the following syntax:

```
Composite-ServiceImportPolicy ::= service-policy ( ',' service-policy )*  
service-policy ::= filters ( ';' parameter )*  
filters ::= filter ( ';' filter )*  
<filter string with no ',' or ';' or '"' chars> | quoted-string
```

The following directives are supported by the Composite-PackageImportPolicy, and the Composite-ServiceImportPolicy and the Composite-BundleRequirePolicy headers:

- composite-symbolic-name – A directive identifying the symbolic name a sibling composite must have that allows capabilities exported from that composite to match an import sharing policy specified by Composite-PackageImportPolicy or Composite-ServiceImportPolicy.

- composite-version – A directive identifying the version a sibling composite must have that allows capabilities exported from that composite to match an import sharing policy specified by Composite-PackageImportPolicy or Composite-ServiceImportPolicy.

Note that the composite-symbolic-name and composite-version directives are not supported on the Composite-BundleRequirePolicy. This is because there is no export policy to allow a Bundle to be exported out of a peer-composite.

### 5.1.2.3 Invalid composite meta-data

A composite bundle must fail to install or update if the new composite meta-data is invalid. The following rules are used to validate composite meta-data:

- The Bundle-SymbolicName key must be specified and must have the composite:=true directive.

- The following must not be included in a composite meta-data
  - DynamicImport-Package
  - Import-Package
5.2 Installing Composite Bundles

Composite bundles are not like normal bundles that have all of their content in a self contained bundle jar. Installing a composite bundle is a two step process to get the following installed:

1. Meta-data declaring the properties of the composite as described in section 5.1.2

2. The constituent bundles which compose a composite bundle

5.2.1 Installing Composite meta-data

A CompositeAdmin service is introduced to provide API to install composite bundles and provide reflective APIs for composite bundles and constituents. The CompositeAdmin service is a core service registered by the system bundle. This service interface contains the following methods:

- `CompositeBundle installCompositeBundle (String location, Map compositeManifest, Map configuration)` – Installs a composite bundle into the parent framework which registered this composite admin service.

- `CompositeBundle getParentCompositeBundle()` - Returns the composite bundle associated with the framework with which this composite admin service is registered. The composite admin service registered in the root framework must return null.

NOTE that getFrameworkUUID has been removed. This method belongs on BundleContext or the new Framework class. Another option is to register a java.lang.String representation of the UUID as a service. Not very nice for clients, but then you could get injected with the framework UUID.

5.2.2 Constituent bundles

Constituent bundles are installed into a composite by using the context returned by the `CompositeBundle.getSystemBundleContext()` method. Any bundles installed using this context will become one of the constituents of the composite bundle and is isolated from any other bundles in the parent framework.

When installing constituent bundles the following rules apply:

- Framework extension bundles must not be allowed to be installed. A BundleException must be thrown when attempting to install a framework extension bundle.
• All bundles within the global tree of bundles must have a unique bundle ID (i.e. Bundle.getBundleId() must be unique). The one exception is the system bundle. Each composite framework must have a system bundle and that system bundle must have a bundle id of 0.

• The bundle IDs within the global tree of bundles must increase according to bundle install order.

• The bundle locations within the global tree of bundles must be unique. If a bundle containing the same location identifier is already installed in the framework where the bundle installation is occurring, the Bundle object for that bundle is returned. The installation must fail with a BundleException if a bundle containing the same location identifier is already installed in another framework in the global tree of bundles.

• The installation of a bundle with a Bundle-SymbolicName and Bundle-Version identical to an existing bundle in the framework where the bundle installation is occurring must fail with a BundleException. Bundles installed in other frameworks in the global tree with a Bundle-SymbolicName and Bundle-Version identical to the bundle being installed does not prevent the installation.

Active constituent bundles will have a valid BundleContext just like normal bundles. Services registered by a constituent bundle must have a unique service ID within the global tree of bundles. In other words a service registered by a constituent must not have the same service ID as any other service registered by any other bundle which is part of the global tree of bundles.

A constituent's bundle context will have a view inside the composite framework. Methods which perform operations on the framework will be isolated to the other constituent bundles and the composites sharing policy:

- **addBundleListener** - BundleListeners added by a constituent bundle will only receive bundle events for other constituent bundles.

- **addServiceListener** - ServiceListeners added by a constituent bundle will only receive service events for the following
  - Services registered by other constituent bundles.
  - Services that match the composite's service import policy.
  - Services that match a service export policy of a child composite.

- **addFrameworkListener** - FrameworkListeners added by a constituent bundle will only receive FrameworkEvents that are relevant to its composite framework.

- **getServiceReference methods** - returns service that meet on of the following requirements
  - Services registered by other constituent bundles
  - Services from the parent framework that match the composite's service import policy.
  - Services that match a service export policy of a child composite.

- **getBundle methods** - return only the constituent bundles.

- **installBundle methods** - installs a constituent bundle into the composite framework.

- **registerService methods** - registers a service in the composite framework's service registry.
5.3 Sharing Policy

A composite bundle declares the sharing policy for what capabilities are imported into the composite and made available for resolving constraints from constituent bundles and what capabilities are exported out of the composite and made available for resolving constraints from bundles installed in the parent framework. The composite bundle itself does not declare capabilities or constraints (Services, Import-Package, Require-Bundle, Fragment-Host, Export-Package etc.). It only defines the policy for sharing capabilities across composite boundaries. The capabilities that can be shared across composite boundaries belong to one of the following types:

- Resolve time – Capabilities defined by Export-Package and Bundle-SymbolicName/Bundle-Version
- Services – Any services registered in the OSGi service registry at runtime

5.3.1 Nested Composites

Composite bundles can be installed in another composite framework. This is referred to as a nested composite. Composite sharing policies can be used to propagate capabilities across arbitrary levels of nested composites. When a composite sharing policy imports a capability then that capability can come from one of the following locations:

- If the import sharing policy does not specify a sibling composite constraint then bundles installed in the parent framework.
- If the import sharing policy does not specify a sibling composite constraint then if the parent framework is a composite framework and that composite framework has an import sharing policy then the capability may come from that parent's parent framework (grand parent framework). Parent delegation can continue all the way up to the root framework as long as each parent framework in the hierarchy has an import sharing policy which matches the desired capability.
- If the parent framework has one or more other children composite bundles (sibling) and the sibling matches any specified peer composite constraint from the import policy and a sibling composite has an export sharing policy then the capability may come from that sibling composite framework. Additionally, sibling composites may have children composites that have export sharing policies. Delegation of a sibling's children can continue all the way to the leave children (grand children etc.) as long as each child composite framework in the hierarchy has an export sharing policy which matches the desired capability.

5.3.2 Resolve Time Sharing Policy

Since composite bundles do not actually define any constraints they should always be resolvable by the framework. Composites define the sharing policy for resolve time with the following composite bundle headers:

- Composite-PackageImportPolicy
- Composite-PackageExportPolicy
- Composite-BundleRequirePolicy
- Composite-BundleProvidePolicy

To illustrate the resolve time sharing policy the following symbols are used:
Example composite meta-data:

```
Bundle-SymbolicName: foo.composite; composite:=true
Bundle-Version: 1.0
Composite-PackageImportPolicy: b; version="[1.0,2.0)"
Composite-PackageExportPolicy: c; version="[1.0,2.0)"
Composite-BundleRequirePolicy: X; bundle-version="[1.0,2.0)"
Composite-BundleProvidePolicy: Z; bundle-version="[1.0,2.0)"
```

If no resolve time sharing policy is defined then constituent bundles only have the resolve time capabilities provided by other constituent bundles when resolving the constraints from constituent bundles and any resolve time capabilities provided by constituent bundles must not be available to resolve constraints from bundles installed in the parent framework.

When a resolve time sharing policy is defined then additional resolve time capabilities may be used which are provided by bundles installed in the parent for resolving constituent constraints (importing) and resolve time capabilities provided by constituent bundles may be used for resolving constraints from bundles installed in the parent framework (exporting). The figure below illustrates the resolve time sharing policy defined above.
The sharing policy allows parent framework bundles W and X to get wired to the exported package c from constituent bundle Y in the composite. The sharing policy also allows constituent bundle Y to get wired to the exported package b from bundle X in the parent framework. The sharing policy also allows constituent bundle Z to get wired to the parent framework bundle X (using Require-Bundle). The sharing policy also allows parent framework bundle V to get wired to the constituent bundle Z.

Any wires established as a result of the composite sharing policy are direct wires to the provider bundles. For example, the PackageAdmin service in the parent framework above will report that the exported package X.b has the constituent bundle Y as an importing bundle. Similarly, the PackageAdmin service in the composite framework above will report that the exported package Y.c has the parent framework bundles W and X as importing bundles.
Note that the bundle require policy is only used to resolve Require-Bundle constraints of constituent bundles. The bundles in the parent framework which match a bundle require policy of a composite do not automatically have their exported packages become available to resolve Import-Package constraints of constituent bundles. The package import policy must be used to make exported packages from bundles installed in the parent framework available to resolve constituent bundle's Import-Package constraints.

Similarly, the bundle provide policy is only used to resolve Require-Bundle constraints of bundles installed in the parent framework. The constituent bundles which match a bundle provide policy of a composite do not automatically have their exported packages become available to resolve Import-Package constraints of bundles installed in the parent framework. The package export policy must be used to make exported packages from constituent bundles available to resolve Import-Package constraints of bundles installed in the parent framework.

With the same declared meta-data for the foo.composite the following diagram illustrates how the resolve time sharing policy isolates the constituent bundles.

Example Composite Diagram
5.3.2.1 Wild Card Resolve Time Sharing Policies

Wild cards may be used to match potential capabilities in resolve time sharing policies. This allows for a composite to import all packages from a parent composite. For package import/export policy the syntax for wild cards is identical to DynamicImport-Package. For bundle require/provide policy the wild card syntax is identical to DynamicImport-Package syntax except the wild card is used to match against a bundle symbolic name of the provider instead of a package name.

Care should be used when using wild card sharing policies. Using a broad wild card (i.e \"*\") without any additional matching attributes greatly reduces the level of isolation and in some cases can completely remove any isolation inside and out of a composite.

The following is an example sharing policy for importing all packages from the parent that have the matching attribute \(x=y\):

\[
\text{Bundle-SymbolicName: composite.importall; composite:=true} \\
\text{Bundle-Version: 1.0} \\
\text{Composite-PackageImportPolicy: *; x=y}
\]

Similarly, the bundle require policy can use wild cards to require all bundles in the parent framework that match:

\[
\text{Bundle-SymbolicName: composite.requireall; composite:=true} \\
\text{Bundle-Version: 1.0} \\
\text{Composite-BundleRequirePolicy: a.b.*}
\]
**Wild card bundle require policy**

```
composites.requireall-1.0.0
```

Note that wild cards can also be specified on the package export policy with something like the following:

```
Bundle-SymbolicName: composite.exportall; composite:=true
Bundle-Version: 1.0
Composite-PackageExportPolicy: *; x=y
```

**Wild card package export policy**

```
composites.exportall-1.0.0
```

Similarly, wild cards can also be specified on the bundle provide policy with something like the following:

\[
\begin{align*}
\text{Bundle-SymbolicName: composite.provideall; composite:=true} \\
\text{Bundle-Version: 1.0} \\
\text{Composite-BundleProvidePolicy: a.b.*}
\end{align*}
\]

**Wild card bundle require policy**

```
composite.provideall-1.0.0
```

---

5.3.2.2 Resolving singleton bundles

The framework must resolve at most one bundle when multiple versions of a singleton bundle with the same symbolic name are installed in the same framework. Singleton bundles installed in different frameworks do not effect the resolution of each other. The global tree of bundles is allowed to have multiple installations of a singleton bundle resolved as long as the resolved singleton bundles are not installed in the same composite framework.

The following diagram illustrates a root framework with two composites (1 and 2) installed and two singleton bundles A and B installed. Each composite (1 and 2) also have two singleton bundles A and B installed.
In this example there are three installations of the singleton bundles A and B in the global tree of bundles. But since all three installations are in separate frameworks the three installations of A and B do not influence the resolution of each other. All three installations of A and B are allowed to resolve in this case.

The bundle require provide policies do not influence the resolution of singleton bundles. Any singleton bundle installed in a composite framework which matches one or more of the symbolic names used in a bundle require or provide policy must not be allowed to resolve. In the above example A.0 is the installation of A in the root framework, A.1 is installed in composite 1 and A.2 is installed in composite 2. If composite 1 has a bundle require or provide policy that matched against the bundle symbolic name of A then A.1 is not allowed to be resolved, only A.0 is allowed to resolve in that case.

The following diagram illustrates a bundle require policy that prevents a singleton constituent bundle from resolving inside a composite.
In this example a singleton bundle A-v1.0.0 is installed in the parent framework. A composite is also installed in the parent framework and the composite has a singleton bundle A-v1.0.1. The composite also has a bundle require sharing policy which matches A-v1.0.0 installed in the parent. This sharing policy prevents the singleton bundle A-1.0.1 from resolving. Note that A-1.0.1 will not be allowed to resolve even if there is no matching version of A found in the parent. This is necessary to prevent future collisions if a version of A is installed later into the parent while version A-1.0.1 is resolved in a composite framework. The following diagram illustrates this:
The bundle provide policy has similar behavior and will prevent singleton constituent bundles from resolving if the singleton constituent bundle has a symbolic name that matches the bundle provide policy. The following diagram illustrates this:

**Singleton Bundles in Composites**

```
composite-1.0.0

A-1.0.1
  a-1.1
  \[\times\]  
  A-[1.0,2.0)
```

Caution must be used when defining a bundle require or provide bundle sharing policy because it will prevent singleton bundles from resolving. For example, using a wildcard of "*" for the bundle require bundle or provide sharing policy will prevent all singleton bundles installed in a composite from resolving.

### 5.3.3 Service Sharing Policy

Composites define a service sharing policy that controls what services registered from bundles installed in the parent framework are available to the constituent bundles and what services registered by composite bundles are available to bundles installed in the parent framework. A composite bundle does not actually register or consume the services itself at runtime it only defines the sharing policy across composite boundaries for services. Composites define the service sharing policy with the following composite bundle headers:

- Composite-ServiceImportPolicy
- Composite-ServiceExportPolicy

To illustrate the service sharing policy the following symbols are used:
Example composite meta-data:

Bundle-SymbolicName: foo.composite; composite:=true
Bundle-Version: 1.0
Composite-ServiceExportPolicy: (export=true)
Composite-ServiceImportPolicy: (import=true)

If no service sharing policy is defined then constituent bundles only have access to services registered by other constituent bundles when getting services and listening for service events and any services registered by constituent bundles must not be available to bundles installed in the parent framework.

When a service sharing policy is defined then additional service which are registered by bundles installed in the parent framework may be accessed by constituent bundles when getting services and listening for service events. The figure below illustrates the service sharing policy defined above.
5.3.4 Transparency of sharing

Capabilities which are shared across framework boundaries are shared in what is referred to as transparent sharing. This is also referred to as direct wiring. For example, when a constituent bundle imports a package which is imported by the composite sharing policy from the parent framework then the constituent bundle will be wired directly to the exporting bundle installed in the parent framework. Similarly, when a constituent bundle gets a service which is imported by a composite sharing policy from the parent framework then the constituent bundle will become a direct user of the service registered by a bundle installed in the parent framework.

This is referred to as transparent sharing because providers of capabilities that satisfy the composite sharing policy can detect the usage of their capability by other bundles which are not installed in the same framework. Bundles can also detect the shared capabilities provided by other bundles which are not installed in the same framework.
5.3.5 Priority of shared capabilities

Capabilities imported into a composite framework do not have any additional priority associated with them. All capabilities visible to a constituent bundle should use the same priority rules as capabilities provided by bundles within a single framework. For example, package version, exporting bundle ID, service ranking, service ID etc.

5.3.6 Sibling Composite Constraints

A composite may require additional control over the provider of an imported resource. For example, the management agent may have a particular composite installed that contains a specific implementation of a set of packages and these packages are exported by that composite’s sharing policy. The management agent also installs another composite which wants to import these packages through the sharing policy. The management agent may require additional control to ensure that the importing composite will only import the packages from a particular sibling composite. The composite directives (composite-symbolic-name and composite-version) can be used to place additional control on the sibling provider of imported shared capabilities. Consider the following diagram:

*Example Composite Diagram*
This diagram illustrates a scenario where the management agent must ensure the constituent bundles in composite C import the package b from the constituent bundle X-1.0.0 in composite A. This can be specified with the following composite meta-data for the foo.composite composite:

- Bundle-SymbolicName: foo.compositeC; composite:=true
- Bundle-Version: 1.0
- Composite-PackageImportPolicy: b; version="[1.0,2.0)"
- composite-symbolic-name:=A;
- composite-version:="[1.0, 2.0)"

The composite directives identify composite bundles that must be a direct sibling composite of the importing composite. A sibling composite can only satisfy a composite affinity constraint if its parent framework is the same as parent framework of the importing composite and the sibling's sharing policy exports the requested capability.

The following diagram illustrates a composite constraint with a sibling composite which has another nested composite containing the requested capability:

**Example Composite Diagram**

Note that bundle X installed in composite B is allowed to satisfy the composite constraint for A because B is nested inside of composite A and composite A has a sharing policy which exports the package b. Siblings are allowed to have nested composites which ultimately provide the requested capability.
The bundle require and service import policies also are able to specify composite constraints with the composite directives. Consider the following diagram:

**Example Composite Diagram**

This can be specified with the following composite meta-data for the C composite:

```
Bundle-SymbolicName: C; composite=true
Bundle-Version: 1.0
Composite-BundleRequirePolicy: X; version="[1.0,2.0)";
    __composite-symbolic-name:=A;
    __composite-version:="[1.0, 2.0)"
Composite-ServiceImportPolicy: "(share=true)";
    __composite-symbolic-name:=A;
    __composite-version:="[1.0, 2.0)"
```

---

Copyright © IBM Corporation 2010 All Rights Reserved
5.4 Lifecycle of composite bundles

The lifecycle of a composite bundle has the same state transitions as a normal bundle. This includes firing the associated BundleEvents related to the state transition. This allows bundle listeners to track the lifecycle of composite bundles. The difference is the relationship the composite framework and the constituent bundles have with the lifecycle of a composite bundle. The following table describes the composite state transition and how it relates to the composite framework and constituent bundles:

<table>
<thead>
<tr>
<th>Composite State Transition</th>
<th>Composite Framework</th>
<th>Constituents</th>
</tr>
</thead>
<tbody>
<tr>
<td>installation -&gt; INSTALLED</td>
<td>Composite framework must have a valid system bundle context. Composite system bundle state is STARTING</td>
<td>No constituents are installed yet. While the composite is in the INSTALLED state, no constituent bundles may be resolved.</td>
</tr>
<tr>
<td>INSTALLED-&gt;RESOLVED</td>
<td>Composite framework must remain at start-level 0. Constituent bundles are allowed to enter the resolved state once the composite is resolved. No constituent bundles are allowed to activate. Their start-level cannot be met.</td>
<td></td>
</tr>
<tr>
<td>RESOLVED-&gt;STARTING</td>
<td>Composite framework sets its start-level to the beginning start-level, the composite framework system bundle's state is set to Bundle.ACTIVE and the FrameworkEvent.STARTED event is fired. Constituent bundles are started according to the start-level specification.</td>
<td></td>
</tr>
<tr>
<td>STARTING-&gt;ACTIVE</td>
<td>No effect</td>
<td>No effect</td>
</tr>
<tr>
<td>ACTIVE-&gt;STOPPING</td>
<td>Composite framework sets its start-level to 0. All constituent bundles must be stopped. Their start-level cannot be met.</td>
<td></td>
</tr>
<tr>
<td>STOPPING-&gt;RESOLVED</td>
<td>Composite system bundle state is set to STARTING. The composite system bundle context must remain valid. No effect</td>
<td></td>
</tr>
<tr>
<td>RESOLVED-&gt;INSTALLED (refresh)</td>
<td>No effect</td>
<td>Refreshing the composite bundle causes a refresh on all constituent bundles</td>
</tr>
<tr>
<td>RESOLVED-&gt;INSTALLED-&gt;UNINSTALLED (uninstall)</td>
<td>Uninstalling a composite bundle invalidates the system bundle context when the composite bundle is enters the UNINSTALLED state. All the constituent bundles are uninstalled after a composite bundle's state enters the INSTALLED and before it enters the UNINSTALLED state.</td>
<td></td>
</tr>
</tbody>
</table>

When the root framework is shutdown all resources associated with child composite frameworks must be released. Among other things this includes invalidating the composite system bundle context.

5.5 Impacts on the system bundle

The composite bundle is the only way to control the lifecycle of the composite framework. All lifecycle methods on the composite framework's system bundle should throw unsupported exceptions. This includes the start, stop, uninstall, and update methods.
5.5.1 Core Services

Core services are services that are registered by the framework implementation or white board patterns used by the framework for extensibility. This includes things like PackageAdmin, StartLevel, ConditionalPermissionAdmin and the URL handler support. Each composite and parent framework will have a system bundle which represents the framework. The core services registered by the system bundle must provide the proper isolation for the framework represented by the system bundle.

Note that the system bundle context provided by a the CompositeBundle.getSystemBundleContext() can be used by management agents to get core services for a particular composite framework. For example, to obtain the StartLevel service for configuring the start-level of constituent bundles.

5.5.1.1 Start Level Service

The start level service must only act upon the bundles which are installed in the framework represented by the system bundle which registered the start level service. The following methods which act upon bundles must throw an IllegalArgumentException if the Bundle parameter is not installed in the same composite framework:

- getBundleStartLevel(Bundle)
- isBundleActivationPolicyUsed(Bundle)
- isBundlePersistentlyStarted(Bundle)
- setBundleStartLevel(Bundle, int)

Other StartLevel methods act upon the particular composite framework represented by the system bundle.

- getInitialBundleStartLevel()
- getStartLevel()
- setInitialBundleStartLevel(int)
- setStartLevel(int)

5.5.1.2 Permission Admin and Conditional Permission Admin

The permission admin and conditional permission admin services must only manage the permissions for the constituent bundles which are installed into the composite framework represented by the system bundle. When a composite bundle is installed the permission and conditional permission admin services associated with the composite framework must use the default default permissions (i.e. AllPermissions) regardless of how the permissions are configured in the parent framework. This is one reason AllPermissions is required to install a composite bundle. This means all constituents installed in a composite framework will be granted AllPermissions by default until the appropriate admin service is used to configure the permissions for the composite's constituent bundles.

Using the permission or conditional permission admin service from a composite framework must not affect the permissions assigned to other bundles in the global tree of bundles which are not installed in the composite framework.
5.5.1.3 URL Handlers

The URL stream handler and content handler support in core uses the white board pattern to discover URLStreamHandlerService services that support various protocols and ContentHandler services that support various MIME types. Protocols supported by URLStreamHandlerService services and MIME types supported by ContentHandler services must only be available to bundles installed in the same framework in which the handler service is registered.

URLStreamHandlerService and ContentHandler services can be imported and exported by a composite service sharing policy to make them available outside of a particular framework.

5.5.1.4 Service Hooks

Service Hooks registered should only have access to events that pertain to the framework they are registered with or match the composite sharing policy. In order for a service hook to be called it must be registered with the same framework or imported from the parent framework or exported by a child composite of the framework where the bundle is installed which is performing the service action (e.g. service registration/modification/unregistration, service lookup, service listening).

The following outlines the rules for each service hook:

EventHook

The event hook is passed a collection of BundleContext objects for bundles which have listeners to which a specific event will be delivered. This collection typically only contains BundleContext objects for bundles installed in the same framework as the EventHook is registered. The composite sharing policy may add additional BundleContext objects from composite or parent frameworks which are allowed to see the BundleEvent. In this case the collection may contain BundleContext for bundles outside the framework for which the EventHook is registered.

A service event bundle is a Bundle object associated with a service event (i.e. the bundle returned by the method BundleEvent.getServiceReference().getBundle()). The EventHook.event(ServiceEvent event, Collection<BundleContext> contexts) method is called on an EventHook registration if one of the following rules apply:

- If the EventHook service is registered in the same framework where the service event bundle is installed.
- If the EventHook service matches the service import policy of the composite where the service event bundle is installed.
- If the EventHook service matches a service export policy of a child composite of the framework where the service event bundle is installed.

FindHook

The find hook is passed a collection of ServiceReference objects to be returned as a result of a find operation. This collection typically only contains ServiceReference objects for services registered in the same framework as the FindHook is registered. The composite sharing policy may add additional ServiceReference objects registered by bundles installed in the composite or parent frameworks. In this case the collection may contain ServiceReference objects registered by bundles outside the framework for which the FindHook is registered.

A find service bundle is a Bundle object associated with a BundleContext performing a service find operation (i.e. BundleContext.getServiceReference methods). The FindHook.find(BundleContext context, ...) method is called on a FindHook registration if one of the following rules apply:

- If the FindHook service is registered in the same framework where the find service bundle is installed.
• If the FindHook service matches the service import policy of the composite where the find service bundle is installed.

• If the FindHook service matches a service export policy of a child composite of the framework where the find service bundle is installed.

**ListenerHook**

The listener hook is passed a collection of ListenerInfo objects for newly added or removed service listeners. This collection typically only contains ListenerInfo objects for service listeners that are registered in the same framework as the ListenerInfo is registered. The composite sharing policy may add additional ListenerInfo objects for service listeners from composite or parent frameworks.

A service listener bundle is a Bundle object associated with a service listener (i.e. the bundle returned by the method BundleContext.getBundle() for the BundleContext for which the service listener was added to). The ListenerHook.added(...) and ListenerHook.removed(...) methods are called on a ListenerHook registration if one of the following rules apply:

• If the ListenerHook service is registered in the same framework where the service listener bundle is installed.

• If the ListenerHook service matches the service import policy of the composite where the service listener bundle is installed.

• If the ListenerHook service matches a service export policy of a child composite of the framework where the service listener bundle is installed.

### 5.5.2 Framework Extension Bundles

System bundle fragments (extension bundles) are not allowed to be installed into a composite framework. Extension bundle are only allowed to be installed into the root parent framework. This is mainly needed to ensure that the design is implementable without forcing a real framework instance to be created with a new class loader to support installation of framework extensions.

### 5.6 Configuration Properties

The proposed solution hides the composite framework instance as an implementation detail. The CompositeAdmin service supports a very limited set of configuration properties which can be used to configure the behavior of the composite framework. Currently only the following configuration property is supported for composite framework configuration. All other framework configuration properties should be ignored.

• org.osgi.framework.startlevel.beginning – to specify the beginning composite framework start-level when a composite is started

#### 5.6.1 BundleContext Properties

The BundleContext interface defines a method for returning information pertaining to Framework properties: getProperty(String). For a composite’s constituent bundles, the BundleContext.getProperty(String) method must use the following search order for a property:

• Framework constants (vendor, implementation, etc)

• Composite configuration properties

• System properties

Copyright © IBM Corporation 2010 All Rights Reserved
5.7 Impacts on White Board and Extenders

The isolation composites provide to constituent bundles has an impact to designs which use the white board pattern or the extender pattern.

5.7.1 White Board

Since a composite framework has an isolated service space any services registered by constituent bundles will not be visible to bundles installed in the parent framework. If a bundle installed in the parent framework uses the white board pattern then they will not be able to use the services registered by the constituent bundles (unless the service sharing policy allows for it).

For example, an EventAdmin implementation installed in the parent will not detect an EventHanders registered by constituent bundles installed in the composite framework. Similarly a ConfigurationAdmin implementation will not detect any ManagedServices registered by constituent bundles installed in the composite framework. There are two options to deal with this.

1. Install the necessary implementation bundles into the composite framework.

2. Use a service sharing policy that exports the necessary services out of the composite. This option may not be appropriate for all white board patterns. For example, Event Admin may be able to support handlers registered by bundles that are not actually installed into their own framework but it may not be possible for ConfigurationAdmin to do the same thing.

XXX – For ConfigurationAdmin the transparent sharing of services makes this much easier to deal with. The configuration admin service can now act properly when a constituent bundle outside the framework of the configuration admin service obtains the service. The specification now mandates that all bundles in the root tree of bundles must have a unique location. This should allow configuration admin to behave correctly no matter what the source is for the ManagedService.

5.7.2 Extenders

Since a composite framework has an isolated bundle space any constituent bundle (and bundle events) will not be visible to bundles installed in the parent framework. If a bundle installed in the parent framework implements an extender then they will not be able to process constituent bundles.

For example, a declarative services implementation installed in the parent will not detect any bundles installed in the composite framework and will not discover any component XML declared in them. There are two options to deal with this:

1. Install the necessary extender implementation bundles into the composite framework.

2. Enhance the extender implementation to use the CompositeBundle API to reflect into a composite framework and act upon the constituent bundles. This can be done by using the composite bundle's system bundle context.

5.8 Open Issues

Other Singleton resource providers?

XXX – These don't really seem like issues anymore with transparent sharing and the fact that an implementation can use virtual frameworks to implement this solution.
Security Considerations

There are several open issues regarding security considerations for using multiple frameworks in the same JVM:

- The security manager delegates to the protection domain of the individual bundles when performing permission checks. Because every bundle has its own protection domain, the presence of multiple frameworks should not impact permission checks in general. **XXX – non-issue.** This is obvious. All properly behaving Java 2 security managers must delegate to the protection domain defined with the Class at class load time.

- The presence/use of nested frameworks may impact the behavior of postponed conditions in the ConditionalPermissionAdmin service. More investigation and clarification is needed in this area. **XXX – This is a non-issue if virtual framework isolation is used.** There will be no need to multiplex the security manager. This does remain an issue if multiple disparate frameworks, but this is not an issue for this RFC.

- A check for AllPermission will be performed when creating a new nested framework if a SecurityManager is enabled. **Per RFC 132, AllPermission is needed to create a framework-- this requirement extends to the creation of nested or child frameworks as well.** A framework implementation must ensure that the composite bundles it creates have sufficient permissions to do the necessary import/export/service tracking operations. **XXX – This is a non-issue.** The API clearly documents that ALL permission is needed to install a composite. The composite no longer needs additional permissions to import/export packages or track services because this RFC uses transparent sharing which removes the composite as an actor upon the shared resources.
7 Javadoc

7.1 org.osgi.service.composite

Interface CompositeAdmin

public interface CompositeAdmin

The composite admin service is registered by a framework which supports composite bundles. The composite admin service is used to install composite bundles.

Version: $Revision: 8585 $

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompositeBundle getParentCompositeBundle()</td>
<td>Returns the parent composite bundle associated with the framework with which this composite admin service is registered.</td>
</tr>
<tr>
<td>CompositeBundle installCompositeBundle(java.lang.String location, java.util.Map&lt;java.lang.String,java.lang.String&gt; compositeManifest, java.util.Map&lt;java.lang.String,java.lang.String&gt; configuration)</td>
<td>Installs a composite bundle into the parent framework with which this composite admin service is registered.</td>
</tr>
</tbody>
</table>

Method Detail

7.1.1 installCompositeBundle


Installs a composite bundle into the parent framework with which this composite admin service is registered.
The following steps are required to install a composite:
1. The composite manifest is verified. If this fails, a BundleException is thrown.
2. If a bundle containing the same location identifier is already installed, if that bundle is a
   composite bundle the CompositeBundle object for that bundle is returned otherwise a
   BundleException is thrown.
3. The composite's associated resources are allocated. The associated resources minimally
   consist of a unique identifier and a composite framework. If this step fails, a BundleException
   is thrown.
4. The composite framework is initialized and its start-level is set to 0. At this point no
   constituent bundles are installed in the composite framework.
5. A bundle event of type BundleEvent.INSTALLED is fired.
6. The CompositeBundle object for the newly installed composite is returned

Postconditions, no exceptions thrown
- getState() in {INSTALLED, RESOLVED}.
- Composite has a unique ID.
- The composite framework is initialized, in the Bundle.STARTING state and its start-level is
  set to 0.

Postconditions, when an exception is thrown
- Composite is not installed and no trace of the composite exists.

Parameters:
- location - The location identifier of the composite to install.
- compositeManifest - The meta-data describing the composite. This includes the symbolic
  name and version and the sharing policy.
- configuration - The configuration parameters to the composite framework. See
  CompositeConstants for the supported configuration parameters.

Returns:
- the installed composite bundle.

Throws:
- org.osgi.framework.BundleException - if the installation failed.
- java.lang.SecurityException - If the caller does not have AllPermission.
- java.lang.IllegalStateException - If this composite admin service is no longer valid.
  For example, if the framework has shutdown.

7.1.2  getParentCompositeBundle

CompositeBundle  getParentCompositeBundle()

Returns the parent composite bundle associated with the framework with which this composite admin
service is registered. The composite admin service registered in the root framework must return
null.

Returns:
- The parent composite bundle or null.

7.2  org.osgi.service.composite

Interface CompositeBundle

All Superinterfaces:
public interface CompositeBundle extends org.osgi.framework.Bundle

A composite bundle is a bundle for which the content is composed of meta-data describing the composite and a set of bundles called constituent bundles. Constituent bundles are isolated from other bundles which are not constituents of the same composite bundle. Composites provide isolation for constituent bundles in the following ways.

- Class space: Constraints specified by constituent bundles (e.g. Import-Package, Require-Bundle, Fragment-Host) can only be resolved against capabilities provided by other constituents within the same composite bundle (Export-Package, Bundle-SymbolicName/Bundle-Version).
- Service Registry: Constituent bundles only have access to services registered by other constituents within the same composite bundle. This includes service events and service references.
- Bundles: Constituent bundles only have access to other Bundle objects that represent constituents within the same composite bundle. This includes bundle events and core API like BundleContext.getBundles, PackageAdmin and StartLevel.

A parent framework refers to the OSGi framework where a composite bundle is installed. A composite framework refers the framework where the constituent bundles are installed and provides the isolation for the constituent bundles. Constituent bundles are isolated but there are many cases where capabilities (packages and services) need to be shared between bundles installed in the parent framework and the constituents within a composite bundle.

Through a sharing policy the composite is in control of what capabilities are shared across framework boundaries. The sharing policy controls what capabilities provided by bundles installed in the parent framework are imported into the composite and made available to constituent bundles. The sharing policy also controls what capabilities provided by constituent bundles are exported out of the composite and made available to bundles installed in the parent framework.

The lifecycle of the constituent bundles are tied to the lifecycle of the composite bundle. See start(int), stop(int), update(Map) and uninstall().

Version: $Revision: 8585 $
Methods inherited from interface org.osgi.framework.Bundle

adapt, getBundleContext, getBundleId, getHeaders, getHeaders, getLastModified, getLocation, getRegisteredServices, getServicesInUse, getSignerCertificates, getState, getSymbolicName, getTypes, getVersion, hasPermission

Methods inherited from interface java.lang.Comparable

compareTo

Method Detail

7.2.1  getSystemBundleContext

org.osgi.framework.BundleContext  getSystemBundleContext()

Returns the system bundle context for the composite framework. This method must return a valid BundleContext as long as this composite bundle is installed. Once a composite bundle is uninstalled this method must return null. The composite system bundle context can be used to install and manage the constituent bundles.

Returns:
the system bundle context for the composite framework

Throws:
java.lang.SecurityException - If the caller does not have the appropriate AdminPermission[system.bundle,CONTEXT], and the Java Runtime Environment supports permissions.

### 7.2.2 start

```java
void start() throws org.osgi.framework.BundleException
```

Starts this composite with no options.

This method performs the same function as calling `start(0)`.

**Specified by:**

`start` in interface `org.osgi.framework.Bundle`

**Throws:**

- `org.osgi.framework.BundleException` - If this composite could not be started.
- `java.lang.IllegalStateException` - If this composite has been uninstalled.
- `java.lang.SecurityException` - If the caller does not have the appropriate AdminPermission[this,EXECUTE], and the Java Runtime Environment supports permissions.

**See Also:**

`start(int)`

### 7.2.3 start

```java
void start(int options) throws org.osgi.framework.BundleException
```

Starts this composite according to `Bundle.start(int)`. When the composite bundle's state is set to `Bundle.STARTING` and after the `BundleEvent.STARTING` event has been fired for this composite bundle, the following steps are required to activate the constituents:

1. The composite framework start-level is set to the beginning start-level.
2. The constituent bundles are started according to the start-level specification.
3. The `FrameworkEvent.STARTED` event is fired for the composite framework.
4. The composite bundle state is set to `Bundle.ACTIVE`
5. The bundle event of type `BundleEvent.STARTED` is fired for this composite.

**Specified by:**

`start` in interface `org.osgi.framework.Bundle`

**Parameters:**

- `options` - The options for starting this composite. See `Bundle.START_TRANSIENT` and `Bundle.START_ACTIVATION_POLICY`. The Framework must ignore unrecognized options.

**Throws:**

- `org.osgi.framework.BundleException` - If this composite could not be started.
- `java.lang.IllegalStateException` - If this composite has been uninstalled.
- `java.lang.SecurityException` - If the caller does not have the appropriate AdminPermission[this,EXECUTE], and the Java Runtime Environment supports permissions.
7.2.4 stop

```java
void stop()
    throws org.osgi.framework.BundleException

Stops this composite with no options.

This method performs the same function as calling stop(0).
```

**Specified by:**
- stop in interface org.osgi.framework.Bundle

**Throws:**
- org.osgi.framework.BundleException - If an error occurred while stopping this composite
- java.lang.IllegalStateException - If this composite has been uninstalled.
- java.lang.SecurityException - If the caller does not have the appropriate AdminPermission[this,EXECUTE], and the Java Runtime Environment supports permissions.

**See Also:**
- start(int)

7.2.5 stop

```java
void stop(int options)
    throws org.osgi.framework.BundleException

Stops this composite according to Bundle.stop(int). When the composite bundle's state is set to Bundle.STOPPING, the following steps are required to stop the constituents:
1. The composite framework start-level is set to the 0.
2. The constituent bundles are stopped according to the start-level specification.
3. The bundle event of type BundleEvent.STARTED is fired for the composite.
```

**Specified by:**
- stop in interface org.osgi.framework.Bundle

**Parameters:**
- options - The options for stopping this bundle. See Bundle.STOP_TRANSIENT. The Framework must ignore unrecognized options.

**Throws:**
- org.osgi.framework.BundleException - If an error occurred while stopping this composite
- java.lang.IllegalStateException - If this composite has been uninstalled.
- java.lang.SecurityException - If the caller does not have the appropriate AdminPermission[this,EXECUTE], and the Java Runtime Environment supports permissions.

**Since:**
- 1.4

7.2.6 uninstall

```java
void uninstall()
    throws org.osgi.framework.BundleException
```
Uninstalls this composite according to `Bundle.uninstall()`. When the composite bundle is uninstalled the following steps are required:

1. After a composite's state is set to `Bundle.INSTALLED` and before a composite's state is set to `Bundle.UNINSTALLED`, all the constituent bundles are uninstalled.
2. When a composite's state is set to `Bundle.UNINSTALLED`, the composite system bundle context becomes invalid.

**Specified by:**
- `uninstall` in interface `org.osgi.framework.Bundle`

**Throws:**
- `org.osgi.framework.BundleException`

---

### 7.2.7 update

```java
void update()
throws org.osgi.framework.BundleException
```

This operation is not supported for composite bundles. A `BundleException` of type `invalid operation` must be thrown.

**Specified by:**
- `update` in interface `org.osgi.framework.Bundle`

**Throws:**
- `org.osgi.framework.BundleException`

---

### 7.2.8 update

```java
void update(java.io.InputStream input)
throws org.osgi.framework.BundleException
```

This operation is not supported for composite bundles. A `BundleException` of type `invalid operation` must be thrown.

**Specified by:**
- `update` in interface `org.osgi.framework.Bundle`

**Throws:**
- `org.osgi.framework.BundleException`

---

### 7.2.9 update

```java
void update(java.util.Map<String, String> compositeManifest)
throws org.osgi.framework.BundleException
```

Updates the meta-data for this composite from a `Map`.

The specified `Map` must not be `null`. 
If this composite’s state is ACTIVE, it must be stopped before the update and started after the update successfully completes.

If this composite has a sharing policy for packages, any packages that are imported by other bundles through the current sharing policy must remain exported for existing importers. A call PackageAdmin.refreshPackages method will force all constituent bundles for this composite to be refreshed, causing the new package sharing policy to be used for all constituent bundles.

The following steps are required to update a composite:

1. If this composite’s state is UNINSTALLED then an IllegalStateException is thrown.
2. The component manifest Map is verified. If this fails, a BundleException is thrown.
3. If this composite’s state is ACTIVE, STARTING or STOPPING, this bundle is stopped as described in the CompositeBundle.stop method. If CompositeBundle.stop throws an exception, the exception is rethrown terminating the update.
4. The meta-data of this composite is updated from the Map.
5. This composite’s state is set to INSTALLED.
6. If the updated version of this composite was successfully installed, a bundle event of type BundleEvent.UPDATED is fired.
7. If this composite’s state was originally ACTIVE, the updated composite is started as described in the CompositeBundle.start method. If CompositeBundle.start throws an exception, a Framework event of type FrameworkEvent.ERROR is fired containing the exception.

Preconditions
- getState() not in { UNINSTALLED }.

Postconditions, no exceptions thrown
- getState() in { INSTALLED, RESOLVED, ACTIVE }.
- This composite has been updated.

Postconditions, when an exception is thrown
- getState() in { INSTALLED, RESOLVED, ACTIVE }.
- Original composite is still used; no update occurred.

Parameters:
- compositeManifest - The Map from which to read the new composite meta-data from. Must not be null.

Throws:
- org.osgi.framework.BundleException - If the the update fails.
- java.lang.IllegalStateException - If this composite has been uninstalled.
- java.lang.SecurityException - If the caller does not have AllPermission.

See Also:
- stop(), start()

7.2.10 loadClass

java.lang.Class<?>, loadClass(java.lang.String name) throws java.lang.ClassNotFoundException

Composite bundles do not have class content. This method must throw a ClassNotFoundException.

Specified by:
- loadClass in interface org.osgi.framework.Bundle

Throws:
- java.lang.ClassNotFoundException - Always thrown for composite bundles.
7.2.11 getResource

```java
java.net.URL getResource(java.lang.String name)
```

Composite bundles do not have content. This method must return null.

**Specified by:**
getResources in interface org.osgi.framework.Bundle

**Returns:**
A null value is always returned for composites.

7.2.12 getResources

```java
java.utilEnumeration<java.net.URL> getResources(java.lang.String name)
```

throws java.io.IOException

Composite bundles do not have content. This method must return null.

**Specified by:**
getResources in interface org.osgi.framework.Bundle

**Returns:**
A null value is always returned for composites.

**Throws:**
java.io.IOException

7.2.13 getEntry

```java
java.net.URL getEntry(java.lang.String path)
```

Composite bundles do not have content. This method must return null.

**Specified by:**
getEntry in interface org.osgi.framework.Bundle

**Returns:**
A null value is always returned for composites.

7.2.14 getEntryPaths

```java
java.util Enumeration<java.lang.String> getEntryPaths(java.lang.String path)
```

Composite bundles do not have content. This method must return null.

**Specified by:**
getEntryPaths in interface org.osgi.framework.Bundle

**Returns:**
A null value is always returned for composites.
7.2.15 findEntries

```java
java.util Enumeration<java.net.URL> findEntries(java.lang.String path,
java.lang.String filePattern,
boolean recurse)
```

Composite bundles do not have content. This method must return null.

Specified by:
findEntries in interface org.osgi.framework.Bundle

Returns:
A null value is always returned for composites.

---

7.3 org.osgi.service.composite

Class CompositeConstants

```java
java.lang.Object
org.osgi.service.composite.CompositeConstants
```

public class CompositeConstants extends java.lang.Object

Defines standard names for composite constants.

Version:
$Revision: 9104 $

---

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static java.lang.String COMPOSITE_BUNDLE_PROVIDE_POLICY</td>
</tr>
<tr>
<td>Composite manifest header (named &quot;Composite-BundleProvidePolicy&quot;) identifying a list of provide bundle constraints to provide out of a composite.</td>
</tr>
<tr>
<td>static java.lang.String COMPOSITE_BUNDLE_REQUIRE_POLICY</td>
</tr>
<tr>
<td>Composite manifest header (named &quot;Composite-BundleRequirePolicy&quot;) identifying a list of require bundle constraints to import into the composite.</td>
</tr>
<tr>
<td>static java.lang.String COMPOSITE_DIRECTIVE</td>
</tr>
<tr>
<td>Manifest header directive identifying whether a bundle is a composite.</td>
</tr>
<tr>
<td>static java.lang.String COMPOSITE_FRAMEWORK_BEGINNING_STARTLEVEL</td>
</tr>
<tr>
<td>A supported configuration parameter for a composite framework.</td>
</tr>
<tr>
<td>static java.lang.String COMPOSITE_PACKAGE_EXPORT_POLICY</td>
</tr>
<tr>
<td>Composite manifest header (named &quot;Composite-PackageExportPolicy&quot;) identifying a list of package constraints to export out of a composite.</td>
</tr>
<tr>
<td>static java.lang.String COMPOSITE_PACKAGE_IMPORT_POLICY</td>
</tr>
<tr>
<td>Composite manifest header (named &quot;Composite-PackageImportPolicy&quot;) identifying a list of package constraints to import into the composite.</td>
</tr>
<tr>
<td>static java.lang.String COMPOSITE_SERVICE_EXPORT_POLICY</td>
</tr>
<tr>
<td>Composite manifest header (named &quot;Composite-ServiceExportPolicy&quot;)</td>
</tr>
</tbody>
</table>
identifying a list of service filters that controls the services to export out of the composite.

static java.lang.String COMPOSITE_SERVICE_IMPORT_POLICY
   Composite manifest header (named "Composite-ServiceImportPolicy") identifying a list of service filters that controls the services to import into the composite.

static java.lang.String COMPOSITE_SYMBOLICNAME_DIRECTIVE
   Manifest header directive identifying the symbolic name a sibling composite must have that allows capabilities exported from that composite to match an import sharing policy specified by Composite-PackageImportPolicy or Composite-ServiceImportPolicy.

static java.lang.String COMPOSITE_VERSION_DIRECTIVE
   Manifest header directive identifying the version a sibling composite must have that allows capabilities exported from that composite to match an import sharing policy specified by Composite-PackageImportPolicy or Composite-ServiceImportPolicy.

Method Summary

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

7.3.1 COMPOSITE_DIRECTIVE

public static final java.lang.String COMPOSITE_DIRECTIVE

Manifest header directive identifying whether a bundle is a composite. The default value is false.

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

Bundle-SymbolicName: com.acme.composite.test; composite=true

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

A valid manifest for a composite bundle must have this directive set to true. Any attempt to install a composite which does not have this directive set to true must result in a BundleException.

See Also:
   Constants.BUNDLE_SYMBOLICNAME, Constant Field Values

7.3.2 COMPOSITE_PACKAGE_IMPORT_POLICY

public static final java.lang.String COMPOSITE_PACKAGE_IMPORT_POLICY
Composite manifest header (named "Composite-PackageImportPolicy") identifying a list of package constraints to import into the composite. Any exported package from a bundle installed in the parent framework which satisfies one of the specified package constraints is available to satisfy Import-Package constraints from constituent bundles.

This header uses the same syntax as the Import-Package header.

See Also:
Constant Field Values

7.3.3 COMPOSITE_PACKAGE_EXPORT_POLICY

```java
public static final java.lang.String COMPOSITE_PACKAGE_EXPORT_POLICY
```

Composite manifest header (named "Composite-PackageExportPolicy") identifying a list of package constraints to export out of a composite. Any exported package from a constituent bundle in the composite which satisfies one of the specified package constraints is available to satisfy Import-Package constraints from bundles installed in the parent framework.

This header uses the same syntax as the Import-Package header.

See Also:
Constant Field Values

7.3.4 COMPOSITE_BUNDLE_REQUIRE_POLICY

```java
public static final java.lang.String COMPOSITE_BUNDLE_REQUIRE_POLICY
```

Composite manifest header (named "Composite-BundleRequirePolicy") identifying a list of require bundle constraints to import into the composite. Any bundle installed in the parent framework which satisfies one of the specified require bundle constraints is available to satisfy Require-Bundle constraints from constituent bundles.

This header uses the same syntax as the Require-Bundle header.

See Also:
Constant Field Values

7.3.5 COMPOSITE_BUNDLE_PROVIDE_POLICY

```java
public static final java.lang.String COMPOSITE_BUNDLE_PROVIDE_POLICY
```

Composite manifest header (named "Composite-BundleProvidePolicy") identifying a list of provide bundle constraints to provide out of a composite. Any constituent bundle installed in the composite which satisfies one of the specified provide bundle constraints is available to satisfy Require-Bundle constraints from bundles installed in the parent framework.

This header uses the same syntax as the Require-Bundle header.
7.3.6 COMPOSITE_SERVICE_IMPORT_POLICY

```java
public static final java.lang.String COMPOSITE_SERVICE_IMPORT_POLICY
```

Composite manifest header (named "Composite-ServiceImportPolicy") identifying a list of service filters that controls the services to import into the composite. See Filter for a description of the filter syntax. Any services registered by bundles installed in the parent framework that match the specified service filter is available to constituent bundles.

See Also: Constant Field Values

7.3.7 COMPOSITE_SERVICE_EXPORT_POLICY

```java
public static final java.lang.String COMPOSITE_SERVICE_EXPORT_POLICY
```

Composite manifest header (named "Composite-ServiceExportPolicy") identifying a list of service filters that controls the services to export out of the composite. See Filter for a description of the filter syntax. Any services registered by constituent bundles that match the specified service filter is available to bundles installed in the parent framework.

See Also: Constant Field Values

7.3.8 COMPOSITE_SYMBOLICNAME_DIRECTIVE

```java
public static final java.lang.String COMPOSITE_SYMBOLICNAME_DIRECTIVE
```

Manifest header directive identifying the symbolic name a sibling composite must have that allows capabilities exported from that composite to match an import sharing policy specified by Composite-PackageImportPolicy or Composite-ServiceImportPolicy.

The directive value is encoded in the a manifest header like:

```
Composite-PackageImportPolicy: org.example.pkg; composite-symbolic-name="org.example.composite"
Composite-ServiceImportPolicy: "(attr=somevalue)"; composite-symbolic-name="org.example.composite"
```

See Also: COMPOSITE_PACKAGE_IMPORT_POLICY, COMPOSITE_SERVICE_IMPORT_POLICY, Constant Field Values
7.3.9  COMPOSITE_VERSION_DIRECTIVE

public static final java.lang.String COMPOSITE_VERSION_DIRECTIVE

Manifest header directive identifying the version a sibling composite must have that allows capabilities exported from that composite to match an import sharing policy specified by Composite-PackageImportPolicy or Composite-ServiceImportPolicy.

The directive value is encoded in the a manifest header like:

    Composite-PackageImportPolicy: org.example.pkg; composite-version="[1.0,1.1)"
    Composite-ServiceImportPolicy: "(attr=somevalue)"; composite-version="[1.0,1.1)"

In most cases a composite-symbolic-name directive should be specified along with the composite-version directive like:

    Composite-PackageImportPolicy: org.example.pkg;
        composite-symbolic-name="org.example.composite"; composite-version="[1.0,1.1)"
    Composite-ServiceImportPolicy: "(attr=somevalue)";
        composite-symbolic-name="org.example.composite"; composite-version="[1.0,1.1)"

See Also:
    COMPOSITE_PACKAGE_IMPORT_POLICY, COMPOSITE_SERVICE_IMPORT_POLICY, Constant Field Values

7.3.10 COMPOSITE_FRAMEWORK_BEGINNING_STARTLEVEL

public static final java.lang.String COMPOSITE_FRAMEWORK_BEGINNING_STARTLEVEL

A supported configuration parameter for a composite framework.

See Also:
    Constants.FRAMEWORK_BEGINNING_STARTLEVEL, Constant Field Values
8 Considered Alternatives

This RFC has considered several alternatives related to child frameworks:

A FrameworkFactory service with two methods: newChildFramework, for creating a child framework, and newFrameworkLink, for defining a one-way FrameworkLink used to share resources between frameworks. The FrameworkLink was one directional, and used/managed two bundles to represent itself (and its shared resources) in the two frameworks. This idea included an immutable object called a LinkDescription, which contained the declarations of packages and services that would be shared by the link.

A CompositeBundleFactory that would create new CompositeBundles with real nested Framework objects. This proposal also shared resources through a surrogate bundle in the composite framework. This avoided doing transparent sharing of capabilities across framework boundaries but it also had a number of lifecycle issues and added complexities.

8.1 Require Bundle Sharing Policy and Import-Package

An alternative to the Composite-BundleRequirePolicy would make any exported packages available from a matching bundle in parent available for import (via Import-Package) from a constituent bundle. There are two complications with this behavior that make this difficult to define and implement.

1. Bundles that re-export other required bundles: The export signature of a required bundle X includes all the packages it exports as well as any packages exported by required bundles Y. This implies that if a require bundle sharing policy matches bundle X then all the exports from bundle Y must also be “imported” into a composite for import by constituents. Additional complication is introduced if the parent framework is also a composite bundle which has a require bundle sharing policy to import bundle Y from its parent framework. Allowing all bundle Y’s exports to become available makes it difficult to resolve Import-Package constraints from constituent bundles because the require bundle sharing policy must be taken into account and its transitive nature regarding re-exporting of bundles.

2. Substitutable exports: Bundles that export and import the same package (name) are said to have substitutable exports. During resolution the framework decides if a bundle is an exporter or an importing of the package. No matter what choice the framework makes (importer or exporter) the bundle itself must maintain a constant set of packages for its export signature for requiring bundles. If bundle X exports and imports package foo and a framework decides bundle X will import package foo from bundle Y then any bundle requiring bundle X must get access to the package Y.foo. The question is should having a require bundle sharing policy which imports bundle X also make the package Y.foo available for Import-Package?

Given the level of complication to both the specification and the implementation it has been decided to restrict the require bundle policy to only apply to sharing of resources for resolving Require-Bundle constraints of constituent bundles and NOT Import-Package constraints of constituents.

8.2 Priority of shared capabilities

Capabilities imported into a composite framework should take priority over capabilities provided by other constituent bundles. Capabilities exported out of a composite should take a lower priority when compared to capabilities provided by other bundles installed in the parent framework.
Using sharing policy to add priority to capability selection was dropped. A framework will need the freedom to produce a valid result in as efficient way as possible. Adding this level of priority may explode the uses algorithm even more. This also complicates things like ServiceReference.compareTo for sorting. For now all capabilities visible to a constituent use the same priority rules as they would if they were installed into a single framework.

8.3 Composite framework configuration

Originally we wanted to allow system packages properties to be defined for composite framework to allow additional control over what the system bundle exports in a composite framework.

- system packages properties – to specify what packages are exported by the system bundle from the current VM
  - org.osgi.framework.system.packages
  - org.osgi.framework.system.packages.extra

In the end this has proven to be a difficult problem to implement. It adds a significant amount of complication to the resolver when determining valid uses constraints. If each composite system bundle can export a different set of packages then the resolver needs to represent each system exports as a different bundle but treat it as the same when determining uses constraints (because the classes are actually loaded by the same class loader regardless of what composite the system bundle is in). For example, the org.osgi.framework package will be loaded by the same class loader and it should not matter what composite the constituent bundle is installed. All bundles will use the same Class objects for the org.osgi.framework package.

For now we will limit the specification to ignore the system.packages configuration properties for composite frameworks.

9 Document Support

9.1 References


9.2 Author’s Address
9.3 Acronyms and Abbreviations

9.4 End of Document
Framework Update

Draft

210 Pages

Abstract

10 point Arial Centered.
# 0 Document Information

## 0.1 Table of Contents

0 Document Information .................................................................................................................. 2  
  0.1 Table of Contents .................................................................................................................. 2  
  0.2 Terminology and Document Conventions ............................................................................ 3  
  0.3 Revision History .................................................................................................................... 3  

1 Introduction ............................................................................................................................... 3  

2 Application Domain .................................................................................................................... 4  

3 Problem Description .................................................................................................................. 4  

4 Requirements ............................................................................................................................. 5  

5 Technical Solution ..................................................................................................................... 5  
  5.1 ee.minimum ........................................................................................................................ 6  
  5.2 Framework API .................................................................................................................. 6  
  5.3 Tracker API ........................................................................................................................ 6  

6 Javadoc ..................................................................................................................................... 6  

7 Considered Alternatives .............................................................................................................. 208  
  7.1 Version 2 API .................................................................................................................... 208  
  7.2 Retroweaving ....................................................................................................................... 208  
  7.3 Moving all the PackageAdmin and StartLevel API into the Framework API ..................... 209  
  7.4 Removed BundleAdapter interface ................................................................................... 209  

8 Security Considerations .............................................................................................................. 209  

9 Document Support ..................................................................................................................... 209  
  9.1 References ........................................................................................................................ 209  
  9.2 Author’s Address ................................................................................................................. 209  
  9.3 Acronyms and Abbreviations ............................................................................................ 210  
  9.4 End of Document ............................................................................................................... 210
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 9.1.

Source code is shown in this typeface.

0.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>11/07/09</td>
<td>Initial draft.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BJ Hargrave</td>
</tr>
<tr>
<td>2nd draft</td>
<td>01/08/10</td>
<td>Updated based upon comments from CPEG. Added &quot;adapt&quot; model and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reid PackageAdmin and StartLevel to exploit. Tracker API is also</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updated.</td>
</tr>
<tr>
<td>3rd draft</td>
<td>01/12/10</td>
<td>Updated based upon comments from CPEG Southampton f2f meeting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed BundleAdapter interface from signature of Bundle.adapt method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This allows, for example, for a bundle to be adapted to its</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProtectionDomain. Then all other uses of BundleAdapter were replaced</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with BundleReference which is an interface we already have.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BundlePackageAdmin is changed to return wiring information for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fragments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>isRemovalPending renamed to isCurrent and isStale renamed to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>isInUse in BundleWiring.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Other javadoc tweaks and improvements.</td>
</tr>
<tr>
<td>4th draft</td>
<td>02/25/10</td>
<td>Minor javadoc updates from implementation experience.</td>
</tr>
</tbody>
</table>

1 Introduction
This RFC proposed updates to the core framework API. It is informed by the prototype work done by Peter Kriens and BJ Hargrave for their JavaOne 2009 presentation as well as subsequent discussion and discovery. The changes are to take advantage of the Java 5 generics language feature while preserving compatibility with the Java 1.4 based platforms such as J2ME Foundation 1.1.

2 Application Domain

The Java programming language received a significant update in the Java 5 release. Many features were added, the most well known being generics. Generics provide additional type safety at compile time. While generics provide no additional runtime type safety, finding errors at compile time is very beneficial. Since generics only apply to compile time, they are “erased” at runtime. That is, at runtime, the types are “raw” and the generic information is not part of the Java type system.

Other language features added in Java 5 include enums and annotations. While both of these are useful, they require additions to the class library which were made in Java 5. Enums are subtypes of java.lang.Enum and annotations are subtypes of java.lang.annotation.Annotation. Thus enums and annotations are not “erasable” like generics.

The OSGi API has been based upon the minimum platform of Java 1.4 language and ee.minimum class libraries. Java 1.4 has reached end of life.

J2ME Foundation 1.1 is based upon the Java 1.4 language and derives from the Java 1.4 SE class libraries. ee.minimum is a subset of the Java 1.4 SE class libraries and the J2ME Foundation 1.1 class libraries.

3 Problem Description

Java 5 has been available since 2004. Since the OSGi API is based upon the Java 1.4 language, it has been unable to take advantage of the Java 5 language features. Aside from embedded, most developers today are using Java 5.

There is pressure to exploit Java 5 language features in the OSGi API but OSGi still has an embedded constituency which uses J2ME Foundation 1.1. So there is a tension between these 2 positions.
4 Requirements

1. Exploit Java 5 language features in the OSGi API.
2. Maintain support for the embedded constituency which is still using J2ME Foundation 1.1
3. Backwards compatibility must be maintained with prior versions of the API.
4. Update the framework API to enhance PackageAdmin to improve introspective access to the wiring state.

5 Technical Solution

Since we appear to have conflicting requirements around exploiting Java 5 language features while supporting J2ME Foundation 1.1 which is based upon Java 1.4 language and class libraries, the solution will be limited.

We can only exploit Java 5 language features which do not require class library changes and which do not require VM changes. This basically reduces the choice to generics. Since generics are “erasable” at runtime, the VM is unaware of the generics information. It is stored in attributes which are used by compilers but not the VM.

However, to use generics, one must use a compiler which accepts the “-source 1.5” compiler option. This generally goes hand-in-hand with the “-target 1.5” compiler option which presents several problems. First, the class files generated are version 49.0 which is not consumable by J2ME Foundation 1.1 VMs which are based on Java 1.4 (version 48.0) class files. It also generates String concatenation code using StringBuilder, which was added in Java 5, rather than the old StringBuffer class. Also, enum and annotations are available for use.

Fortunately, there is a largely undocumented compiler option to the rescue: “-target jsr14”[3][4]. This compiler option was created during the development of the Java 5 language features to allow some of them to be used on Java 1.4 runtimes. This compiler option generates version 48.0 class files which include generic signature attributes that will run on Java 1.4 based class libraries. These class files can be loaded on Java 1.4 based VMs and can be used for compiling Java 5 code to access the generic signature attributes.

So the technical solution is based upon using the "-target jsr14" compiler option to create class files for the OSGi API which execute on Java 1.4 based environments but exploit generics for the constituency using Java 5. So while not exploiting all the new Java 5 language features, the OSGi API will look more modern while still supporting the embedded constituency.
5.1 ee.minimum

To properly take advantage of generics, we must compile the OSGi API using generified class libraries. However using the Java 5 class libraries exposed us to making mistakes and using classes or members which do not exist in J2ME Foundation 1.1.

Since we already use ee.minimum 1.2 in most place to prevent this sort of mistake, we must update ee.minimum to add generic signatures without introducing any new API. That is, the updated ee.minimum, after erasing the generic signatures, must be identical to ee.minimum 1.2.

This has been implementation and is now available as ee.minimum 1.2.1 in the build.

5.2 Framework API

The framework package is updated in several ways. First generics signatures are added to existing API where appropriate. Second, new API is added to exploit generic type safety. This is around the service registry API which is where types objects are used in the API. Finally, in order to tidy up the overall framework API, an adapt method was added to the Bundle interface to allow it to be adapted to PackageAdmin and StartLevel types without the need to get those services from the service registry. The PackageAdmin service API and the Startlevel service API are updated to work with the updated framework API. This includes introducing a BundleWiring type which represents the resolved state of a bundle. Several methods on the PackageAdmin and Startlevel services become deprecated.

5.3 Tracker API

The tracker API is also updated to be generified. A new constructor is added to ServiceTracker which take a class argument for type safety.

New getTracked methods are added to both BundleTracker and ServiceTracker to return a map for keys to values.

6 Javadoc
<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.osgi.framework.launch</td>
<td>Framework Launch Package Version 1.0.</td>
<td>153</td>
</tr>
<tr>
<td>org.osgi.service.packageadmin</td>
<td>Package Admin Package Version 1.3.</td>
<td>162</td>
</tr>
<tr>
<td>org.osgi.service.startlevel</td>
<td>Start Level Package Version 1.2.</td>
<td>183</td>
</tr>
<tr>
<td>org.osgi.util.tracker</td>
<td>Tracker Package Version 1.5.</td>
<td>191</td>
</tr>
</tbody>
</table>
## Package org.osgi.framework


See:  
- [Description](#)

### Interface Summary

<table>
<thead>
<tr>
<th><strong>Package</strong></th>
<th><strong>Class</strong></th>
<th><strong>Description</strong></th>
<th><strong>Page</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>AllServiceListerner</td>
<td><strong>AllServiceListerner</strong></td>
<td>A ServiceEvent listener that does not filter based upon package wiring.</td>
<td>16</td>
</tr>
<tr>
<td>Bundle</td>
<td><strong>Bundle</strong></td>
<td>An installed bundle in the Framework.</td>
<td>17</td>
</tr>
<tr>
<td>BundleActivator</td>
<td><strong>BundleActivator</strong></td>
<td>Customizes the starting and stopping of a bundle.</td>
<td>37</td>
</tr>
<tr>
<td>BundleContext</td>
<td><strong>BundleContext</strong></td>
<td>A bundle's execution context within the Framework.</td>
<td>39</td>
</tr>
<tr>
<td>BundleListener</td>
<td><strong>BundleListener</strong></td>
<td>A BundleEvent listener.</td>
<td>64</td>
</tr>
<tr>
<td>BundleReference</td>
<td><strong>BundleReference</strong></td>
<td>A reference to a Bundle.</td>
<td>69</td>
</tr>
<tr>
<td>Configurable</td>
<td><strong>Configurable</strong></td>
<td>Deprecated. As of 1.2.</td>
<td>70</td>
</tr>
<tr>
<td>Constants</td>
<td><strong>Constants</strong></td>
<td>Defines standard names for the OSGi environment system properties, service properties, and Manifest header attribute keys.</td>
<td>71</td>
</tr>
<tr>
<td>Filter</td>
<td><strong>Filter</strong></td>
<td>An RFC 1960-based Filter.</td>
<td>101</td>
</tr>
<tr>
<td>FrameworkListener</td>
<td><strong>FrameworkListener</strong></td>
<td>A FrameworkEvent listener.</td>
<td>109</td>
</tr>
<tr>
<td>ServiceFactory</td>
<td><strong>ServiceFactory</strong></td>
<td>Allows services to provide customized service objects in the OSGi environment.</td>
<td>128</td>
</tr>
<tr>
<td>ServiceListener</td>
<td><strong>ServiceListener</strong></td>
<td>A ServiceEvent listener.</td>
<td>130</td>
</tr>
<tr>
<td>ServiceReference</td>
<td><strong>ServiceReference</strong></td>
<td>A reference to a service.</td>
<td>135</td>
</tr>
<tr>
<td>ServiceRegistration</td>
<td><strong>ServiceRegistration</strong></td>
<td>A registered service.</td>
<td>139</td>
</tr>
<tr>
<td>SynchronousBundleListener</td>
<td><strong>SynchronousBundleListener</strong></td>
<td>A synchronous BundleEvent listener.</td>
<td>141</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th><strong>Package</strong></th>
<th><strong>Class</strong></th>
<th><strong>Description</strong></th>
<th><strong>Page</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminPermission</td>
<td><strong>AdminPermission</strong></td>
<td>A bundle's authority to perform specific privileged administrative operations on or to get sensitive information about a bundle.</td>
<td>10</td>
</tr>
<tr>
<td>BundleEvent</td>
<td><strong>BundleEvent</strong></td>
<td>An event from the Framework describing a bundle lifecycle change.</td>
<td>53</td>
</tr>
<tr>
<td>BundlePermission</td>
<td><strong>BundlePermission</strong></td>
<td>A bundle's authority to require or provide a bundle or to receive or attach fragments.</td>
<td>65</td>
</tr>
<tr>
<td>FrameworkEvent</td>
<td><strong>FrameworkEvent</strong></td>
<td>A general event from the Framework.</td>
<td>104</td>
</tr>
<tr>
<td>FrameworkUtil</td>
<td><strong>FrameworkUtil</strong></td>
<td>Framework Utility class.</td>
<td>110</td>
</tr>
<tr>
<td>PackagePermission</td>
<td><strong>PackagePermission</strong></td>
<td>A bundle's authority to import or export a package.</td>
<td>116</td>
</tr>
<tr>
<td>ServiceEvent</td>
<td><strong>ServiceEvent</strong></td>
<td>An event from the Framework describing a service lifecycle change.</td>
<td>121</td>
</tr>
<tr>
<td>ServicePermission</td>
<td><strong>ServicePermission</strong></td>
<td>A bundle's authority to register or get a service.</td>
<td>131</td>
</tr>
<tr>
<td>Version</td>
<td><strong>Version</strong></td>
<td>Version identifier for bundles and packages.</td>
<td>142</td>
</tr>
</tbody>
</table>

### Exception Summary

<table>
<thead>
<tr>
<th><strong>Package</strong></th>
<th><strong>Class</strong></th>
<th><strong>Description</strong></th>
<th><strong>Page</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleException</td>
<td><strong>BundleException</strong></td>
<td>A Framework exception used to indicate that a bundle lifecycle problem occurred.</td>
<td>58</td>
</tr>
<tr>
<td>InvalidSyntaxException</td>
<td><strong>InvalidSyntaxException</strong></td>
<td>A Framework exception used to indicate that a filter string has an invalid syntax.</td>
<td>113</td>
</tr>
</tbody>
</table>
Package org.osgi.framework Description


Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest.

For example:

Import-Package: org.osgi.framework;version="[1.6,2.0)"
Class AdminPermission

org.osgi.framework

java.lang.Object
  java.security.Permission
    java.security.BasicPermission
      org.osgi.framework.AdminPermission

All Implemented Interfaces:
  Guard, Serializable

final public class AdminPermission
  extends BasicPermission

A bundle's authority to perform specific privileged administrative operations on or to get sensitive information about
a bundle. The actions for this permission are:

<table>
<thead>
<tr>
<th>Action</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>class</td>
<td>Bundle.loadClass</td>
</tr>
<tr>
<td>execute</td>
<td>Bundle.start</td>
</tr>
<tr>
<td></td>
<td>Bundle.stop</td>
</tr>
<tr>
<td></td>
<td>StartLevel.setBundleStartLevel</td>
</tr>
<tr>
<td>extensionLifecycle</td>
<td>BundleContext.installBundle for extension bundles</td>
</tr>
<tr>
<td></td>
<td>Bundle.update for extension bundles</td>
</tr>
<tr>
<td></td>
<td>Bundle.uninstall for extension bundles</td>
</tr>
<tr>
<td>lifecycle</td>
<td>BundleContext.installBundle</td>
</tr>
<tr>
<td></td>
<td>Bundle.update</td>
</tr>
<tr>
<td></td>
<td>Bundle.uninstall</td>
</tr>
<tr>
<td>listener</td>
<td>BundleContext.addBundleListener for SynchronousBundleListener</td>
</tr>
<tr>
<td></td>
<td>BundleContext.removeBundleListener for SynchronousBundleListener</td>
</tr>
<tr>
<td>metadata</td>
<td>Bundle.getHeaders</td>
</tr>
<tr>
<td></td>
<td>Bundle.getLocation</td>
</tr>
<tr>
<td>resolve</td>
<td>PackageAdmin.refreshPackages</td>
</tr>
<tr>
<td></td>
<td>PackageAdmin.resolveBundles</td>
</tr>
<tr>
<td>resource</td>
<td>Bundle.getResource</td>
</tr>
<tr>
<td></td>
<td>Bundle.getResources</td>
</tr>
<tr>
<td></td>
<td>Bundle.getEntry</td>
</tr>
<tr>
<td></td>
<td>Bundle.getEntryPaths</td>
</tr>
<tr>
<td></td>
<td>Bundle.findEntries</td>
</tr>
<tr>
<td>startlevel</td>
<td>Bundle resource/entry URL creation</td>
</tr>
<tr>
<td></td>
<td>StartLevel.setStartLevel</td>
</tr>
<tr>
<td>context</td>
<td>Bundle.getBundleContext</td>
</tr>
<tr>
<td></td>
<td>StartLevel.setInitialBundleStartLevel</td>
</tr>
</tbody>
</table>

The special action "*" will represent all actions. The resolve action is implied by the class, execute and resource actions.

The name of this permission is a filter expression. The filter gives access to the following attributes:

- signer - A Distinguished Name chain used to sign a bundle. Wildcards in a DN are not matched according
to the filter string rules, but according to the rules defined for a DN chain.
- location - The location of a bundle.
- id - The bundle ID of the designated bundle.
- name - The symbolic name of a bundle.

Filter attribute names are processed in a case sensitive manner.

Version:
SVRevision: 8338 $
### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
</table>
| static String **CLASS**  
The action string class. | 11 |
| static String **CONTEXT**  
The action string context. | 13 |
| static String **EXECUTE**  
The action string execute. | 12 |
| static String **EXTENSIONLIFECYCLE**  
The action string extensionLifecycle. | 12 |
| static String **LIFECYCLE**  
The action string lifecycle. | 12 |
| static String **LISTENER**  
The action string listener. | 12 |
| static String **METADATA**  
The action string metadata. | 12 |
| static String **RESOLVE**  
The action string resolve. | 12 |
| static String **RESOURCE**  
The action string resource. | 13 |
| static String **STARTLEVEL**  
The action string startlevel. | 13 |

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
</table>
| **AdminPermission()**  
Creates a new AdminPermission object that matches all bundles and has all actions. | 13 |
| **AdminPermission(String filter, String actions)**  
Create a new AdminPermission. | 13 |
| **AdminPermission(Bundle bundle, String actions)**  
Create a new requested AdminPermission object to be used by the code that must perform checkPermission. | 14 |

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
</table>
| boolean **equals**(Object obj)  
Determines the equality of two AdminPermission objects. | 15 |
| String **getActions**()  
Returns the canonical string representation of the AdminPermission actions. | 15 |
| int **hashCode**()  
Returns the hash code value for this object. | 15 |
| boolean **implies**(Permission p)  
Determines if the specified permission is implied by this object. | 14 |
| Permission Collection **newPermissionCollection**()  
Returns a new PermissionCollection object suitable for storing AdminPermissionS. | 15 |

### Field Detail

**CLASS**

```java
public static final String CLASS = "class"
```

The action string class. The class action implies the resolve action.
**EXECUTE**

```java
public static final String EXECUTE = "execute"
```

The action string execute. The execute action implies the resolve action.

Since: 1.3

**EXTENSIONLIFECYCLE**

```java
public static final String EXTENSIONLIFECYCLE = "extensionLifecycle"
```

The action string extensionLifecycle.

Since: 1.3

**LIFECYCLE**

```java
public static final String LIFECYCLE = "lifecycle"
```

The action string lifecycle.

Since: 1.3

**LISTENER**

```java
public static final String LISTENER = "listener"
```

The action string listener.

Since: 1.3

**METADATA**

```java
public static final String METADATA = "metadata"
```

The action string metadata.

Since: 1.3

**RESOLVE**

```java
public static final String RESOLVE = "resolve"
```
The action string resolve. The resolve action is implied by the class, execute and resource actions.

Since: 1.3

**RESOURCE**

public static final String RESOURCE = "resource"

The action string resource. The resource action implies the resolve action.

Since: 1.3

**STARTLEVEL**

public static final String STARTLEVEL = "startlevel"

The action string startlevel.

Since: 1.3

**CONTEXT**

public static final String CONTEXT = "context"

The action string context.

Since: 1.4

### Constructor Detail

**AdminPermission**

public AdminPermission()

Creates a new AdminPermission object that matches all bundles and has all actions. Equivalent to AdminPermission("*", "*");

**AdminPermission**

public AdminPermission(String filter,
                        String actions)

Create a new AdminPermission. This constructor must only be used to create a permission that is going to be checked.

Examples:

(signer=\\*,o=ACME,c=US)
&{signer=\\*,o=ACME,c=US}{name=com.acme.*}(location=http://www.acme.com/bundles/*))
(id=-1)
When a signer key is used within the filter expression the signer value must escape the special filter chars (*, (, )).

Null arguments are equivalent to "*".

Parameters:
- `filter` - A filter expression that can use signer, location, id, and name keys. A value of "*" or null matches all bundle. Filter attribute names are processed in a case sensitive manner.
- `actions` - class, execute, extensionLifecycle, lifecycle, listener, metadata, resolve, resource, startlevel or context. A value of "*" or null indicates all actions.

Throws:
- `IllegalArgumentException` - If the filter has an invalid syntax.

AdminPermission

public AdminPermission(Bundle bundle,
                      String actions)

Creates a new requested AdminPermission object to be used by the code that must perform checkPermission. AdminPermission objects created with this constructor cannot be added to an AdminPermission permission collection.

Parameters:
- `bundle` - A bundle.
- `actions` - class, execute, extensionLifecycle, lifecycle, listener, metadata, resolve, resource, startlevel, context. A value of "*" or null indicates all actions.

Since:
1.3

Method Detail

implies

public boolean implies(Permission p)

Determines if the specified permission is implied by this object. This method throws an exception if the specified permission was not constructed with a bundle.

This method returns `true` if the specified permission is an AdminPermission AND

- this object's filter matches the specified permission's bundle ID, bundle symbolic name, bundle location and bundle signer distinguished name chain OR
- this object's filter is "*"

AND this object's actions include all of the specified permission's actions.

Special case: if the specified permission was constructed with "*" filter, then this method returns `true` if this object's filter is "*" and this object's actions include all of the specified permission's actions.

Overrides:
- `implies in class BasicPermission`

Parameters:
- `p` - The requested permission.

Returns:
- `true` if the specified permission is implied by this object; `false` otherwise.
getActions

public String getActions()

Returns the canonical string representation of the AdminPermission actions.

Always returns present AdminPermission actions in the following order: class, execute, extensionLifecycle, lifecycle, listener, metadata, resolve, resource, startlevel, context.

Overrides:
getActions in class BasicPermission

Returns:
Canonical string representation of the AdminPermission actions.

newPermissionCollection

public PermissionCollection newPermissionCollection()

Returns a new PermissionCollection object suitable for storing AdminPermissionS.

Overrides:
newPermissionCollection in class BasicPermission

Returns:
A new PermissionCollection object.

equals

public boolean equals(Object obj)

Determines the equality of two AdminPermission objects.

Overrides:
equals in class BasicPermission

Parameters:
obj - The object being compared for equality with this object.

Returns:
true if obj is equivalent to this AdminPermission; false otherwise.

hashCode

public int hashCode()

Returns the hash code value for this object.

Overrides:
hashCode in class BasicPermission

Returns:
Hash code value for this object.
public interface AllServiceListener
extends ServiceListener

A ServiceEvent listener that does not filter based upon package wiring. AllServiceListener is a listener interface that may be implemented by a bundle developer. When a ServiceEvent is fired, it is synchronously delivered to an AllServiceListener. The Framework may deliver ServiceEvent objects to an AllServiceListener out of order and may concurrently call and/or reenter an AllServiceListener.

An AllServiceListener object is registered with the Framework using the BundleContext.addServiceListener method. AllServiceListener objects are called with a ServiceEvent object when a service is registered, modified, or is in the process of unregistering.

ServiceEvent object delivery to AllServiceListener objects is filtered by the filter specified when the listener was registered. If the Java Runtime Environment supports permissions, then additional filtering is done. ServiceEvent objects are only delivered to the listener if the bundle which defines the listener object's class has the appropriate ServicePermission to get the service using at least one of the named classes under which the service was registered.

Unlike normal ServiceListener objects, AllServiceListener objects receive all ServiceEvent objects regardless of whether the package source of the listening bundle is equal to the package source of the bundle that registered the service. This means that the listener may not be able to cast the service object to any of its corresponding service interfaces if the service object is retrieved.

Since:
1.3
Version:
$Revision: 5673 $
See Also:
ServiceEvent, ServicePermission
ThreadSafe

Methods inherited from interface org.osgi.framework.ServiceListener
serviceChanged
public interface Bundle
extends Comparable(Bundle>

An installed bundle in the Framework.

A Bundle object is the access point to define the lifecycle of an installed bundle. Each bundle installed in the OSGi environment must have an associated Bundle object.

A bundle must have a unique identity, a long, chosen by the Framework. This identity must not change during the lifecycle of a bundle, even when the bundle is updated. Uninstalling and then reinstalling the bundle must create a new unique identity.

A bundle can be in one of six states:

- UNINSTALLED
- INSTALLED
- RESOLVED
- STARTING
- STOPPING
- ACTIVE

Values assigned to these states have no specified ordering; they represent bit values that may be ORed together to determine if a bundle is in one of the valid states.

A bundle should only execute code when its state is one of STARTING, ACTIVE, or STOPPING. An UNINSTALLED bundle can not be set to another state; it is a zombie and can only be reached because references are kept somewhere.

The Framework is the only entity that is allowed to create Bundle objects, and these objects are only valid within the Framework that created them.

Bundles have a natural ordering such that if two Bundles have the same bundle_id they are equal. A Bundle is less than another Bundle if it has a lower bundle_id and is greater if it has a higher bundle id.

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int ACTIVE</td>
<td>The bundle is now running.</td>
<td>21</td>
</tr>
<tr>
<td>int INSTALLED</td>
<td>The bundle is installed but not yet resolved.</td>
<td>19</td>
</tr>
<tr>
<td>int RESOLVED</td>
<td>The bundle is resolved and is able to be started.</td>
<td>20</td>
</tr>
<tr>
<td>int SIGNERS_ALL</td>
<td>Request that all certificates used to sign the bundle be returned.</td>
<td>21</td>
</tr>
</tbody>
</table>
Interface Bundle

| int | SIGNERS_TRUSTED | Request that only certificates used to sign the bundle that are trusted by the framework be returned. |
| int | START_ACTIVATION_POLICY | The bundle start operation must activate the bundle according to the bundle's declared activation policy. |
| int | START_TRANSIENT | The bundle start operation is transient and the persistent autostart setting of the bundle is not modified. |
| int | STARTING | The bundle is in the process of starting. |
| int | STOP_TRANSIENT | The bundle stop is transient and the persistent autostart setting of the bundle is not modified. |
| int | STOPPING | The bundle is in the process of stopping. |
| int | TYPE_FRAGMENT | Bundle type indicating the bundle is a fragment bundle. |
| int | UNINSTALLED | The bundle is uninstalled and may not be used. |

### Method Summary

<table>
<thead>
<tr>
<th>A</th>
<th>adapt(Class type)</th>
<th>Adapt a bundle to the specified type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration&lt;URL&gt;</td>
<td>findEntries(String path, String filePattern, boolean recurse)</td>
<td>Returns entries in this bundle and its attached fragments.</td>
</tr>
<tr>
<td>BundleContext</td>
<td>getBundleContext()</td>
<td>Returns this bundle's BundleContext.</td>
</tr>
<tr>
<td>long</td>
<td>getBundleId()</td>
<td>Returns this bundle's unique identifier.</td>
</tr>
<tr>
<td>URL</td>
<td>getEntry(String path)</td>
<td>Returns a URL to the entry at the specified path in this bundle.</td>
</tr>
<tr>
<td>Enumeration&lt;String&gt;</td>
<td>getEntryPaths(String path)</td>
<td>Returns an Enumeration of all the paths (String objects) to entries within this bundle whose longest sub-path matches the specified path.</td>
</tr>
<tr>
<td>Dictionary&lt;String,String&gt;</td>
<td>getHeaders()</td>
<td>Returns this bundle's Manifest headers and values.</td>
</tr>
<tr>
<td>Dictionary&lt;String,String&gt;</td>
<td>getHeaders(String locale)</td>
<td>Returns this bundle's Manifest headers and values localized to the specified locale.</td>
</tr>
<tr>
<td>long</td>
<td>getLastModified()</td>
<td>Returns the time when this bundle was last modified.</td>
</tr>
<tr>
<td>String</td>
<td>getLocation()</td>
<td>Returns this bundle's location identifier.</td>
</tr>
<tr>
<td>ServiceReference&lt;[]&gt;</td>
<td>getRegisteredServices()</td>
<td>Returns this bundle's ServiceReference list for all services it has registered or null if this bundle has no registered services.</td>
</tr>
<tr>
<td>URL</td>
<td>getResource(String name)</td>
<td>Find the specified resource from this bundle's class loader.</td>
</tr>
<tr>
<td>Enumeration&lt;URL&gt;</td>
<td>getResources(String name)</td>
<td>Find the specified resources from this bundle's class loader.</td>
</tr>
<tr>
<td>ServiceReference&lt;[]&gt;</td>
<td>getServicesInUse()</td>
<td>Returns this bundle's ServiceReference list for all services it is using or returns null if this bundle is not using any services.</td>
</tr>
</tbody>
</table>
Map<X509Certificate, List<X509Certificate>>

**getSignerCertificates**(int signersType)

Return the certificates for the signers of this bundle and the certificate chains for those signers.

**getState**()

Returns this bundle's current state.

**getSymbolicName**()

Returns the symbolic name of this bundle as specified by its Bundle-SymbolicName manifest header.

**getTypes**()

Returns the special types of this bundle.

**getVersion**()

Returns the version of this bundle as specified by its Bundle-Version manifest header.

**hasPermission**(Object permission)

Determines if this bundle has the specified permissions.

**loadClass**(String name)

Loads the specified class using this bundle's class loader.

**start**()

Starts this bundle with no options.

**start**(int options)

Starts this bundle.

**stop**()

Stops this bundle with no options.

**stop**(int options)

stops this bundle.

**uninstall**()

Uninstalls this bundle.

**update**()

Updates this bundle.

**update**(InputStream input)

Updates this bundle from an InputStream.

<table>
<thead>
<tr>
<th>Field Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UNINSTALLED</strong></td>
</tr>
<tr>
<td>public static final int UNINSTALLED = 1</td>
</tr>
</tbody>
</table>

The bundle is uninstalled and may not be used. The UNINSTALLED state is only visible after a bundle is uninstalled; the bundle is in an unusable state but references to the Bundle object may still be available and used for introspection. The value of UNINSTALLED is 0x00000001.

<table>
<thead>
<tr>
<th>INSTALLATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>public static final int INSTALLED = 2</td>
</tr>
</tbody>
</table>

The bundle is installed but not yet resolved. A bundle is in the INSTALLED state when it has been installed in the Framework but is not or cannot be resolved.
This state is visible if the bundle's code dependencies are not resolved. The Framework may attempt to resolve an INSTALLED bundle's code dependencies and move the bundle to the RESOLVED state.

The value of INSTALLED is 0x00000002.

RESOLVED

public static final int RESOLVED = 4

The bundle is resolved and is able to be started.

A bundle is in the RESOLVED state when the Framework has successfully resolved the bundle's code dependencies. These dependencies include:

- The bundle's class path from its Constants.BUNDLE_CLASSPATH Manifest header.
- The bundle's package dependencies from its Constants.EXPORT_PACKAGE and Constants.IMPORT_PACKAGE Manifest headers.
- The bundle's required bundle dependencies from its Constants.REQUIRE_BUNDLE Manifest header.
- A fragment bundle's host dependency from its Constants.FRAGMENT_HOST Manifest header.

Note that the bundle is not active yet. A bundle must be put in the RESOLVED state before it can be started. The Framework may attempt to resolve a bundle at any time.

The value of RESOLVED is 0x00000004.

STARTING

public static final int STARTING = 8

The bundle is in the process of starting.

A bundle is in the STARTING state when its start method is active. A bundle must be in this state when the bundle’s BundleActivator.start() is called. If the BundleActivator.start method completes without exception, then the bundle has successfully started and must move to the ACTIVE state.

If the bundle has a lazy activation policy, then the bundle may remain in this state for some time until the activation is triggered.

The value of STARTING is 0x00000008.

STOPPING

public static final int STOPPING = 16

The bundle is in the process of stopping.

A bundle is in the STOPPING state when its stop method is active. A bundle must be in this state when the bundle's BundleActivator.stop() method is called. When the BundleActivator.stop method completes the bundle is stopped and must move to the RESOLVED state.

The value of STOPPING is 0x00000010.
**ACTIVE**

public static final int **ACTIVE** = 32

The bundle is now running.

A bundle is in the **ACTIVE** state when it has been successfully started and activated.

The value of **ACTIVE** is 0x00000020.

**START_TRANSIENT**

public static final int **START_TRANSIENT** = 1

The bundle start operation is transient and the persistent autostart setting of the bundle is not modified.

This bit may be set when calling `start(int)` to notify the framework that the autostart setting of the bundle must not be modified. If this bit is not set, then the autostart setting of the bundle is modified.

Since: 1.4
See Also: `start(int)`

**START_ACTIVATION_POLICY**

public static final int **START_ACTIVATION_POLICY** = 2

The bundle start operation must activate the bundle according to the bundle's declared activation policy.

This bit may be set when calling `start(int)` to notify the framework that the bundle must be activated using the bundle's declared activation policy.

Since: 1.4
See Also: `Constants.BUNDLE_ACTIVATIONPOLICY, start(int)`

**STOP_TRANSIENT**

public static final int **STOP_TRANSIENT** = 1

The bundle stop is transient and the persistent autostart setting of the bundle is not modified.

This bit may be set when calling `stop(int)` to notify the framework that the autostart setting of the bundle must not be modified. If this bit is not set, then the autostart setting of the bundle is modified.

Since: 1.4
See Also: `stop(int)`

**SIGNERS_ALL**

public static final int **SIGNERS_ALL** = 1
Request that all certificates used to sign the bundle be returned.

Since: 1.5
See Also: getSignerCertificates(int)

SIGNERS_TRUSTED

public static final int SIGNERS_TRUSTED = 2

Request that only certificates used to sign the bundle that are trusted by the framework be returned.

Since: 1.5
See Also: getSignerCertificates(int)

TYPE_FRAGMENT

public static final int TYPE_FRAGMENT = 1

Bundle type indicating the bundle is a fragment bundle.

Since: 1.6
See Also: getTypes()

Method Detail

getState

int getState()

Returns this bundle's current state.

A bundle can be in only one state at any time.

Returns: An element of UNINSTALLED, INSTALLED, RESOLVED, STARTING, STOPPING, ACTIVE.

start

void start(int options)
   throws BundleException

Starts this bundle.

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

If the Framework implements the optional Start Level service and the current start level is less than this bundle's start level:

- If the START_TRANSIENT option is set, then a BundleException is thrown indicating this bundle cannot be started due to the Framework's current start level.
Otherwise, the Framework must set this bundle's persistent autostart setting to *Started with declared activation* if the `START_ACTIVATION_POLICY` option is set or *Started with eager activation* if not set.

When the Framework's current start level becomes equal to or more than this bundle's start level, this bundle will be started.

Otherwise, the following steps are required to start this bundle:

1. If this bundle is in the process of being activated or deactivated then this method must wait for activation or deactivation to complete before continuing. If this does not occur in a reasonable time, a `BundleException` is thrown to indicate this bundle was unable to be started.
2. If this bundle's state is `ACTIVE` then this method returns immediately.
3. If the `START_TRANSIENT` option is not set then set this bundle's autostart setting to *Started with declared activation* if the `START_ACTIVATION_POLICY` option is set or *Started with eager activation* if not set. When the Framework is restarted and this bundle's autostart setting is not `Stopped`, this bundle must be automatically started.
4. If this bundle's state is not `RESOLVED`, an attempt is made to resolve this bundle. If the Framework cannot resolve this bundle, a `BundleException` is thrown.
5. If the `START_ACTIVATION_POLICY` option is set and this bundle's declared activation policy is *lazy* then:
   - If this bundle's state is `STARTING` then this method returns immediately.
   - This bundle's state is set to `STARTING`.
   - A bundle event of type `BundleEvent.LAZY_ACTIVATION` is fired.
   - This method returns immediately and the remaining steps will be followed when this bundle's activation is later triggered.
6. This bundle's state is set to `STARTING`.
7. A bundle event of type `BundleEvent.STARTING` is fired.
8. The `BundleActivator.start()` method of this bundle's `BundleActivator`, if one is specified, is called. If the `BundleActivator` is invalid or throws an exception then:
   - This bundle's state is set to `STOPPING`.
   - A bundle event of type `BundleEvent.STOPPING` is fired.
   - Any services registered by this bundle must be unregistered.
   - Any services used by this bundle must be released.
   - Any listeners registered by this bundle must be removed.
   - This bundle's state is set to `RESOLVED`.
   - A bundle event of type `BundleEvent.STOPPED` is fired.
   - A `BundleException` is then thrown.
9. If this bundle's state is `UNINSTALLED`, because this bundle was uninstalled while the `BundleActivator.start()` method was running, a `BundleException` is thrown.
10. This bundle's state is set to `ACTIVE`.
11. A bundle event of type `BundleEvent.STARTED` is fired.

**Preconditions**

- `getState()` in `{ INSTALLED, RESOLVED }` or `{ INSTALLED, RESOLVED, STARTING }` if this bundle has a lazy activation policy.

**Postconditions, no exceptions thrown**

- Bundle autostart setting is modified unless the `START_TRANSIENT` option was set.
- `getState()` in `{ ACTIVE }` unless the lazy activation policy was used.
- `BundleActivator.start()` has been called and did not throw an exception unless the lazy activation policy was used.

**Postconditions, when an exception is thrown**

- Depending on when the exception occurred, bundle autostart setting is modified unless the `START_TRANSIENT` option was set.
- `getState()` not in `{ STARTING, ACTIVE }`.
Parameters:

- **options** - The options for starting this bundle. See `START_TRANSIENT` and `START_ACTIVATION_POLICY`. The Framework must ignore unrecognized options.

Throws:

- `BundleException` - If this bundle could not be started. This could be because a code dependency could not be resolved or the specified BundleActivator could not be loaded or threw an exception or this bundle is a fragment.
- `IllegalStateException` - If this bundle has been uninstalled or this bundle tries to change its own state.
- `SecurityException` - If the caller does not have the appropriate AdminPermission[this, EXECUTE], and the Java Runtime Environment supports permissions.

Since: 1.4

**start**

```java
void start() throws BundleException
```

Starts this bundle with no options.

This method performs the same function as calling `start(0)`.

Throws:

- `BundleException` - If this bundle could not be started. This could be because a code dependency could not be resolved or the specified BundleActivator could not be loaded or threw an exception or this bundle is a fragment.
- `IllegalStateException` - If this bundle has been uninstalled or this bundle tries to change its own state.
- `SecurityException` - If the caller does not have the appropriate AdminPermission[this, EXECUTE], and the Java Runtime Environment supports permissions.

See Also:

- `start(int)`

**stop**

```java
void stop(int options) throws BundleException
```

Stops this bundle.

The following steps are required to stop a bundle:

1. If this bundle's state is UNINSTALLED then an `IllegalStateException` is thrown.
2. If this bundle is in the process of being activated or deactivated then this method must wait for activation or deactivation to complete before continuing. If this does not occur in a reasonable time, a BundleException is thrown to indicate this bundle was unable to be stopped.
3. If the `STOP_TRANSIENT` option is not set then set this bundle's persistent autostart setting to `Stopped`. When the Framework is restarted and this bundle's autostart setting is `Stopped`, this bundle must not be automatically started.
4. If this bundle's state is not `STARTING` or `ACTIVE` then this method returns immediately.
5. This bundle's state is set to `STOPPING`.
6. A bundle event of type `BundleEvent.STOPPING` is fired.
7. If this bundle's state was `ACTIVE` prior to setting the state to `STOPPING`, the `BundleActivator.stop()` method of this bundle's BundleActivator, if one is specified, is called. If that method throws an exception, this method must continue to stop this bundle and a BundleException must be thrown after completion of the remaining steps.
8. Any services registered by this bundle must be unregistered.
9. Any services used by this bundle must be released.
10. Any listeners registered by this bundle must be removed.
11. If this bundle's state is **UNINSTALLED**, because this bundle was uninstalled while the `BundleActivator.stop` method was running, a `BundleException` must be thrown.

12. This bundle's state is set to **RESOLVED**.

13. A bundle event of type `BundleEvent.STOPPED` is fired.

**Preconditions**

- `getState()` in `{ACTIVE}`.

**Postconditions, no exceptions thrown**

- Bundle autostart setting is modified unless the `STOP_TRANSIENT` option was set.
- `getState()` not in `{ACTIVE, STOPPING}`.
- `BundleActivator.stop` has been called and did not throw an exception.

**Postconditions, when an exception is thrown**

- Bundle autostart setting is modified unless the `STOP_TRANSIENT` option was set.

**Parameters:**

- `options` - The options for stopping this bundle. See `STOP_TRANSIENT`. The Framework must ignore unrecognized options.

**Throws:**

- `BundleException` - If this bundle's `BundleActivator` threw an exception or this bundle is a fragment.
- `IllegalStateException` - If this bundle has been uninstalled or this bundle tries to change its own state.
- `SecurityException` - If the caller does not have the appropriate `AdminPermission[this,EXECUTE]`, and the Java Runtime Environment supports permissions.

**Since:**

1.4

---

**stop**

```java
void stop() throws BundleException
```

Stops this bundle with no options.

This method performs the same function as calling `stop(0)`.

**Throws:**

- `BundleException` - If this bundle's `BundleActivator` threw an exception or this bundle is a fragment.
- `IllegalStateException` - If this bundle has been uninstalled or this bundle tries to change its own state.
- `SecurityException` - If the caller does not have the appropriate `AdminPermission[this,EXECUTE]`, and the Java Runtime Environment supports permissions.

**See Also:**

`start(int)`

---

**update**

```java
void update(InputStream input) throws BundleException
```

Updates this bundle from an `InputStream`. 

If the specified InputStream is null, the Framework must create the InputStream from which to read the updated bundle by interpreting, in an implementation dependent manner, this bundle's Bundle-UpdateLocation Manifest header, if present, or this bundle's original location.

If this bundle's state is ACTIVE, it must be stopped before the update and started after the update successfully completes.

If this bundle has exported any packages that are imported by another bundle, these packages must remain exported until the PackageAdmin.refreshPackages method has been has been called or the Framework is relaunched.

The following steps are required to update a bundle:

1. If this bundle's state is UNINSTALLED then an IllegalStateException is thrown.
2. If this bundle's state is ACTIVE, STARTING or STOPPING, this bundle is stopped as described in the Bundle.stop method. If Bundle.stop throws an exception, the exception is rethrown terminating the update.
3. The updated version of this bundle is read from the input stream and installed. If the Framework is unable to install the updated version of this bundle, the original version of this bundle must be restored and a BundleException must be thrown after completion of the remaining steps.
4. This bundle's state is set to INSTALLED.
5. If the updated version of this bundle was successfully installed, a bundle event of type BundleEvent.UPDATED is fired.
6. If this bundle's state was originally ACTIVE, the updated bundle is started as described in the Bundle.start method. If Bundle.start throws an exception, a Framework event of type FrameworkEvent.ERROR is fired containing the exception.

Preconditions

- getState() not in {UNINSTALLED}.

Postconditions, no exceptions thrown

- getState() in {INSTALLED, RESOLVED, ACTIVE}.
- This bundle has been updated.

Postconditions, when an exception is thrown

- getState() in {INSTALLED, RESOLVED, ACTIVE}.
- Original bundle is still used; no update occurred.

Parameters:

- input - The InputStream from which to read the new bundle or null to indicate the Framework must create the input stream from this bundle's Bundle-UpdateLocation Manifest header, if present, or this bundle's original location. The input stream must always be closed when this method completes, even if an exception is thrown.

Throws:

- BundleException - If the input stream cannot be read or the update fails.
- IllegalArgumentException - If this bundle has been uninstalled or this bundle tries to change its own state.
- SecurityException - If the caller does not have the appropriate AdminPermission[this,LIFECYCLE] for both the current bundle and the updated bundle, and the Java Runtime Environment supports permissions.

See Also:

stop(), start()

update

void update() throws BundleException
Updates this bundle.

This method performs the same function as calling `update(InputStream)` with a null InputStream.

**Throws:**
- `BundleException` - If the update fails.
- `IllegalStateException` - If this bundle has been uninstalled or this bundle tries to change its own state.
- `SecurityException` - If the caller does not have the appropriate `AdminPermission[this,LIFECYCLE]` for both the current bundle and the updated bundle, and the Java Runtime Environment supports permissions.

**See Also:**
- `update(InputStream)`

### uninstall

```java
void uninstall() throws BundleException
```

Uninstalls this bundle.

This method causes the Framework to notify other bundles that this bundle is being uninstalled, and then puts this bundle into the **UNINSTALLED** state. The Framework must remove any resources related to this bundle that it is able to remove.

If this bundle has exported any packages, the Framework must continue to make these packages available to their importing bundles until the `PackageAdmin.refreshPackages` method has been called or the Framework is relaunched.

The following steps are required to uninstall a bundle:

1. If this bundle's state is **UNINSTALLED** then an `IllegalStateException` is thrown.
2. If this bundle's state is **ACTIVE**, **STARTING** or **STOPPING**, this bundle is stopped as described in the `Bundle.stop` method. If `Bundle.stop` throws an exception, a Framework event of type `FrameworkEvent.ERROR` is fired containing the exception.
3. This bundle's state is set to **UNINSTALLED**.
4. A bundle event of type `BundleEvent.UNINSTALLED` is fired.
5. This bundle and any persistent storage area provided for this bundle by the Framework are removed.

**Preconditions**

- `getState()` not in `{UNINSTALLED}`.

**Postconditions, no exceptions thrown**

- `getState()` in `{UNINSTALLED}`.
- This bundle has been uninstalled.

**Postconditions, when an exception is thrown**

- `getState()` not in `{UNINSTALLED}`.
- This Bundle has not been uninstalled.

**Throws:**
- `BundleException` - If the uninstall failed. This can occur if another thread is attempting to change this bundle's state and does not complete in a timely manner.
- `IllegalStateException` - If this bundle has been uninstalled or this bundle tries to change its own state.
- `SecurityException` - If the caller does not have the appropriate `AdminPermission[this,LIFECYCLE]`, and the Java Runtime Environment supports permissions.
getHeaders

Dictionary<String,String> getHeaders()

Returns this bundle's Manifest headers and values. This method returns all the Manifest headers and values from the main section of this bundle's Manifest file; that is, all lines prior to the first blank line.

Manifest header names are case-insensitive. The methods of the returned Dictionary object must operate on header names in a case-insensitive manner. If a Manifest header value starts with "%", it must be localized according to the default locale. If no localization is found for a header value, the header value without the leading "%" is returned.

For example, the following Manifest headers and values are included if they are present in the Manifest file:

- Bundle-Name
- Bundle-Vendor
- Bundle-Version
- Bundle-Description
- Bundle-DocURL
- Bundle-ContactAddress

This method must continue to return Manifest header information while this bundle is in the UNINSTALLED state.

Returns:
A Dictionary object containing this bundle's Manifest headers and values.

Throws:
SecurityException - If the caller does not have the appropriate AdminPermission[this,METADATA], and the Java Runtime Environment supports permissions.

See Also:
Constants.BUNDLE_LOCALIZATION

getBundleId

long getBundleId()

Returns this bundle's unique identifier. This bundle is assigned a unique identifier by the Framework when it was installed in the OSGi environment.

A bundle's unique identifier has the following attributes:

- Is unique and persistent.
- Is a long.
- Its value is not reused for another bundle, even after a bundle is uninstalled.
- Does not change while a bundle remains installed.
- Does not change when a bundle is updated.

This method must continue to return this bundle's unique identifier while this bundle is in the UNINSTALLED state.

Returns:
The unique identifier of this bundle.

getLocation

String getLocation()
Returns this bundle's location identifier.

The location identifier is the location passed to BundleContext.installBundle when a bundle is installed. The location identifier does not change while this bundle remains installed, even if this bundle is updated.

This method must continue to return this bundle's location identifier while this bundle is in the UNINSTALLED state.

Returns: The string representation of this bundle's location identifier.
Throws: SecurityException - If the caller does not have the appropriate AdminPermission[this,METADATA], and the Java Runtime Environment supports permissions.

getRegisteredServices

ServiceReference<?>[] getRegisteredServices()

Returns this bundle's ServiceReference list for all services it has registered or null if this bundle has no registered services.

If the Java runtime supports permissions, a ServiceReference object to a service is included in the returned list only if the caller has the ServicePermission to get the service using at least one of the named classes the service was registered under.

The list is valid at the time of the call to this method, however, as the Framework is a very dynamic environment, services can be modified or unregistered at anytime.

Returns: An array of ServiceReference objects or null.
Throws: IllegalStateException - If this bundle has been uninstalled.
See Also: ServiceRegistration, ServiceReference, ServicePermission

getServicesInUse

ServiceReference<?>[] getServicesInUse()

Returns this bundle's ServiceReference list for all services it is using or returns null if this bundle is not using any services. A bundle is considered to be using a service if its use count for that service is greater than zero.

If the Java Runtime Environment supports permissions, a ServiceReference object to a service is included in the returned list only if the caller has the ServicePermission to get the service using at least one of the named classes the service was registered under.

The list is valid at the time of the call to this method, however, as the Framework is a very dynamic environment, services can be modified or unregistered at anytime.

Returns: An array of ServiceReference objects or null.
Throws: IllegalStateException - If this bundle has been uninstalled.
See Also: ServiceReference, ServicePermission
hasPermission

boolean hasPermission(Object permission)

Determines if this bundle has the specified permissions.

If the Java Runtime Environment does not support permissions, this method always returns true.

permission is of type Object to avoid referencing the java.security.Permission class directly. This is
to allow the Framework to be implemented in Java environments which do not support permissions.

If the Java Runtime Environment does support permissions, this bundle and all its resources including
embedded JAR files, belong to the same java.securityProtectionDomain; that is, they must share the
same set of permissions.

Parameters:
permission - The permission to verify.

Returns:
true if this bundle has the specified permission or the permissions possessed by this bundle imply
the specified permission; false if this bundle does not have the specified permission or
permission is not an instanceof java.security.Permission.

Throws:
IllegalStateException - If this bundle has been uninstalled.

getResource

URL getResource(String name)

Find the specified resource from this bundle's class loader. This bundle's class loader is called to search for
the specified resource. If this bundle's state is INSTALLED, this method must attempt to resolve this bundle
before attempting to get the specified resource. If this bundle cannot be resolved, then only this bundle
must be searched for the specified resource. Imported packages cannot be searched when this bundle has
not been resolved. If this bundle is a fragment bundle then null is returned.

Note: Jar and zip files are not required to include directory entries. URLs to directory entries will not be
returned if the bundle contents do not contain directory entries.

Parameters:
name - The name of the resource. See 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 appropriate AdminPermission[this,RESOURCE],
and the Java Runtime Environment supports permissions.

Throws:
IllegalStateException - If this bundle has been uninstalled.

Since: 1.1

See Also:
getEntry(), findEntries()

getHeaders

Dictionary<String,String> getHeaders(String locale)

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

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 must be localized according to the specified locale. If a locale is specified and cannot be found, then the header values must be returned using the default locale. Localizations are searched for in the following order:

bn + "_" + Ls + "_" + Cs + "_" + Vs
bn + "_" + Ls + "_" + Cs
bn + "_" + Ls
bn + "_" + Ld + "_" + Cd + "_" + Vd
bn + "_" + Ld + "_" + Cd
bn + "_" + Ld
bn

Where bn is this bundle's localization basename, Ls, Cs and Vs are the specified locale (language, country, variant) and Ld, Cd and Vd are the default locale (language, country, variant). If null is specified as the locale string, the header values must be localized using the default locale. If the empty string ("") is specified as the locale string, the header values must not be localized and the raw (unlocalized) header values, including any leading "%", must be returned. If no localization is found for a header value, the header value without the leading "%" is returned.

This method must continue to return Manifest header information while this bundle is in the UNINSTALLED state, however the header values must only be available in the raw and default locale values.

Parameters:
locale - The locale name into which the header values are to be localized. If the specified locale is null then the locale returned by java.util.Locale.getDefault is used. If the specified locale is the empty string, this method will return the raw (unlocalized) manifest headers including any leading "%".

Returns:
A Dictionary object containing this bundle's Manifest headers and values.

Throws:
SecurityException - If the caller does not have the appropriate AdminPermission[this,METADATA], and the Java Runtime Environment supports permissions.

Since: 1.3
See Also: getHeaders(), Constants.BUNDLE_LOCALIZATION

getSymbolicName

String getSymbolicName()

Returns the symbolic name of this bundle as specified by its Bundle-SymbolicName manifest header. The bundle symbolic name together with a version must identify a unique bundle. The bundle symbolic name should be based on the reverse domain name naming convention like that used for java packages.

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

Returns:
The symbolic name of this bundle or null if this bundle does not have a symbolic name.

Since: 1.3

loadClass

Class<?> loadClass(String name)
throws ClassNotFoundException

Loads the specified class using this bundle's class loader.

If this bundle is a fragment bundle then this method must throw a ClassNotFoundException.
If this bundle's state is **INSTALLED**, this method must attempt to resolve this bundle before attempting to load the class.

If this bundle cannot be resolved, a Framework event of type `FrameworkEvent.ERROR` is fired containing a `BundleException` with details of the reason this bundle could not be resolved. This method must then 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:**
- `ClassNotFoundException` - If no such class can be found or if this bundle is a fragment bundle or if the caller does not have the appropriate `AdminPermission[this,CLASS]`, and the Java Runtime Environment supports permissions.
- `IllegalStateException` - If this bundle has been uninstalled.

**Since:**
1.3

---

### getResources

```java
Enumeration<URL> getResource(String name)
throws IOException
```

Find the specified resources from this bundle's class loader. This bundle's class loader is called to search for the specified resources. If this bundle's state is **INSTALLED**, this method must attempt to resolve this bundle before attempting to get the specified resources. If this bundle cannot be resolved, then only this bundle must be searched for the specified resources. Imported packages cannot be searched when a bundle has not been resolved. If this bundle is a fragment bundle then null is returned.

Note: Jar and zip files are not required to include directory entries. URLs to directory entries will not be returned if the bundle contents do not contain directory entries.

**Parameters:**
- `name` - The name of the resource. See `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 appropriate `AdminPermission[this,RESOURCE]`, and the Java Runtime Environment supports permissions.

**Throws:**
- `IOException` - If there is an I/O error.
- `IllegalStateException` - If this bundle has been uninstalled.

**Since:**
1.3

---

### getEntryPaths

```java
Enumeration<String> getEntryPaths(String path)
```

Returns an Enumeration of all the paths (String objects) to entries within this bundle whose longest sub-path matches the specified path. This bundle's class loader is not used to search for entries. Only the contents of this bundle are searched.

The specified path is always relative to the root of this bundle and may begin with a "/". A path value of "/" indicates the root of this bundle.
Returned paths indicating subdirectory paths end with a "/". The returned paths are all relative to the root of this bundle and must not begin with "/".

Note: Jar and zip files are not required to include directory entries. Paths to directory entries will not be returned if the bundle contents do not contain directory entries.

**Parameters:**
- path - The path name for which to return entry paths.

**Returns:**
- An Enumeration of the entry paths (String objects) or null if no entry could be found or if the caller does not have the appropriate AdminPermission[this,RESOURCE] and the Java Runtime Environment supports permissions.

**Throws:**
- IllegalStateException - If this bundle has been uninstalled.

**Since:** 1.3

---

**getEntry**

**URL getEntry(String path)**

Returns a URL to the entry at the specified path in this bundle. This bundle's class loader is not used to search for the entry. Only the contents of this bundle are searched for the entry.

The specified path is always relative to the root of this bundle and may begin with "/". A path value of "/" indicates the root of this bundle.

Note: Jar and zip files are not required to include directory entries. URLs to directory entries will not be returned if the bundle contents do not contain directory entries.

**Parameters:**
- path - The path name of the entry.

**Returns:**
- A URL to the entry, or null if no entry could be found or if the caller does not have the appropriate AdminPermission[this,RESOURCE] and the Java Runtime Environment supports permissions.

**Throws:**
- IllegalStateException - If this bundle has been uninstalled.

**Since:** 1.3

---

**getLastModified**

**long getLastModified()**

Returns the time when this bundle was last modified. A bundle is considered to be modified when it is installed, updated or uninstalled.

The time value is the number of milliseconds since January 1, 1970, 00:00:00 GMT.

**Returns:**
- The time when this bundle was last modified.

**Since:** 1.3

---

**findEntries**

**Enumeration<URL> findEntries(String path,**
- **String filePattern,**
- **boolean recurse)**
Returns entries in this bundle and its attached fragments. This bundle's class loader is not used to search for entries. Only the contents of this bundle and its attached fragments are searched for the specified entries. If this bundle's state is INSTALLED, this method must attempt to resolve this bundle before attempting to find entries.

This method is intended to be used to obtain configuration, setup, localization and other information from this bundle. This method takes into account that the "contents" of this bundle can be extended with fragments. This "bundle space" is not a namespace with unique members; the same entry name can be present multiple times. This method therefore returns an enumeration of URL objects. These URLs can come from different JARs but have the same path name. This method can either return only entries in the specified path or recurse into subdirectories returning entries in the directory tree beginning at the specified path. Fragments can be attached after this bundle is resolved, possibly changing the set of URLs returned by this method. If this bundle is not resolved, only the entries in the JAR file of this bundle are returned.

Examples:

```java
// List all XML files in the OSGI-INF directory and below
Enumeration e = b.findEntries("OSGI-INF", ".*\.xml", true);

// Find a specific localization file
Enumeration e = b
   .findEntries("OSGI-INF/l10n", "bundle_nl_DU.properties", false);
if (e.hasMoreElements())
   return (URL) e.nextElement();
```

Note: Jar and zip files are not required to include directory entries. URLs to directory entries will not be returned if the bundle contents do not contain directory entries.

Parameters:
- **path** - The path name in which to look. The path is always relative to the root of this bundle and may begin with "/". A path value of "/" indicates the root of this bundle.
- **filePattern** - The file name pattern for selecting entries in the specified path. The pattern is only matched against the last element of the entry path. If the entry is a directory then the trailing "/" is not used for pattern matching. Substring matching is supported, as specified in the Filter specification, using the wildcard character ("*"). If null is specified, this is equivalent to "*" and matches all files.
- **recurse** - If true, recurse into subdirectories. Otherwise only return entries from the specified path.

Returns: An enumeration of URL objects for each matching entry, or null if an entry could not be found or if the caller does not have the appropriate AdminPermission[this,RESOURCE], and the Java Runtime Environment supports permissions. The URLs are sorted such that entries from this bundle are returned first followed by the entries from attached fragments in ascending bundle id order. If this bundle is a fragment, then only matching entries in this fragment are returned.

Throws: IllegalStateException - If this bundle has been uninstalled.

Since: 1.3

### getBundleContext

**BundleContext getBundleContext()**

Returns this bundle's BundleContext. The returned BundleContext can be used by the caller to act on behalf of this bundle.

If this bundle is not in the STARTING, ACTIVE, or STOPPING states or this bundle is a fragment bundle, then this bundle has no valid BundleContext. This method will return null if this bundle has no valid BundleContext.

Returns: A BundleContext for this bundle or null if this bundle has no valid BundleContext.
Throws:  
SecurityException - If the caller does not have the appropriate
AdminPermission[<this, CONTEXT>], and the Java Runtime Environment supports permissions.

Since:  
1.4

getSignerCertificates

Map<X509Certificate, List<X509Certificate>> getSignerCertificates(int signersType)

Return the certificates for the signers of this bundle and the certificate chains for those signers.

Parameters:  

signersType - If SIGNERS_ALL is specified, then information on all signers of this bundle is returned. If SIGNERS_TRUSTED is specified, then only information on the signers of this bundle trusted by the framework is returned.

Returns:  
The X509Certificates for the signers of this bundle and the X509Certificate chains for those signers. The keys of the Map are the X509Certificates of the signers of this bundle. The value for a key is a List containing the X509Certificate chain for the signer. The first item in the List is the signer's X509Certificate which is then followed by the rest of the X509Certificate chain. The returned Map will be empty if there are no signers. The returned Map is the property of the caller who is free to modify it.

Throws:  
IllegalArgumentException - If the specified signersType is not SIGNERS_ALL or SIGNERS_TRUSTED.

Since:  
1.5

getVersion

Version getVersion()

Returns the version of this bundle as specified by its Bundle-Version manifest header. If this bundle does not have a specified version then Version.emptyVersion is returned.

This method must continue to return this bundle's version while this bundle is in the UNINSTALLED state.

Returns:  
The version of this bundle.

Since:  
1.5

adapt

A adapt(Class<A> type)

Adapt a bundle to the specified type.

Adapting a bundle to the specified type may require certain checks, including security checks, to succeed. If a check does not succeed, then the bundle cannot be adapted and null is returned.

Type Parameters:  

A - The type to which the bundle is to be adapted.

Parameters:  

type - Class object for the type to which the bundle is to be adapted.
Returns:
The object, of the specified type, to which the bundle has been adapted or null if the bundle cannot be adapted to the specified type.

Since:
1.6

getTypes

int getTypes()

Returns the special types of this bundle. The bundle type values are:

- TYPE_FRAGMENT

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 this bundle is not one or more of the defined types then 0 is returned.

Returns:
The special types of this bundle. The type values are ORed together.

Since:
1.6
public interface BundleActivator

Customizes the starting and stopping of a bundle.

BundleActivator is an interface that may be implemented when a bundle is started or stopped. The Framework can create instances of a bundle's BundleActivator as required. If an instance's BundleActivator.start method executes successfully, it is guaranteed that the same instance's BundleActivator.stop method will be called when the bundle is to be stopped. The Framework must not concurrently call a BundleActivator object.

BundleActivator is specified through the Bundle-Activator Manifest header. A bundle can only specify a single BundleActivator in the Manifest file. Fragment bundles must not have a BundleActivator. The form of the Manifest header is:

    Bundle-Activator: class-name

where class-name is a fully qualified Java classname.

The specified BundleActivator class must have a public constructor that takes no parameters so that a BundleActivator object can be created by Class.newInstance().

Version: $Revision: 6361 $
NotThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void start(BundleContext context)</td>
<td>37</td>
</tr>
<tr>
<td>void stop(BundleContext context)</td>
<td>38</td>
</tr>
</tbody>
</table>

**Method Detail**

**start**

void start(BundleContext context)

Called when this bundle is started so the Framework can perform the bundle-specific activities necessary to start this bundle.

This method must complete and return to its caller in a timely manner.

**Parameters:**

context - The execution context of the bundle being started.

**Throws:**

Exception - If this method throws an exception, this bundle is marked as stopped and the Framework will remove this bundle's listeners, unregister all services registered by this bundle, and release all services used by this bundle.
stop

void stop(BundleContext context) throws Exception

Called when this bundle is stopped so the Framework can perform the bundle-specific activities necessary to stop the bundle. In general, this method should undo the work that the BundleActivator.start method started. There should be no active threads that were started by this bundle when this bundle returns. A stopped bundle must not call any Framework objects.

This method must complete and return to its caller in a timely manner.

Parameters:
context - The execution context of the bundle being stopped.

Throws:
Exception - If this method throws an exception, the bundle is still marked as stopped, and the Framework will remove the bundle's listeners, unregister all services registered by the bundle, and release all services used by the bundle.
public interface BundleContext
extends BundleReference

A bundle's execution context within the Framework. The context is used to grant access to other methods so that this bundle can interact with the Framework.

BundleContext methods allow a bundle to:

- Subscribe to events published by the Framework.
- Register service objects with the Framework service registry.
- Retrieve ServiceReferences from the Framework service registry.
- Get and release service objects for a referenced service.
- Install new bundles in the Framework.
- Get the list of bundles installed in the Framework.
- Get the Bundle object for a bundle.
- Create File objects for files in a persistent storage area provided for the bundle by the Framework.

A BundleContext object will be created and provided to the bundle associated with this context when it is started using the BundleActivator.start() method. The same BundleContext object will be passed to the bundle associated with this context when it is stopped using the BundleActivator.stop() method. A BundleContext object is generally for the private use of its associated bundle and is not meant to be shared with other bundles in the OSGi environment.

The Bundle object associated with a BundleContext object is called the context bundle.

The BundleContext object is only valid during the execution of its context bundle; that is, during the period from when the context bundle is in the STARTING, STOPPING, and ACTIVE bundle states. If the BundleContext object is used subsequently, an IllegalStateException must be thrown. The BundleContext object must never be reused after its context bundle is stopped.

The Framework is the only entity that can create BundleContext objects and they are only valid within the Framework that created them.

A Bundle can be adapted to its BundleContext. In order for this to succeed, the caller must have the appropriate AdminPermission[bundle,CONTEXT] if the Java Runtime Environment supports permissions.

Version:
$Revision: 8569 $
ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addBundleListener(BundleListener listener)</td>
<td>44</td>
</tr>
<tr>
<td>Adds the specified BundleListener object to the context bundle's list of listeners if not already present.</td>
<td></td>
</tr>
<tr>
<td>void addFrameworkListener(FrameworkListener listener)</td>
<td>45</td>
</tr>
<tr>
<td>Adds the specified FrameworkListener object to the context bundle's list of listeners if not already present.</td>
<td></td>
</tr>
<tr>
<td>void addServiceListener(ServiceListener listener)</td>
<td>43</td>
</tr>
<tr>
<td>Adds the specified ServiceListener object to the context bundle's list of listeners.</td>
<td></td>
</tr>
</tbody>
</table>
void **addServiceListener** (**ServiceListener** listener, String filter)**

Adds the specified **ServiceListener** object with the specified filter to the context bundle's list of listeners.

**void** **createFilter**(String filter)**

Creates a **filter** object.

**Filter** **getServiceReference**(String clazz, String filter)**

Returns an array of **ServiceReference** objects.

**String** **getString**(String key)**

Returns the value of the specified property.

**ServiceRegistration**<**S**> **registerService**(Class<**S**> clazz, **S** service, Dictionary<String, Object> properties)**

Registers the specified service object with the specified properties under the specified class name with the Framework.

**ServiceRegistration**<**?> registerService**(String clazz, **Object** service, Dictionary<String, Object> properties)**

Registers the specified service object with the specified properties under the specified class name with the Framework.

**ServiceRegistration**<**?> registerService**(String[] clazzes, **Object** service, Dictionary<String, Object> properties)**

Registers the specified service object with the specified properties under the specified class names into the Framework.

**void** **removeBundleListener**(**BundleListener** listener)**

Removes the specified **BundleListener** object from the context bundle's list of listeners.

**void** **removeFrameworkListener**(**FrameworkListener** listener)**

Removes the specified **FrameworkListener** object from the context bundle's list of listeners.

**void** **removeServiceListener**(**ServiceListener** listener)**

Removes the specified **ServiceListener** object from the context bundle's list of listeners.
Interface BundleContext

<table>
<thead>
<tr>
<th>boolean</th>
<th>ungetService(ServiceReference&lt;?&gt; reference)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Releases the service object referenced by the specified ServiceReference object.</td>
</tr>
</tbody>
</table>

## Method Detail

### ungetService

```java
public boolean ungetService(ServiceReference<?> reference)
```

Releases the service object referenced by the specified ServiceReference object.

### getProperty

```java
public String getProperty(String key)
```

Returns the value of the specified property. If the key is not found in the Framework properties, the system properties are then searched. The method returns null if the property is not found.

All bundles must have permission to read properties whose names start with "org.osgi."

**Parameters:**

- `key`: The name of the requested property.

**Returns:**

- The value of the requested property, or null if the property is undefined.

**Throws:**

- `SecurityException` - If the caller does not have the appropriate PropertyPermission to read the property, and the Java Runtime Environment supports permissions.

### getBundle

```java
public Bundle getBundle()
```

Returns the Bundle object associated with this BundleContext. This bundle is called the context bundle.

**Specified by:**

getBundle in interface BundleReference

**Returns:**

- The Bundle object associated with this BundleContext.

**Throws:**

- `IllegalStateException` - If this BundleContext is no longer valid.

### installBundle

```java
public Bundle installBundle(String location, InputStream input)
```

Installs a bundle from the specified InputStream object.

If the specified InputStream is null, the Framework must create the InputStream from which to read the bundle by interpreting, in an implementation dependent manner, the specified location.

The specified location identifier will be used as the identity of the bundle. Every installed bundle is uniquely identified by its location identifier which is typically in the form of a URL.

The following steps are required to install a bundle:

1. If a bundle containing the same location identifier is already installed, the Bundle object for that bundle is returned.
2. The bundle's content is read from the input stream. If this fails, a BundleException is thrown.
3. The bundle's associated resources are allocated. The associated resources minimally consist of a unique identifier and a persistent storage area if the platform has file system support. If this step fails, a BundleException is thrown.
4. The bundle's state is set to INSTALLED.
5. A bundle event of type BundleEvent.INSTALLED is fired.
6. The Bundle object for the newly or previously installed bundle is returned.

Postconditions, no exceptions thrown

- `getState()` in {INSTALLED, RESOLVED}.
- Bundle has a unique ID.

Postconditions, when an exception is thrown

- Bundle is not installed and no trace of the bundle exists.

Parameters:
- `location` - The location identifier of the bundle to install.
- `input` - The `InputStream` object from which this bundle will be read or null to indicate the Framework must create the input stream from the specified location identifier. The input stream must always be closed when this method completes, even if an exception is thrown.

Returns:
The Bundle object of the installed bundle.

Throws:
- `BundleException` - If the input stream cannot be read or the installation failed.
- `SecurityException` - If the caller does not have the appropriate `AdminPermission[installed bundle,LIFECYCLE]`, and the Java Runtime Environment supports permissions.
- `IllegalStateException` - If this BundleContext is no longer valid.

installBundle

```java
Bundle installBundle(String location)
```

Installs a bundle from the specified `location` identifier.

This method performs the same function as calling `installBundle(String, InputStream)` with the specified `location` identifier and a null `InputStream`.

Parameters:
- `location` - The location identifier of the bundle to install.

Returns:
The Bundle object of the installed bundle.

Throws:
- `BundleException` - If the installation failed.
- `SecurityException` - If the caller does not have the appropriate `AdminPermission[installed bundle,LIFECYCLE]`, and the Java Runtime Environment supports permissions.
- `IllegalStateException` - If this BundleContext is no longer valid.

See Also:
- `installBundle(String, InputStream)`

getBundle

```java
Bundle getBundle(long id)
```

Returns the bundle with the specified identifier.

Parameters:
- `id` - The identifier of the bundle to retrieve.

Returns:
- A Bundle object or null if the identifier does not match any installed bundle.
Interface BundleContext

getBundles

```java
Bundle[] getBundles()
```

Returns a list of all installed bundles.

This method returns a list of all bundles installed in the OSGi environment at the time of the call to this method. However, since the Framework is a very dynamic environment, bundles can be installed or uninstalled at anytime.

Returns:
An array of Bundle objects, one object per installed bundle.

addServiceListener

```java
void addServiceListener(ServiceListener listener,
                        String filter)
```

Adds the specified ServiceListener object with the specified filter to the context bundle's list of listeners. See Filter for a description of the filter syntax. ServiceListener objects are notified when a service has a lifecycle state change.

If the context bundle's list of listeners already contains a listener \( l \) such that \( l \equiv \text{listener} \), then this method replaces that listener's filter (which may be null) with the specified one (which may be null).

The listener is called if the filter criteria is met. To filter based upon the class of the service, the filter should reference the Constants.OBJECTCLASS property. If filter is null, all services are considered to match the filter.

When using a filter, it is possible that the ServiceEvents for the complete lifecycle of a service will not be delivered to the listener. For example, if the filter only matches when the property \( x \) has the value 1, the listener will not be called if the service is registered with the property \( x \) not set to the value 1. Subsequently, when the service is modified setting property \( x \) to the value 1, the filter will match and the listener will be called with a ServiceEvent of type MODIFIED. Thus, the listener will not be called with a ServiceEvent of type REGISTERED.

If the Java Runtime Environment supports permissions, the ServiceListener object will be notified of a service event only if the bundle that is registering it has the ServicePermission to get the service using at least one of the named classes the service was registered under.

Parameters:
- listener - The ServiceListener object to be added.
- filter - The filter criteria.

Throws:
- IllegalArgumentException - If filter contains an invalid filter string that cannot be parsed.
- IllegalStateException - If this BundleContext is no longer valid.

See Also:
- ServiceEvent, ServiceListener, ServicePermission

addServiceListener

```java
void addServiceListener(ServiceListener listener)
```

Adds the specified ServiceListener object to the context bundle's list of listeners.

This method is the same as calling BundleContext.addServiceListener(ServiceListener listener, String filter) with filter set to null.
**Interface BundleContext**

```java
Parameters:
listener - The ServiceListener object to be added.

Throws:
IllegalStateException - If this BundleContext is no longer valid.

See Also:
addServiceListener(ServiceListener, String)
```

**removeServiceListener**

```java
void removeServiceListener(ServiceListener listener)

Removes the specified ServiceListener object from the context bundle's list of listeners.

If `listener` is not contained in this context bundle's list of listeners, this method does nothing.

Parameters:
listener - The ServiceListener to be removed.

Throws:
IllegalStateException - If this BundleContext is no longer valid.
```

**addBundleListener**

```java
void addBundleListener(BundleListener listener)

Adds the specified BundleListener object to the context bundle's list of listeners if not already present. BundleListener objects are notified when a bundle has a lifecycle state change.

If the context bundle's list of listeners already contains a listener `l` such that `(l==listener)`, this method does nothing.

Parameters:
listener - The BundleListener to be added.

Throws:
IllegalStateException - If this BundleContext is no longer valid.
SecurityException - If listener is a SynchronousBundleListener and the caller does not have the appropriate AdminPermission[context bundle,LISTENER], and the Java Runtime Environment supports permissions.

See Also:
BundleEvent, BundleListener
```

**removeBundleListener**

```java
void removeBundleListener(BundleListener listener)

Removes the specified BundleListener object from the context bundle's list of listeners.

If `listener` is not contained in the context bundle's list of listeners, this method does nothing.

Parameters:
listener - The BundleListener object to be removed.

Throws:
IllegalStateException - If this BundleContext is no longer valid.
SecurityException - If listener is a SynchronousBundleListener and the caller does not have the appropriate AdminPermission[context bundle,LISTENER], and the Java Runtime Environment supports permissions.
```
Interface BundleContext

### addFrameworkListener

```java
void addFrameworkListener(FrameworkListener listener)
```

Adds the specified FrameworkListener object to the context bundle's list of listeners if not already present. FrameworkListeners are notified of general Framework events.

If the context bundle's list of listeners already contains a listener \(l\) such that \(l==\text{listener}\), this method does nothing.

**Parameters:**
- `listener` - The FrameworkListener object to be added.

**Throws:**
- `IllegalStateException` - If this BundleContext is no longer valid.

**See Also:**
- `FrameworkEvent`, `FrameworkListener`

### removeFrameworkListener

```java
void removeFrameworkListener(FrameworkListener listener)
```

Removes the specified FrameworkListener object from the context bundle's list of listeners.

If `listener` is not contained in the context bundle's list of listeners, this method does nothing.

**Parameters:**
- `listener` - The FrameworkListener object to be removed.

**Throws:**
- `IllegalStateException` - If this BundleContext is no longer valid.

### registerService

```java
ServiceRegistration<?> registerService(String[] clazzes, Object service, Dictionary<String, Object> properties)
```

Registers the specified service object with the specified properties under the specified class names into the Framework. A `ServiceRegistration` object is returned. The `ServiceRegistration` object is for the private use of the bundle registering the service and should not be shared with other bundles. The registering bundle is defined to be the context bundle. Other bundles can locate the service by using either the `getServiceReferences()` or `getServiceReference()` method.

A bundle can register a service object that implements the `ServiceFactory` interface to have more flexibility in providing service objects to other bundles.

The following steps are required to register a service:

1. If `service` is not a `ServiceFactory`, an `IllegalArgumentException` is thrown if `service` is not an instance of all the specified class names.
2. The Framework adds the following service properties to the service properties from the specified `Dictionary` (which may be null):
   - A property named `Constants.SERVICE_ID` identifying the registration number of the service
   - A property named `Constants.OBJECTCLASS` containing all the specified classes.
   - Properties with these names in the specified `Dictionary` will be ignored.
3. The service is added to the Framework service registry and may now be used by other bundles.
4. A service event of type `ServiceEvent.REGISTERED` is fired.
5. A `ServiceRegistration` object for this registration is returned.
Parameters:
- clazzes - The class names under which the service can be located. The class names in this array will be stored in the service's properties under the key Constants.OBJECTCLASS.
- service - The service object or a ServiceFactory object.
- properties - The properties for this service. The keys in the properties object must all be String objects. See Constants for a list of standard service property keys. Changes should not be made to this object after calling this method. To update the service's properties the ServiceRegistration.setProperties() method must be called. The set of properties may be null if the service has no properties.

Returns:
A ServiceRegistration object for use by the bundle registering the service to update the service's properties or to unregister the service.

Throws:
- IllegalArgumentException - If one of the following is true:
  - service is null.
  - service is not a ServiceFactory object and is not an instance of all the named classes in clazzes.
  - properties contains case variants of the same key name.
- SecurityException - If the caller does not have the ServicePermission to register the service for all the named classes and the Java Runtime Environment supports permissions.
- IllegalStateException - If this BundleContext is no longer valid.

See Also:
ServiceRegistration, ServiceFactory

registerService

ServiceRegistration<? registerService(String clazz,
Object service,
Dictionary<String,Object> properties)

Registers the specified service object with the specified properties under the specified class name with the Framework.

This method is otherwise identical to registerService(String[], Object, Dictionary) and is provided as a convenience when service will only be registered under a single class name. Note that even in this case the value of the service's Constants.OBJECTCLASS property will be an array of string, rather than just a single string.

Parameters:
- clazz - The class name under which the service can be located.
- service - The service object or a ServiceFactory object.
- properties - The properties for this service.

Returns:
A ServiceRegistration object for use by the bundle registering the service to update the service's properties or to unregister the service.

Throws:
IllegalStateException - If this BundleContext is no longer valid.

See Also:
registerService(String[], Object, Dictionary)

registerService

ServiceRegistration<S registerService(Class<S> clazz,
S service,
Dictionary<String, Object> properties)

Registers the specified service object with the specified properties under the specified class name with the Framework.
This method is otherwise identical to registerService(String[], Object, Dictionary) and is provided as a convenience when service will only be registered under a single class name. Note that even in this case the value of the service's Constants.OBJECTCLASS property will be an array of string, rather than just a single string.

Type Parameters:
- S - Type of Service.

Parameters:
- clazz - The class name under which the service can be located.
- service - The service object or a ServiceFactory object.
- properties - The properties for this service.

Returns:
- A ServiceRegistration object for use by the bundle registering the service to update the service’s properties or to unregister the service.

Throws:
- IllegalStateException - If this BundleContext is no longer valid.

Since: 1.6

See Also:
- registerService(String[], Object, Dictionary)

getServiceReferences

ServiceReference<?>[] getServiceReferences(String clazz,
                                        String filter)
            throws InvalidSyntaxException

Returns an array of ServiceReference objects. The returned array of ServiceReference objects contains services that were registered under the specified class, match the specified filter expression, and the packages for the class names under which the services were registered match the context bundle's packages as defined in ServiceReference.isAssignableTo(Bundle, String).

The list is valid at the time of the call to this method. However since the Framework is a very dynamic environment, services can be modified or unregistered at any time.

The specified filter expression is used to select the registered services whose service properties contain keys and values which satisfy the filter expression. See Filter for a description of the filter syntax. If the specified filter is null, all registered services are considered to match the filter. If the specified filter expression cannot be parsed, an InvalidSyntaxException will be thrown with a human readable message where the filter became unparsable.

The result is an array of ServiceReference objects for all services that meet all of the following conditions:

- If the specified class name, clazz, is not null, the service must have been registered with the specified class name. The complete list of class names with which a service was registered is available from the service's objectClass property.
- If the specified filter is not null, the filter expression must match the service.
- If the Java Runtime Environment supports permissions, the caller must have ServicePermission with the GET action for at least one of the class names under which the service was registered.
- For each class name with which the service was registered, calling ServiceReference.isAssignableTo(Bundle, String) with the context bundle and the class name on the service’s ServiceReference object must return true.

Parameters:
- clazz - The class name with which the service was registered or null for all services.
- filter - The filter expression or null for all services.

Returns:
- An array of ServiceReference objects or null if no services are registered which satisfy the search.
getAllServiceReferences

ServiceReference<[]> getAllServiceReferences(String clazz,
String filter)
throws InvalidSyntaxException

Returns an array of ServiceReference objects. The returned array of ServiceReference objects contains services that were registered under the specified class and match the specified filter expression.

The list is valid at the time of the call to this method. However since the Framework is a very dynamic environment, services can be modified or unregistered at any time.

The specified filter expression is used to select the registered services whose service properties contain keys and values which satisfy the filter expression. See Filter for a description of the filter syntax. If the specified filter is null, all registered services are considered to match the filter. If the specified filter expression cannot be parsed, an InvalidSyntaxException will be thrown with a human readable message where the filter became unparsable.

The result is an array of ServiceReference objects for all services that meet all of the following conditions:

- If the specified class name, clazz, is not null, the service must have been registered with the specified class name. The complete list of class names with which a service was registered is available from the service’s objectClass property.
- If the specified filter is not null, the filter expression must match the service.
- If the Java Runtime Environment supports permissions, the caller must have ServicePermission with the GET action for at least one of the class names under which the service was registered.

Parameters:
clazz - The class name with which the service was registered or null for all services.
filter - The filter expression or null for all services.

Returns:
An array of ServiceReference objects or null if no services are registered which satisfy the search.

Throws:
InvalidSyntaxException - If the specified filter contains an invalid filter expression that cannot be parsed.
IllegalStateException - If this BundleContext is no longer valid.

Since: 1.3

ggetServiceReference

ServiceReference<?> getServiceReference(String clazz)

Returns a ServiceReference object for a service that implements and was registered under the specified class.

The returned ServiceReference object is valid at the time of the call to this method. However as the Framework is a very dynamic environment, services can be modified or unregistered at any time.

This method is the same as calling getServiceReferences(String, String) with a null filter expression and then finding the reference with the highest priority. It is provided as a convenience for when the caller is interested in any service that implements the specified class.
If multiple such services exist, the service with the highest priority is selected. This priority is defined as the service reference with the highest ranking (as specified in its `Constants.SERVICE_RANKING` property) is returned.

If there is a tie in ranking, the service with the lowest service ID (as specified in its `Constants.SERVICE_ID` property); that is, the service that was registered first is returned.

**Parameters:**
- `clazz` - The class name with which the service was registered.

**Returns:**
- A `ServiceReference` object, or `null` if no services are registered which implement the named class.

**Throws:**
- `IllegalStateException` - If this BundleContext is no longer valid.

**See Also:**
- `getServiceReferences(String, String)`

---

### getServiceImplReference

```java
ServiceReference<S> getServiceReference(Class<S> clazz)
```

Returns a `ServiceReference` object for a service that implements and was registered under the specified class.

The returned `ServiceReference` object is valid at the time of the call to this method. However as the Framework is a very dynamic environment, services can be modified or unregistered at any time.

This method is the same as calling `getServiceReferences(Class, String)` with a null filter expression. It is provided as a convenience for when the caller is interested in any service that implements the specified class.

If multiple such services exist, the service with the highest ranking (as specified in its `Constants.SERVICE_RANKING` property) is returned.

If there is a tie in ranking, the service with the lowest service ID (as specified in its `Constants.SERVICE_ID` property); that is, the service that was registered first is returned.

**Type Parameters:**
- `S` - Type of Service.

**Parameters:**
- `clazz` - The class name with which the service was registered.

**Returns:**
- A `ServiceReference` object, or `null` if no services are registered which implement the named class.

**Throws:**
- `IllegalStateException` - If this BundleContext is no longer valid.

**Since:**
- 1.6

**See Also:**
- `getServiceReferences(Class, String)`

---

### getServiceReferences

```java
Collection<ServiceReference<S>> getServiceReferences(Class<S> clazz, String filter)
```

Returns a collection of `ServiceReference` objects. The returned collection of `ServiceReference` objects contains services that were registered under the specified class, match the specified filter expression, and

**Throws:**
- `InvalidSyntaxException`
the packages for the class names under which the services were registered match the context bundle's packages as defined in `ServiceReference.isAssignableTo(Bundle, String)`.

The collection is valid at the time of the call to this method. However since the Framework is a very dynamic environment, services can be modified or unregistered at any time.

The specified filter expression is used to select the registered services whose service properties contain keys and values which satisfy the filter expression. See Filter for a description of the filter syntax. If the specified filter is null, all registered services are considered to match the filter. If the specified filter expression cannot be parsed, an InvalidSyntaxException will be thrown with a human readable message where the filter became unparsable.

The result is a collection of ServiceReference objects for all services that meet all of the following conditions:

- If the specified class name, clazz, is not null, the service must have been registered with the specified class name. The complete list of class names with which a service was registered is available from the service's `objectClass` property.
- If the specified filter is not null, the filter expression must match the service.
- If the Java Runtime Environment supports permissions, the caller must have ServicePermission with the GET action for at least one of the class names under which the service was registered.
- For each class name with which the service was registered, calling `ServiceReference.isAssignableTo(Bundle, String)` with the context bundle and the class name on the service's `ServiceReference` object must return true.

Type Parameters:
- `S` - Type of Service

Parameters:
- clazz - The class name with which the service was registered. Must not be null.
- filter - The filter expression or null for all services.

Returns:
- A collection of ServiceReference objects. May be empty if no services are registered which satisfy the search.

Throws:
- `InvalidSyntaxException` - If the specified filter contains an invalid filter expression that cannot be parsed.
- `IllegalStateException` - If this BundleContext is no longer valid.

Since:
- 1.6

**getService**

```java
S getService(ServiceReference<S> reference)
```

Returns the service object referenced by the specified ServiceReference object.

A bundle's use of a service is tracked by the bundle's use count of that service. Each time a service's service object is returned by `getService(ServiceReference)` the context bundle's use count for that service is incremented by one. Each time the service is released by `ungetService(ServiceReference)` the context bundle's use count for that service is decremented by one.

When a bundle's use count for a service drops to zero, the bundle should no longer use that service.

This method will always return `null` when the service associated with this reference has been unregistered.

The following steps are required to get the service object:

1. If the service has been unregistered, `null` is returned.
2. The context bundle's use count for this service is incremented by one.
3. If the context bundle's use count for the service is currently one and the service was registered with an object implementing the `ServiceFactory` interface, the `ServiceFactory.getService(Bundle, ServiceRegistration)` method is called to create a service object for the context bundle. This service object is cached by the Framework. While the context bundle's use count for the service is greater than zero, subsequent calls to get the services's service object for the context bundle will return the cached service object. If the service object returned by the `ServiceFactory` object is not an `instanceof` all the classes named when the service was registered or the `ServiceFactory` object throws an exception, null is returned and a Framework event of type `FrameworkEvent.ERROR` containing a `ServiceException` describing the error is fired.

4. The service object for the service is returned.

Type Parameters:

s - Type of Service.

Parameters:

- reference - A reference to the service.

Returns:

A service object for the service associated with `reference` or null if the service is not registered, the service object returned by a `ServiceFactory` does not implement the classes under which it was registered or the `ServiceFactory` threw an exception.

Throws:

- `SecurityException` - If the caller does not have the `ServicePermission` to get the service using at least one of the named classes the service was registered under and the Java Runtime Environment supports permissions.
- `IllegalStateException` - If this BundleContext is no longer valid.
- `IllegalArgumentException` - If the specified `ServiceReference` was not created by the same framework instance as this BundleContext.

See Also:

`ungetService(ServiceReference)`, `ServiceFactory`

### ungetService

```java
boolean ungetService(<serviceReference> reference)
```

Releases the service object referenced by the specified `ServiceReference` object. If the context bundle's use count for the service is zero, this method returns `false`. Otherwise, the context bundle's use count for the service is decremented by one.

The service's service object should no longer be used and all references to it should be destroyed when a bundle's use count for the service drops to zero.

The following steps are required to unget the service object:

1. If the context bundle's use count for the service is zero or the service has been unregistered, `false` is returned.
2. The context bundle's use count for this service is decremented by one.
3. If the context bundle's use count for the service is currently zero and the service was registered with a `ServiceFactory` object, the `ServiceFactory.ungetService(Bundle, ServiceRegistration, Object)` method is called to release the service object for the context bundle.
4. `true` is returned.

Parameters:

- reference - A reference to the service to be released.

Returns:

- `false` if the context bundle's use count for the service is zero or if the service has been unregistered; `true` otherwise.

Throws:

- `IllegalStateException` - If this BundleContext is no longer valid.
- `IllegalArgumentException` - If the specified `ServiceReference` was not created by the same framework instance as this BundleContext.
Interface BundleContext

See Also:
getService(), ServiceFactory

dataFile

File getDataFile(String filename)

Creates a File object for a file in the persistent storage area provided for the bundle by the Framework. This method will return null if the platform does not have file system support.

A File object for the base directory of the persistent storage area provided for the context bundle by the Framework can be obtained by calling this method with an empty string as filename.

If the Java Runtime Environment supports permissions, the Framework will ensure that the bundle has the java.io.FilePermission with actions read,write,delete for all files (recursively) in the persistent storage area provided for the context bundle.

Parameters:
filename - A relative name to the file to be accessed.

Returns:
A File object that represents the requested file or null if the platform does not have file system support.

Throws:
IllegalStateException - If this BundleContext is no longer valid.

createFilter

Filter createFilter(String filter)
throws InvalidSyntaxException

Creates a Filter object. This Filter object may be used to match a ServiceReference object or a Dictionary object.

If the filter cannot be parsed, an InvalidSyntaxException will be thrown with a human readable message where the filter became unparsable.

Parameters:
filter - The filter string.

Returns:
A Filter object encapsulating the filter string.

Throws:
InvalidSyntaxException - If filter contains an invalid filter string that cannot be parsed.
NullPointerException - If filter is null.
IllegalStateException - If this BundleContext is no longer valid.

Since:
1.1

See Also:
"Framework specification for a description of the filter string syntax."
FrameworkUtil.createFilter(String)
Class BundleEvent

org.osgi.framework

java.lang.Object
  java.util.EventObject
    org.osgi.framework.BundleEvent

All Implemented Interfaces:
  Serializable

public class BundleEvent
  extends EventObject

An event from the Framework describing a bundle lifecycle change.

BundleEvent objects are delivered to SynchronousBundleListener and BundleListener when a change occurs in a bundle's lifecycle. A type code is used to identify the event type for future extendability.

OSGi Alliance reserves the right to extend the set of types.

Version:
$Revision: 6542 $

See Also:
  BundleListener, SynchronousBundleListener

Immutable

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int INSTALLED</td>
<td>The bundle has been installed.</td>
<td>54</td>
</tr>
<tr>
<td>static int LAZY_ACTIVATION</td>
<td>The bundle will be lazily activated.</td>
<td>56</td>
</tr>
<tr>
<td>static int RESOLVED</td>
<td>The bundle has been resolved.</td>
<td>55</td>
</tr>
<tr>
<td>static int STARTED</td>
<td>The bundle has been started.</td>
<td>54</td>
</tr>
<tr>
<td>static int STARTING</td>
<td>The bundle is about to be activated.</td>
<td>55</td>
</tr>
<tr>
<td>static int STOPPED</td>
<td>The bundle has been stopped.</td>
<td>54</td>
</tr>
<tr>
<td>static int STOPPING</td>
<td>The bundle is about to deactivated.</td>
<td>55</td>
</tr>
<tr>
<td>static int UNINSTALLED</td>
<td>The bundle has been uninstalled.</td>
<td>55</td>
</tr>
<tr>
<td>static int UNRESOLVED</td>
<td>The bundle has been unresolved.</td>
<td>55</td>
</tr>
<tr>
<td>static int UPDATED</td>
<td>The bundle has been updated.</td>
<td>54</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleEvent(int type, Bundle bundle)</td>
<td>Creates a bundle event of the specified type.</td>
<td>56</td>
</tr>
</tbody>
</table>
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Bundle getBundle()</code></td>
<td>Returns the bundle which had a lifecycle change.</td>
</tr>
<tr>
<td><code>int getType()</code></td>
<td>Returns the type of lifecycle event.</td>
</tr>
</tbody>
</table>

### Field Detail

**INSTALLED**

```java
public static final int INSTALLED = 1
```

The bundle has been installed.

See Also: [BundleContext.installBundle(String)](

**STARTED**

```java
public static final int STARTED = 2
```

The bundle has been started.

The bundle's [BundleActivator start](

See Also: [Bundle.start()](

**STOPPED**

```java
public static final int STOPPED = 4
```

The bundle has been stopped.

The bundle's [BundleActivator stop](

See Also: [Bundle.stop()](

**UPDATED**

```java
public static final int UPDATED = 8
```

The bundle has been updated.

See Also: [Bundle.update()](

---

*OSGi Javadoc -- 2/25/10*
UNINSTALLED

public static final int UNINSTALLED = 16

The bundle has been uninstalled.

See Also:
Bundle.uninstall()
Class BundleEvent

Since: 1.3
See Also: Bundle.stop()

LAZY_ACTIVATION

public static final int LAZY_ACTIVATION = 512

The bundle will be lazily activated.

The bundle has a lazy activation policy and is waiting to be activated. It is now in the STARTING state and has a valid BundleContext. This event is only delivered to SynchronousBundleListener. It is not delivered to BundleListener.

Since: 1.4

Constructor Detail

BundleEvent

public BundleEvent(int type, Bundle bundle)

Creates a bundle event of the specified type.

Parameters:
- type - The event type.
- bundle - The bundle which had a lifecycle change.

Method Detail

getBundle

public Bundle getBundle()

Returns the bundle which had a lifecycle change. This bundle is the source of the event.

Returns:
- The bundle that had a change occur in its lifecycle.

getType

public int getType()

Returns the type of lifecycle event. The type values are:

- INSTALLED
- RESOLVED
- LAZY_ACTIVATION
- STARTING
- STARTED
- STOPPING
- STOPPED
- UPDATED
- UNRESOLVED
- **UNINSTALLED**

**Returns:**

The type of lifecycle event.
Class BundleException

org.osgi.framework

java.lang.Object
  | java.lang.Throwable
  | java.lang.Exception
  | org.osgi.framework.BundleException

All Implemented Interfaces:
  Serializable

public class BundleException
extends Exception

A Framework exception used to indicate that a bundle lifecycle problem occurred.

A BundleException object is created by the Framework to denote an exception condition in the lifecycle of a bundle. BundleExceptions should not be created by bundle developers. A type code is used to identify the exception type for future extendability.

OSGi Alliance reserves the right to extend the set of types.

This exception conforms to the general purpose exception chaining mechanism.

Version:
  $Revision: 8271$

Field Summary

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int ACTIVATOR_ERROR</td>
<td>60</td>
</tr>
<tr>
<td>The bundle activator was in error.</td>
<td></td>
</tr>
<tr>
<td>static int DUPLICATE_BUNDLE_ERROR</td>
<td>61</td>
</tr>
<tr>
<td>The install or update operation failed because another already installed bundle has the same symbolic name and version.</td>
<td></td>
</tr>
<tr>
<td>static int INVALID_OPERATION</td>
<td>59</td>
</tr>
<tr>
<td>The operation was invalid.</td>
<td></td>
</tr>
<tr>
<td>static int MANIFEST_ERROR</td>
<td>60</td>
</tr>
<tr>
<td>The bundle manifest was in error.</td>
<td></td>
</tr>
<tr>
<td>static int NATIVECODE_ERROR</td>
<td>60</td>
</tr>
<tr>
<td>The bundle could not be resolved due to an error with the Bundle-NativeCode header.</td>
<td></td>
</tr>
<tr>
<td>static int RESOLVE_ERROR</td>
<td>60</td>
</tr>
<tr>
<td>The bundle was not resolved.</td>
<td></td>
</tr>
<tr>
<td>static int SECURITY_ERROR</td>
<td>60</td>
</tr>
<tr>
<td>The operation failed due to insufficient permissions.</td>
<td></td>
</tr>
<tr>
<td>static int START_TRANSIENT_ERROR</td>
<td>61</td>
</tr>
<tr>
<td>The start transient operation failed because the start level of the bundle is greater than the current framework start level</td>
<td></td>
</tr>
<tr>
<td>static int STATECHANGE_ERROR</td>
<td>60</td>
</tr>
<tr>
<td>The operation failed to complete the requested lifecycle state change.</td>
<td></td>
</tr>
<tr>
<td>static int UNSUPPORTED_OPERATION</td>
<td>59</td>
</tr>
<tr>
<td>No exception type is unspecified.</td>
<td></td>
</tr>
<tr>
<td>static int UNSPECIFIED</td>
<td>59</td>
</tr>
<tr>
<td>The operation was unsupported.</td>
<td></td>
</tr>
</tbody>
</table>
# Class BundleException

## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BundleException(String msg)</code></td>
<td>61</td>
</tr>
<tr>
<td>Creates a <code>BundleException</code> with the specified message.</td>
<td></td>
</tr>
<tr>
<td><code>BundleException(String msg, int type)</code></td>
<td>62</td>
</tr>
<tr>
<td>Creates a <code>BundleException</code> with the specified message and type.</td>
<td></td>
</tr>
<tr>
<td><code>BundleException(String msg, int type, Throwable cause)</code></td>
<td>61</td>
</tr>
<tr>
<td>Creates a <code>BundleException</code> with the specified message, type and exception cause.</td>
<td></td>
</tr>
<tr>
<td><code>BundleException(String msg, Throwable cause)</code></td>
<td>61</td>
</tr>
<tr>
<td>Creates a <code>BundleException</code> with the specified message and exception cause.</td>
<td></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get Cause()</code></td>
<td>62</td>
</tr>
<tr>
<td>Returns the cause of this exception or <code>null</code> if no cause was set.</td>
<td></td>
</tr>
<tr>
<td><code>get Nested Exception()</code></td>
<td>62</td>
</tr>
<tr>
<td>Returns the cause of this exception or <code>null</code> if no cause was specified when this exception was created.</td>
<td></td>
</tr>
<tr>
<td><code>getType()</code></td>
<td>63</td>
</tr>
<tr>
<td>Returns the type for this exception or <code>UNSPECIFIED</code> if the type was unspecified or unknown.</td>
<td></td>
</tr>
<tr>
<td><code>init Cause(Throwable cause)</code></td>
<td>62</td>
</tr>
<tr>
<td>Initializes the cause of this exception to the specified value.</td>
<td></td>
</tr>
</tbody>
</table>

## Field Detail

### UNSPECIFIED

```java
public static final int UNSPECIFIED = 0
```

No exception type is unspecified.

**Since:** 1.5

### UNSUPPORTED_OPERATION

```java
public static final int UNSUPPORTED_OPERATION = 1
```

The operation was unsupported.

**Since:** 1.5

### INVALID_OPERATION

```java
public static final int INVALID_OPERATION = 2
```

The operation was invalid.

**Since:** 1.5
Class BundleException

**MANIFEST_ERROR**

public static final int MANIFEST_ERROR = 3

The bundle manifest was in error.

Since:
1.5

**RESOLVE_ERROR**

public static final int RESOLVE_ERROR = 4

The bundle was not resolved.

Since:
1.5

**ACTIVATOR_ERROR**

public static final int ACTIVATOR_ERROR = 5

The bundle activator was in error.

Since:
1.5

**SECURITY_ERROR**

public static final int SECURITY_ERROR = 6

The operation failed due to insufficient permissions.

Since:
1.5

**STATECHANGE_ERROR**

public static final int STATECHANGE_ERROR = 7

The operation failed to complete the requested lifecycle state change.

Since:
1.5

**NATIVECODE_ERROR**

public static final int NATIVECODE_ERROR = 8

The bundle could not be resolved due to an error with the Bundle-NativeCode header.

Since:
1.5
Class BundleException

DUPLICATE_BUNDLE_ERROR

public static final int DUPLICATE_BUNDLE_ERROR = 9

The install or update operation failed because another already installed bundle has the same symbolic name and version.

Since: 1.5

START_TRANSIENT_ERROR

public static final int START_TRANSIENT_ERROR = 10

The start transient operation failed because the start level of the bundle is greater than the current framework start level

Since: 1.5

Constructor Detail

BundleException

public BundleException(String msg, Throwable cause)

Creates a BundleException with the specified message and exception cause.

Parameters:
  msg - The associated message.
  cause - The cause of this exception.

BundleException

public BundleException(String msg)

Creates a BundleException with the specified message.

Parameters:
  msg - The message.

BundleException

public BundleException(String msg, int type, Throwable cause)

Creates a BundleException with the specified message, type and exception cause.

Parameters:
  msg - The associated message.
  type - The type for this exception.
  cause - The cause of this exception.
Class BundleException

Since: 1.5

BundleException

public BundleException(String msg, int type)

Creates a BundleException with the specified message and type.

Parameters:

msg - The message.
type - The type for this exception.

Since: 1.5

Method Detail

getNestedException

public Throwable getNestedException()

Returns the cause of this exception or null if no cause was specified when this exception was created.

This method predates the general purpose exception chaining mechanism. The getCause() method is now the preferred means of obtaining this information.

Returns: The result of calling getCause().

getCause

public Throwable getCause()

Returns the cause of this exception or null if no cause was set.

Overrides:

getCause in class Throwable

Returns: The cause of this exception or null if no cause was set.

Since: 1.3

initCause

public Throwable initCause(Throwable cause)

Initializes the cause of this exception to the specified value.

Overrides:

initCause in class Throwable

Parameters:

cause - The cause of this exception.

Returns: This exception.
Throws:
   IllegalArgumentException - If the specified cause is this exception.
   IllegalStateException - If the cause of this exception has already been set.
Since: 1.3

getType

public int getType()

   Returns the type for this exception or UNSPECIFIED if the type was unspecified or unknown.

Returns:  
The type of this exception.
Since: 1.5
public interface BundleListener
extends EventListener

A BundleEvent listener. BundleListener is a listener interface that may be implemented by a bundle developer. When a BundleEvent is fired, it is asynchronously delivered to a BundleListener. The Framework delivers BundleEvent objects to a BundleListener in order and must not concurrently call a BundleListener.

A BundleListener object is registered with the Framework using the BundleContext.addBundleListener() method. BundleListeners are called with a BundleEvent object when a bundle has been installed, resolved, started, stopped, updated, unresolved, or uninstalled.

See Also:
BundleEvent
NotThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void bundleChanged(BundleEvent event)</td>
<td>64</td>
</tr>
</tbody>
</table>

Receives notification that a bundle has had a lifecycle change.

Method Detail

bundleChanged

void bundleChanged(BundleEvent event)

Receives notification that a bundle has had a lifecycle change.

Parameters:

event - The BundleEvent.
final public class BundlePermission
extends BasicPermission

A bundle's authority to require or provide a bundle or to receive or attach fragments.

A bundle symbolic name defines a unique fully qualified name. Wildcards may be used.

name ::= <symbolic name> | <symbolic name ending in ".*"> | *

Examples:

org.osgi.example.bundle
org.osgi.example.*
* 

BundlePermission has four actions: provide, require, host, and fragment. The provide action implies the require action.

Since: 1.3
Version: $Revision: 8338 $
ThreadSafe

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String FRAGMENT</td>
<td>The action string fragment.</td>
</tr>
<tr>
<td>static String HOST</td>
<td>The action string host.</td>
</tr>
<tr>
<td>static String PROVIDE</td>
<td>The action string provide.</td>
</tr>
<tr>
<td>static String REQUIRE</td>
<td>The action string require.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundlePermission(String symbolicName, String actions)</td>
<td>Defines the authority to provide and/or require and or specify a host fragment symbolic name within the OSGi environment.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(Object obj)</td>
<td>Determines the equality of two BundlePermission objects.</td>
</tr>
</tbody>
</table>
### Class BundlePermission

<table>
<thead>
<tr>
<th>String</th>
<th>getActions()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the canonical string representation of the BundlePermission actions.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th>hashCode()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the hash code value for this object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>implies(Permission p)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Determines if the specified permission is implied by this object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Permission Collection</th>
<th>newPermissionCollection()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a new PermissionCollection object suitable for storing BundlePermission objects.</td>
</tr>
</tbody>
</table>

### Field Detail

**PROVIDE**

public static final String PROVIDE = "provide"

The action string provide. The provide action implies the require action.

**REQUIRE**

public static final String REQUIRE = "require"

The action string require. The require action is implied by the provide action.

**HOST**

public static final String HOST = "host"

The action string host.

**FRAGMENT**

public static final String FRAGMENT = "fragment"

The action string fragment.

### Constructor Detail

**BundlePermission**

public BundlePermission(String symbolicName, String actions)

Defines the authority to provide and/or require and or specify a host fragment symbolic name within the OSGi environment.

Bundle Permissions are granted over all possible versions of a bundle. A bundle that needs to provide a bundle must have the appropriate BundlePermission for the symbolic name; a bundle that requires a bundle must have the appropriate BundlePermission for that symbolic name; a bundle that specifies a fragment host must have the appropriate BundlePermission for that symbolic name.
Class BundlePermission

Parameters:
symbolicName - The bundle symbolic name.
actions - provide, require, host, fragment (canonical order).

Method Detail

implies

public boolean implies(Permission p)

Determines if the specified permission is implied by this object.

This method checks that the symbolic name of the target is implied by the symbolic name of this object.
The list of BundlePermission actions must either match or allow for the list of the target object to imply the
target BundlePermission action.

The permission to provide a bundle implies the permission to require the named symbolic name.

x.y.*, "provide" \rightarrow x.y.z, "provide" is true
x.*, "require" \rightarrow x.y, "require" is true
x.*, "provide" \rightarrow x.y, "require" is true
x.y, "provide" \rightarrow x.y.z, "provide" is false

Overrides:
implies in class BasicPermission

Parameters:
p - The requested permission.

Returns:
true if the specified BundlePermission action is implied by this object; false otherwise.

getActions

public String getActions()

Returns the canonical string representation of the BundlePermission actions.

Always returns present BundlePermission actions in the following order: provide, require, host, fragment.

Overrides:
getActions in class BasicPermission

Returns:
Canonical string representation of the BundlePermission actions.

newPermissionCollection

public PermissionCollection newPermissionCollection()

Returns a new PermissionCollection object suitable for storing BundlePermission objects.

Overrides:
newPermissionCollection in class BasicPermission

Returns:
A new PermissionCollection object.
equals

public boolean equals(Object obj)

Determines the equality of two BundlePermission objects. This method checks that specified bundle has the same bundle symbolic name and BundlePermission actions as this BundlePermission object.

Overrides: equals in class BasicPermission

Parameters:
obj - The object to test for equality with this BundlePermission object.

Returns:
true if obj is a BundlePermission, and has the same bundle symbolic name and actions as this BundlePermission object; false otherwise.

hashCode

public int hashCode()

Returns the hash code value for this object.

Overrides: hashCode in class BasicPermission

Returns:
A hash code value for this object.
public interface BundleReference

A reference to a Bundle.

Since:
1.5

Version:
$Revision: 6860 $

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getBundle()</td>
<td>Returns the Bundle object associated with this BundleReference.</td>
</tr>
</tbody>
</table>

Method Detail

getBundle

Bundle getBundle()

    Returns the Bundle object associated with this BundleReference.

    Returns:
    The Bundle object associated with this BundleReference.
public interface Configurable

Deprecated.

Supports a configuration object.

Configurable is an interface that should be used by a bundle developer in support of a configurable service. Bundles that need to configure a service may test to determine if the service object is an instanceof Configurable.

Version:
$Revision: 8271 $

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getConfigurationObject ()</td>
<td>70</td>
</tr>
</tbody>
</table>

getConfigurationObject

Object getConfigurationObject ()

  Deprecated. As of 1.2. Please use Configuration Admin service.

  Returns this service's configuration object.

  Services implementing Configurable should take care when returning a service configuration object since this object is probably sensitive.

  If the Java Runtime Environment supports permissions, it is recommended that the caller is checked for some appropriate permission before returning the configuration object.

  Returns: The configuration object for this service.
  Throws: 
  
  SecurityException - If the caller does not have an appropriate permission and the Java Runtime Environment supports permissions.
public interface Constants

Defines standard names for the OSGi environment system properties, service properties, and Manifest header attribute keys.

The values associated with these keys are of type String, unless otherwise indicated.

Since: 1.1
Version: $Revision: 8523 $

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String ACTIVATION_LAZY</td>
<td>88</td>
</tr>
<tr>
<td>Bundle activation policy declaring the bundle must be activated when the first class load is made from the bundle.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_ACTIVATIONPOLICY</td>
<td>88</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's activation policy.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_ACTIVATOR</td>
<td>78</td>
</tr>
<tr>
<td>Manifest header attribute identifying the bundle's activator class.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_CATEGORY</td>
<td>76</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's category.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_CLASSPATH</td>
<td>76</td>
</tr>
<tr>
<td>Manifest header identifying a list of directories and embedded JAR files, which are bundle resources used to extend the bundle's classpath.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_CONTACTADDRESS</td>
<td>78</td>
</tr>
<tr>
<td>Manifest header identifying the contact address where problems with the bundle may be reported; for example, an email address.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_COPYRIGHT</td>
<td>76</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's copyright information.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_DESCRIPTION</td>
<td>76</td>
</tr>
<tr>
<td>Manifest header containing a brief description of the bundle's functionality.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_DOCURL</td>
<td>78</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's documentation URL, from which further information about the bundle may be obtained.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_LOCALIZATION</td>
<td>82</td>
</tr>
<tr>
<td>Manifest header identifying the base name of the bundle's localization entries.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_LOCALIZATION_DEFAULT_BASENAME</td>
<td>82</td>
</tr>
<tr>
<td>Default value for the Bundle-Localization manifest header.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_MANIFESTVERSION</td>
<td>83</td>
</tr>
<tr>
<td>Manifest header identifying the bundle manifest version.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_NAME</td>
<td>76</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's name.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_NATIVECODE</td>
<td>77</td>
</tr>
<tr>
<td>Manifest header identifying a number of hardware environments and the native language code libraries that the bundle is carrying for each of these environments.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_NATIVECODE_LANGUAGE</td>
<td>80</td>
</tr>
<tr>
<td>Manifest header attribute identifying the language in which the native bundle code is written specified in the Bundle-NativeCode manifest header.</td>
<td></td>
</tr>
</tbody>
</table>
### Interface Constants

<table>
<thead>
<tr>
<th>String</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUNDLE_NATIVECODE_OSNAMESPACE</td>
<td>Manifest header attribute identifying the operating system required to run native bundle code specified in the Bundle-NativeCode manifest header.</td>
</tr>
<tr>
<td>BUNDLE_NATIVECODE_OSVersions</td>
<td>Manifest header attribute identifying the operating system version required to run native bundle code specified in the Bundle-NativeCode manifest header.</td>
</tr>
<tr>
<td>BUNDLE_NATIVECODE_PROCESSOR</td>
<td>Manifest header attribute identifying the processor required to run native bundle code specified in the Bundle-NativeCode manifest header.</td>
</tr>
<tr>
<td>BUNDLE_REQUIREDEXECUTIONENVIRONMENT</td>
<td>Manifest header identifying the required execution environment for the bundle.</td>
</tr>
<tr>
<td>BUNDLE_SYMBOLICNAME</td>
<td>Manifest header identifying the bundle's symbolic name.</td>
</tr>
<tr>
<td>BUNDLE_SYMBOLICNAME_ATTRIBUTE</td>
<td>Manifest header attribute identifying the symbolic name of a bundle that exports a package specified in the Import-Package manifest header.</td>
</tr>
<tr>
<td>BUNDLE_UPDATELOCATION</td>
<td>Manifest header identifying the location from which a new bundle version is obtained during a bundle update operation.</td>
</tr>
<tr>
<td>BUNDLE_VENDOR</td>
<td>Manifest header identifying the bundle's vendor.</td>
</tr>
<tr>
<td>BUNDLE_VERSION</td>
<td>Manifest header identifying the bundle's version.</td>
</tr>
<tr>
<td>BUNDLE_VERSION_ATTRIBUTE</td>
<td>Manifest header attribute identifying a range of versions for a bundle specified in the Require-Bundle or Fragment-Host manifest headers.</td>
</tr>
<tr>
<td>DYNAMICIMPORT_PACKAGE</td>
<td>Manifest header identifying the packages that the bundle may dynamically import during execution.</td>
</tr>
<tr>
<td>EXCLUDE_DIRECTIVE</td>
<td>Manifest header directive identifying a list of classes to exclude in the exported package.</td>
</tr>
<tr>
<td>EXPORT_PACKAGE</td>
<td>Manifest header identifying the packages that the bundle offers to the Framework for export.</td>
</tr>
<tr>
<td>EXPORT_SERVICE</td>
<td>Deprecated. As of 1.2.</td>
</tr>
<tr>
<td>EXTENSION_BOOTCLASSPATH</td>
<td>Manifest header directive value identifying the type of extension fragment.</td>
</tr>
<tr>
<td>EXTENSION_DIRECTIVE</td>
<td>Manifest header directive identifying the type of the extension fragment.</td>
</tr>
<tr>
<td>EXTENSION_FRAMEWORK</td>
<td>Manifest header directive value identifying the type of extension fragment.</td>
</tr>
<tr>
<td>FRAGMENT_ATTACHMENT_ALWAYS</td>
<td>Manifest header directive value identifying a fragment attachment type of always.</td>
</tr>
<tr>
<td>FRAGMENT_ATTACHMENT_DIRECTIVE</td>
<td>Manifest header directive identifying if and when a fragment may attach to a host bundle.</td>
</tr>
<tr>
<td>FRAGMENT_ATTACHMENT_NEVER</td>
<td>Manifest header directive value identifying a fragment attachment type of never.</td>
</tr>
<tr>
<td>FRAGMENT_ATTACHMENT_RESOLVETIME</td>
<td>Manifest header directive value identifying a fragment attachment type of resolve-time.</td>
</tr>
<tr>
<td>FRAGMENT_HOST</td>
<td>Manifest header identifying the symbolic name of another bundle for which that the bundle is a fragment.</td>
</tr>
<tr>
<td>FRAMEWORK_BEGINNING_STARTLEVEL</td>
<td>Specifies the beginning start level of the framework.</td>
</tr>
<tr>
<td>String</td>
<td>FRAMEWORK_BOOTDELEGATION</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td>Framework environment property identifying packages for which the Framework must delegate class loading to the parent class loader of the bundle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_BUNDLE_PARENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies the parent class loader type for all bundle class loaders.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_BUNDLE_PARENT_APP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies to use the application class loader as the parent class loader for all bundle class loaders.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_BUNDLE_PARENT_BOOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies to use of the boot class loader as the parent class loader for all bundle class loaders.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_BUNDLE_PARENT_EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies to use the extension class loader as the parent class loader for all bundle class loaders.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_BUNDLE_PARENT_FRAMEWORK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies to use the framework class loader as the parent class loader for all bundle class loaders.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_COMMAND_ABSPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified the substitution string for the absolute path of a file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_EXECPERMISSION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies an optional OS specific command to set file permissions on extracted native code.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_EXECUTIONENVIRONMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Framework environment property identifying execution environments provided by the Framework.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_LANGUAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Framework environment property identifying the Framework implementation language (see ISO 639 for possible values).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_LIBRARY_EXTENSIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies a comma separated list of additional library file extensions that must be used when a bundle's class loader is searching for native libraries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_OS_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Framework environment property identifying the Framework host-computer's operating system.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_OS_VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Framework environment property identifying the Framework host-computer's operating system version number.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_PROCESSOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Framework environment property identifying the Framework host-computer's processor name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_SECURITY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies the type of security manager the framework must use.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_SECURITY_OSGI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies that a security manager that supports all security aspects of the OSGi core specification including postponed conditions must be installed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_STORAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specified the persistent storage area used by the framework.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_STORAGE_CLEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies if and when the persistent storage area for the framework should be cleaned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_STORAGE_CLEAN_ONFIRSTINIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies that the framework storage area must be cleaned before the framework is initialized for the first time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_SYSTEMPACKAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Framework environment property identifying packages which the system bundle must export.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>FRAMEWORK_SYSTEMPACKAGES_EXTRA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Framework environment property identifying extra packages which the system bundle must export from the current execution environment.</td>
</tr>
<tr>
<td>String</td>
<td>FRAMEWORK_TRUST_REPOSITORIES</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>String</td>
<td>FRAMEWORK_UUID</td>
</tr>
<tr>
<td>String</td>
<td>FRAMEWORK_VENDOR</td>
</tr>
<tr>
<td>String</td>
<td>FRAMEWORK_VERSION</td>
</tr>
<tr>
<td>String</td>
<td>FRAMEWORK_WINDOWSYSTEM</td>
</tr>
<tr>
<td>String</td>
<td>IMPORT_PACKAGE</td>
</tr>
<tr>
<td>String</td>
<td>IMPORT_SERVICE</td>
</tr>
<tr>
<td>String</td>
<td>INCLUDE_DIRECTIVE</td>
</tr>
<tr>
<td>String</td>
<td>MANDATORY_DIRECTIVE</td>
</tr>
<tr>
<td>String</td>
<td>OBJECTCLASS</td>
</tr>
<tr>
<td>String</td>
<td>PACKAGE_SPECIFICATION_VERSION</td>
</tr>
<tr>
<td>String</td>
<td>REMOTE_CONFIGS_SUPPORTED</td>
</tr>
<tr>
<td>String</td>
<td>REMOTE_INTENTS_SUPPORTED</td>
</tr>
<tr>
<td>String</td>
<td>REQUIRE_BUNDLE</td>
</tr>
<tr>
<td>String</td>
<td>RESOLUTION_DIRECTIVE</td>
</tr>
<tr>
<td>String</td>
<td>RESOLUTION_MANDATORY</td>
</tr>
<tr>
<td>String</td>
<td>RESOLUTION OPTIONAL</td>
</tr>
<tr>
<td>String</td>
<td>SELECTION_FILTER_ATTRIBUTE</td>
</tr>
<tr>
<td>String</td>
<td>SERVICE_DESCRIPTION</td>
</tr>
<tr>
<td>String</td>
<td>SERVICE_IMPORTED_CONFIGS</td>
</tr>
<tr>
<td>String</td>
<td>SERVICE_IMPORTED_INTENTS</td>
</tr>
<tr>
<td>String</td>
<td>SERVICE_IMPORTED_INTENTS_EXTRA</td>
</tr>
<tr>
<td>String</td>
<td>SERVICE_IMPORTED_INTERFACES</td>
</tr>
<tr>
<td>String</td>
<td>SERVICE_ID</td>
</tr>
<tr>
<td>String</td>
<td>Property Name</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>SERVICE_IMPORTED</td>
<td>Service property identifying the service as imported.</td>
</tr>
<tr>
<td>SERVICE_IMPORTED_CONFIGS</td>
<td>Service property identifying the configuration types used to import the service.</td>
</tr>
<tr>
<td>SERVICE_INTENTS</td>
<td>Service property identifying the intents that this service implement.</td>
</tr>
<tr>
<td>SERVICE_PID</td>
<td>Service property identifying a service's persistent identifier.</td>
</tr>
<tr>
<td>SERVICE_RANKING</td>
<td>Service property identifying a service's ranking number.</td>
</tr>
<tr>
<td>SERVICE_VENDOR</td>
<td>Service property identifying a service's vendor.</td>
</tr>
<tr>
<td>SINGLETON_DIRECTIVE</td>
<td>Manifest header directive identifying whether a bundle is a singleton.</td>
</tr>
<tr>
<td>SUPPORTS_BOOTCLASSPATH_EXTENSION</td>
<td>Framework environment property identifying whether the Framework supports bootclasspath extension bundles.</td>
</tr>
<tr>
<td>SUPPORTS_FRAMEWORK_EXTENSION</td>
<td>Framework environment property identifying whether the Framework supports framework extension bundles.</td>
</tr>
<tr>
<td>SUPPORTS_FRAMEWORK_FRAGMENT</td>
<td>Framework environment property identifying whether the Framework supports fragment bundles.</td>
</tr>
<tr>
<td>SUPPORTS_FRAMEWORK_REQUIREBUNDLE</td>
<td>Framework environment property identifying whether the Framework supports the <code>Require-Bundle</code> manifest header.</td>
</tr>
<tr>
<td>SYSTEM_BUNDLE_LOCATION</td>
<td>Location identifier of the OSGi <code>system bundle</code> , which is defined to be &quot;System Bundle&quot;.</td>
</tr>
<tr>
<td>SYSTEM_BUNDLE_SYMBOLICNAME</td>
<td>Alias for the symbolic name of the OSGi <code>system bundle</code> .</td>
</tr>
<tr>
<td>USES_DIRECTIVE</td>
<td>Manifest header directive identifying a list of packages that an exported package uses.</td>
</tr>
<tr>
<td>VERSION_ATTRIBUTE</td>
<td>Manifest header attribute identifying the version of a package specified in the Export-Package or Import-Package manifest header.</td>
</tr>
<tr>
<td>VISIBILITY_DIRECTIVE</td>
<td>Manifest header directive identifying the visibility of a required bundle in the Require-Bundle manifest header.</td>
</tr>
<tr>
<td>VISIBILITY_PRIVATE</td>
<td>Manifest header directive value identifying a private visibility type.</td>
</tr>
<tr>
<td>VISIBILITY_REEXPORT</td>
<td>Manifest header directive value identifying a reexport visibility type.</td>
</tr>
</tbody>
</table>

**Field Detail**

**SYSTEM_BUNDLE_LOCATION**

```java
public static final String SYSTEM_BUNDLE_LOCATION = "System Bundle"
```

Location identifier of the OSGi `system bundle` , which is defined to be "System Bundle".

**SYSTEM_BUNDLE_SYMBOLICNAME**

```java
public static final String SYSTEM_BUNDLE_SYMBOLICNAME = "system.bundle"
```
Interface Constants

Alias for the symbolic name of the OSGi system bundle. It is defined to be "system.bundle".

Since: 1.3

BUNDLECATEGORY

public static final String BUNDLECATEGORY = "Bundle-Category"

Manifest header identifying the bundle's category.

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

BUNDLECLASSPATH

public static final String BUNDLECLASSPATH = "Bundle-ClassPath"

Manifest header identifying a list of directories and embedded JAR files, which are bundle resources used to extend the bundle's classpath.

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

BUNDLECOPYRIGHT

public static final String BUNDLECOPYRIGHT = "Bundle-Copyright"

Manifest header identifying the bundle's copyright information.

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

BUNDLEDESCRIPTION

public static final String BUNDLEDESCRIPTION = "Bundle-Description"

Manifest header containing a brief description of the bundle's functionality.

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

BUNDLENAMESPACE

public static final String BUNDLENAMESPACE = "Bundle-Name"

Manifest header identifying the bundle's name.

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

```java
public static final String BUNDLE_NATIVECODE = "Bundle-NativeCode"
```

Manifest header identifying a number of hardware environments and the native language code libraries that the bundle is carrying for each of these environments.

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

---

**EXPORT_PACKAGE**

```java
public static final String EXPORT_PACKAGE = "Export-Package"
```

Manifest header identifying the packages that the bundle offers to the Framework for export.

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

---

**EXPORT_SERVICE**

```java
public static final String EXPORT_SERVICE = "Export-Service"
```

Deprecated.

Manifest header identifying the fully qualified class names of the services that the bundle may register (used for informational purposes only).

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

---

**IMPORT_PACKAGE**

```java
public static final String IMPORT_PACKAGE = "Import-Package"
```

Manifest header identifying the packages on which the bundle depends.

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

---

**DYNAMICIMPORT_PACKAGE**

```java
public static final String DYNAMICIMPORT_PACKAGE = "DynamicImport-Package"
```

Manifest header identifying the packages that the bundle may dynamically import during execution.

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

**Since:**

1.2
IMPORT_SERVICE

public static final String IMPORT_SERVICE = "Import-Service"

Deprecated.

Manifest header identifying the fully qualified class names of the services that the bundle requires (used for informational purposes only).

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

BUNDLE_VENDOR

public static final String BUNDLE_VENDOR = "Bundle-Vendor"

Manifest header identifying the bundle's vendor.

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

BUNDLE_VERSION

public static final String BUNDLE_VERSION = "Bundle-Version"

Manifest header identifying the bundle's version.

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

BUNDLE_DOCURL

public static final String BUNDLE_DOCURL = "Bundle-DocURL"

Manifest header identifying the bundle's documentation URL, from which further information about the bundle may be obtained.

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

BUNDLE_CONTACTADDRESS

public static final String BUNDLE_CONTACTADDRESS = "Bundle-ContactAddress"

Manifest header identifying the contact address where problems with the bundle may be reported; for example, an email address.

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

BUNDLE_ACTIVATOR

public static final String BUNDLE_ACTIVATOR = "Bundle-Activator"
Interface Constants

Manifest header attribute identifying the bundle's activator class.

If present, this header specifies the name of the bundle resource class that implements the BundleActivator interface and whose start and stop methods are called by the Framework when the bundle is started and stopped, respectively.

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

BUNDLE_UPDATELOCATION

public static final String BUNDLE_UPDATELOCATION = "Bundle-UpdateLocation"

Manifest header identifying the location from which a new bundle version is obtained during a bundle update operation.

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

PACKAGE_SPECIFICATION_VERSION

public static final String PACKAGE_SPECIFICATION_VERSION = "specification-version"

Deprecated.

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

BUNDLE_NATIVECODE_PROCESSOR

public static final String BUNDLE_NATIVECODE_PROCESSOR = "processor"

Manifest header attribute identifying the processor required to run native bundle code specified in the Bundle-NativeCode manifest header).

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

Bundle-NativeCode: http.so ; processor=x86 ...

See Also:

BUNDLE_NATIVECODE

BUNDLE_NATIVECODE_OSNAMESPACE

public static final String BUNDLE_NATIVECODE_OSNAMESPACE = "osname"

Manifest header attribute identifying the operating system required to run native bundle code specified in the Bundle-NativeCode manifest header).

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

Bundle-NativeCode: http.so ; osname=Linux ...

See Also:

BUNDLE_NATIVECODE
**Interface Constants**

**BUNDLE_NATIVECODE_OSVersions**

```java
public static final String BUNDLE_NATIVECODE_OSVersions = "osversion"
```

Manifest header attribute identifying the operating system version required to run native bundle code specified in the Bundle-NativeCode manifest header.

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

```text
Bundle-NativeCode: http.so ; osversion="2.34" ...
```

See Also: [BUNDLE_NATIVECODE](#)

**BUNDLE_NATIVECODE_LANGUAGE**

```java
public static final String BUNDLE_NATIVECODE_LANGUAGE = "language"
```

Manifest header attribute identifying the language in which the native bundle code is written specified in the Bundle-NativeCode manifest header. See ISO 639 for possible values.

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

```text
Bundle-NativeCode: http.so ; language=nl_be ...
```

See Also: [BUNDLE_NATIVECODE](#)

**BUNDLE_REQUIREDEXECUTIONENVIRONMENT**

```java
public static final String BUNDLE_REQUIREDEXECUTIONENVIRONMENT = "Bundle-RequiredExecutionEnvironment"
```

Manifest header identifying the required execution environment for the bundle. The service platform may run this bundle if any of the execution environments named in this header matches one of the execution environments it implements.

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

Since: 1.2

**BUNDLE_SYMBOLICNAME**

```java
public static final String BUNDLE_SYMBOLICNAME = "Bundle-SymbolicName"
```

Manifest header 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
**SINGLETON_DIRECTIVE**

public static final String SINGLETON_DIRECTIVE = "singleton"

Manifest header directive identifying whether a bundle is a singleton. The default value is false.

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

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

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

Since: 1.3
See Also: BUNDLE_SYMBOLICNAME

**FRAGMENT_ATTACHMENT_DIRECTIVE**

public static final String FRAGMENT_ATTACHMENT_DIRECTIVE = "fragment-attachment"

Manifest header directive identifying if and when a fragment may attach to a host bundle. The default value is always.

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

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

Since: 1.3
See Also: BUNDLE_SYMBOLICNAME, FRAGMENT_ATTACHMENT_ALWAYS, FRAGMENT_ATTACHMENT_RESOLVETIME, FRAGMENT_ATTACHMENT_NEVER

**FRAGMENT_ATTACHMENT_ALWAYS**

public static final String FRAGMENT_ATTACHMENT_ALWAYS = "always"

Manifest header directive value 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 directive 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_DIRECTIVE

**FRAGMENT_ATTACHMENT_RESOLVETIME**

public static final String FRAGMENT_ATTACHMENT_RESOLVETIME = "resolve-time"

Manifest header directive value 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 directive 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_DIRECTIVE

### FRAGMENT_ATTACHMENT_NEVER

```
public static final String FRAGMENT_ATTACHMENT_NEVER = "never"
```

Manifest header directive value 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 directive 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_DIRECTIVE

### BUNDLE_LOCALIZATION

```
public static final String BUNDLE_LOCALIZATION = "Bundle-Localization"
```

Manifest header identifying the base name of the bundle's localization entries.

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

Since: 1.3
See Also: BUNDLE_LOCALIZATION_DEFAULT_BASENAME

### BUNDLE_LOCALIZATION_DEFAULT_BASENAME

```
public static final String BUNDLE_LOCALIZATION_DEFAULT_BASENAME = "OSGI-INF/l10n/bundle"
```

Default value for the Bundle-Localization manifest header.

Since: 1.3
See Also: BUNDLE_LOCALIZATION

### REQUIRE_BUNDLE

```
public static final String REQUIRE_BUNDLE = "Require-Bundle"
```

Manifest header 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

**BUNDLE_VERSION_ATTRIBUTE**

public static final String BUNDLE_VERSION_ATTRIBUTE = "bundle-version"

Manifest header attribute 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: REQUIRE_BUNDLE

**FRAGMENT_HOST**

public static final String FRAGMENT_HOST = "Fragment-Host"

Manifest header 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

**SELECTION_FILTER_ATTRIBUTE**

public static final String SELECTION_FILTER_ATTRIBUTE = "selection-filter"

Manifest header attribute 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)"; ...
```

Since: 1.3
See Also: BUNDLE_NATIVECODE

**BUNDLE_MANIFESTVERSION**

public static final String BUNDLE_MANIFESTVERSION = "Bundle-ManifestVersion"
Manifest header 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 Release 4, or later, syntax must specify a bundle manifest version.

The bundle manifest version defined by OSGi Release 4 or, more specifically, by version 1.3 of the OSGi Core Specification is "2".

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

Since: 1.3

---

**VERSION_ATTRIBUTE**

public static final String VERSION_ATTRIBUTE = "version"

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

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: EXPORT_PACKAGE, IMPORT_PACKAGE

---

**BUNDLE_SYMBOLICNAME_ATTRIBUTE**

public static final String BUNDLE_SYMBOLICNAME_ATTRIBUTE = "bundle-symbolic-name"

Manifest header attribute identifying the symbolic name of a bundle that 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: IMPORT_PACKAGE

---

**RESOLUTION_DIRECTIVE**

public static final String RESOLUTION_DIRECTIVE = "resolution"

Manifest header directive identifying the resolution type in the Import-Package or Require-Bundle manifest header. The default value is mandatory.

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
Interface Constants

See Also:
IMPORT_PACKAGE, REQUIRE_BUNDLE, RESOLUTION_MANDATORY, RESOLUTION_OPTIONAL

RESOLUTION_MANDATORY

public static final String RESOLUTION_MANDATORY = "mandatory"

Manifest header directive value 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

RESOLUTION_OPTIONAL

public static final String RESOLUTION_OPTIONAL = "optional"

Manifest header directive value 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

USES_DIRECTIVE

public static final String USES_DIRECTIVE = "uses"

Manifest header directive identifying a list of packages that an exported package uses.

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

    Export-Package: org.osgi.util.tracker; uses:="org.osgi.framework"

Since: 1.3
See Also: EXPORT_PACKAGE

INCLUDE_DIRECTIVE

public static final String INCLUDE_DIRECTIVE = "include"
Manifest header directive identifying a list of classes to include in the exported package.

This directive is used by the Export-Package manifest header to identify a list of classes of the specified package which must be allowed to be exported. The directive value is encoded in the Export-Package manifest header like:

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

This directive is also used by the Bundle-ActivationPolicy manifest header to identify the packages from which class loads will trigger lazy activation. The directive value is encoded in the Bundle-ActivationPolicy manifest header like:

```
Bundle-ActivationPolicy: lazy; include:="org.osgi.framework"
```

Since: 1.3

See Also: EXPORT_PACKAGE, BUNDLE_ACTIVATIONPOLICY

---

**EXCLUDE_DIRECTIVE**

```java
public static final String EXCLUDE_DIRECTIVE = "exclude"
```

Manifest header directive identifying a list of classes to exclude in the exported package.

This directive is used by the Export-Package manifest header to identify a list of classes of the specified package which must not be allowed to be exported. The directive value is encoded in the Export-Package manifest header like:

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

This directive is also used by the Bundle-ActivationPolicy manifest header to identify the packages from which class loads will not trigger lazy activation. The directive value is encoded in the Bundle-ActivationPolicy manifest header like:

```
Bundle-ActivationPolicy: lazy; exclude:="org.osgi.framework"
```

Since: 1.3

See Also: EXPORT_PACKAGE, BUNDLE_ACTIVATIONPOLICY

---

**MANDATORY_DIRECTIVE**

```java
public static final String MANDATORY_DIRECTIVE = "mandatory"
```

Manifest header directive 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: EXPORT_PACKAGE
### VISIBILITY_DIRECTIVE

public static final String `VISIBILITY_DIRECTIVE` = "visibility"

Manifest header directive identifying the visibility of a required bundle in the Require-Bundle manifest header. The default value is `private`.

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`, `VISIBILITY_PRIVATE`, `VISIBILITY_REEXPORT`

### VISIBILITY_PRIVATE

public static final String `VISIBILITY_PRIVATE` = "private"

Manifest header directive value 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`

### VISIBILITY_REEXPORT

public static final String `VISIBILITY_REEXPORT` = "reexport"

Manifest header directive value 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`

### EXTENSION_DIRECTIVE

public static final String `EXTENSION_DIRECTIVE` = "extension"

Manifest header directive identifying the type of the extension fragment.

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

```
Fragment-Host: system.bundle; extension:="framework"
```
Interface Constants

Since: 1.3
See Also: FRAGMENT_HOST, EXTENSION_FRAMEWORK, EXTENSION_BOOTCLASSPATH

EXTENSION_FRAMEWORK

class static final String EXTENSION_FRAMEWORK = "framework"

Manifest header directive value identifying the type of extension fragment. An extension fragment type of framework indicates that the extension fragment is to be loaded by the framework's class loader.

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

Fragment-Host: system.bundle; extension:="framework"

Since: 1.3
See Also: EXTENSION_DIRECTIVE

EXTENSION_BOOTCLASSPATH

class static final String EXTENSION_BOOTCLASSPATH = "bootclasspath"

Manifest header directive value identifying the type of extension fragment. An extension fragment type of bootclasspath indicates that the extension fragment is to be loaded by the boot class loader.

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

Fragment-Host: system.bundle; extension:="bootclasspath"

Since: 1.3
See Also: EXTENSION_DIRECTIVE

BUNDLE_ACTIVATIONPOLICY

class static final String BUNDLE_ACTIVATIONPOLICY = "Bundle-ActivationPolicy"

Manifest header identifying the bundle's activation policy.

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

Since: 1.4
See Also: ACTIVATION_LAZY, INCLUDE_DIRECTIVE, EXCLUDE_DIRECTIVE

ACTIVATION_LAZY

class static final String ACTIVATION_LAZY = "lazy"

Bundle activation policy declaring the bundle must be activated when the first class load is made from the bundle.
A bundle with the lazy activation policy that is started with the `START_ACTIVATION_POLICY` option will wait in the `STARTING` state until the first class load from the bundle occurs. The bundle will then be activated before the class is returned to the requester.

The activation policy value is specified as in the Bundle-ActivationPolicy manifest header like:

```
Bundle-ActivationPolicy: lazy
```

**Since:** 1.4

**See Also:**

`Bundle_ACTIVATIONPOLICY`, `Bundle.start(int)`, `Bundle.START_ACTIVATION_POLICY`

---

### FRAMEWORK_VERSION

```
public static final String FRAMEWORK_VERSION = "org.osgi.framework.version"
```

Framework environment property identifying the Framework version.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

---

### FRAMEWORK_VENDOR

```
public static final String FRAMEWORK_VENDOR = "org.osgi.framework.vendor"
```

Framework environment property identifying the Framework implementation vendor.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

---

### FRAMEWORK_LANGUAGE

```
public static final String FRAMEWORK_LANGUAGE = "org.osgi.framework.language"
```

Framework environment property identifying the Framework implementation language (see ISO 639 for possible values).

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

---

### FRAMEWORK_OS_NAME

```
public static final String FRAMEWORK_OS_NAME = "org.osgi.framework.os.name"
```

Framework environment property identifying the Framework host-computer's operating system.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

---

### FRAMEWORK_OS_VERSION

```
public static final String FRAMEWORK_OS_VERSION = "org.osgi.framework.os.version"
```

Framework environment property identifying the Framework host-computer's operating system version number.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.
Interface Constants

**FRAMEWORK_PROCESSOR**

```java
public static final String FRAMEWORK_PROCESSOR = "org.osgi.framework.processor"
```

Framework environment property identifying the Framework host-computer's processor name.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

**FRAMEWORK_EXECUTIONENVIRONMENT**

```java
public static final String FRAMEWORK_EXECUTIONENVIRONMENT = "org.osgi.framework.executionenvironment"
```

Framework environment property identifying execution environments provided by the Framework.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

*Since: 1.2*

**FRAMEWORK_BOOTDELEGATION**

```java
public static final String FRAMEWORK_BOOTDELEGATION = "org.osgi.framework.bootdelegation"
```

Framework environment property identifying packages for which the Framework must delegate class loading to the parent class loader of the bundle.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

*Since: 1.3*

*See Also:* [FRAMEWORK_BUNDLE_PARENT](#)

**FRAMEWORK_SYSTEMPACKAGES**

```java
public static final String FRAMEWORK_SYSTEMPACKAGES = "org.osgi.framework.system.packages"
```

Framework environment property identifying packages which the system bundle must export.

If this property is not specified then the framework must calculate a reasonable default value for the current execution environment.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

*Since: 1.3*

**FRAMEWORK_SYSTEMPACKAGES_EXTRA**

```java
public static final String FRAMEWORK_SYSTEMPACKAGES_EXTRA = "org.osgi.framework.system.packages.extra"
```

Framework environment property identifying extra packages which the system bundle must export from the current execution environment.
This property is useful for configuring extra system packages in addition to the system packages calculated by the framework.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

**Since:** 1.5

**See Also:** `FRAMEWORK_SYSTEMPACKAGES`

---

**SUPPORTS_FRAMEWORK_EXTENSION**

```java
public static final String SUPPORTS_FRAMEWORK_EXTENSION =
"org.osgi.supports.framework.extension"
```

Framework environment property identifying whether the Framework supports framework extension bundles.

As of version 1.4, the value of this property must be `true`. The Framework must support framework extension bundles.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

**Since:** 1.3

---

**SUPPORTS_BOOTCLASSPATH_EXTENSION**

```java
public static final String SUPPORTS_BOOTCLASSPATH_EXTENSION =
"org.osgi.supports.bootclasspath.extension"
```

Framework environment property identifying whether the Framework supports bootclasspath extension bundles.

If the value of this property is `true`, then the Framework supports bootclasspath extension bundles. The default value is `false`.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

**Since:** 1.3

---

**SUPPORTS_FRAMEWORK_FRAGMENT**

```java
public static final String SUPPORTS_FRAMEWORK_FRAGMENT =
"org.osgi.supports.framework.fragment"
```

Framework environment property identifying whether the Framework supports fragment bundles.

As of version 1.4, the value of this property must be `true`. The Framework must support fragment bundles.

The value of this property may be retrieved by calling the `BundleContext.getProperty` method.

**Since:** 1.3
Interface Constants

SUPPORTS_FRAMEWORK_REQUIREBUNDLE

```java
public static final String SUPPORTS_FRAMEWORK_REQUIREBUNDLE =
"org.osgi.supports.framework.requirebundle"
```

Framework environment property identifying whether the Framework supports the Require-Bundle manifest header.

As of version 1.4, the value of this property must be true. The Framework must support the Require-Bundle manifest header.

The value of this property may be retrieved by calling the BundleContext.getProperty method.

Since: 1.3

FRAMEWORK_SECURITY

```java
public static final String FRAMEWORK_SECURITY = "org.osgi.framework.security"
```

Specifies the type of security manager the framework must use. If not specified then the framework will not set the VM security manager.

Since: 1.5

See Also: FRAMEWORK_SECURITY_OSGI

FRAMEWORK_SECURITY_OSGI

```java
public static final String FRAMEWORK_SECURITY_OSGI = ".osgi"
```

Specifies that a security manager that supports all security aspects of the OSGi core specification including postponed conditions must be installed.

If this value is specified and there is a security manager already installed, then a SecurityException must be thrown when the Framework is initialized.

Since: 1.5

See Also: FRAMEWORK_SECURITY

FRAMEWORK_STORAGE

```java
public static final String FRAMEWORK_STORAGE = "org.osgi.framework.storage"
```

Specified the persistent storage area used by the framework. The value of this property must be a valid file path in the file system to a directory. If the specified directory does not exist then the framework will create the directory. If the specified path exists but is not a directory or if the framework fails to create the storage directory, then framework initialization must fail. The framework is free to use this directory as it sees fit. This area can not be shared with anything else.

If this property is not set, the framework should use a reasonable platform default for the persistent storage area.

Since: 1.5
Interface Constants

FRAMEWORK_STORAGE_CLEAN

public static final String FRAMEWORK_STORAGE_CLEAN = "org.osgi.framework.storage.clean"

Specifies if and when the persistent storage area for the framework should be cleaned. If this property is
not set, then the framework storage area must not be cleaned.

Since: 1.5
See Also: FRAMEWORK_STORAGE_CLEAN_ONFIRSTINIT

FRAMEWORK_STORAGE_CLEAN_ONFIRSTINIT

public static final String FRAMEWORK_STORAGE_CLEAN_ONFIRSTINIT = "onFirstInit"

Specifies that the framework storage area must be cleaned before the framework is initialized for the first
time. Subsequent inits, starts or updates of the framework will not result in cleaning the framework storage
area.

Since: 1.5

FRAMEWORK_LIBRARY_EXTENSIONS

public static final String FRAMEWORK_LIBRARY_EXTENSIONS =
"org.osgi.framework.library.extensions"

Specifies a comma separated list of additional library file extensions that must be used when a bundle's
class loader is searching for native libraries. If this property is not set, then only the library name returned
by System.mapLibraryName(String) will be used to search. This is needed for certain operating systems
which allow more than one extension for a library. For example, AIX allows library extensions of .a and
.so, but System.mapLibraryName(String) will only return names with the .a extension.

Since: 1.5

FRAMEWORK_EXECPERMISSION

public static final String FRAMEWORK_EXECPERMISSION =
"org.osgi.framework.command.execpermission"

Specifies an optional OS specific command to set file permissions on extracted native code. On some
operating systems, it is required that native libraries be set to executable. This optional property allows you
to specify the command. For example, on a UNIX style OS, this property could have the following value.

chmod +rx ${abspath}

The ${abspath} is used by the framework to substitute the actual absolute file path.

Since: 1.5
### FRAMEWORK_COMMAND_ABSPATH

```java
public static final String FRAMEWORK_COMMAND_ABSPATH = "abspath"
```

Specified the substitution string for the absolute path of a file.

**Since:** 1.6

**See Also:** FRAMEWORK_EXECPERMISSION

### FRAMEWORK_TRUST_REPOSITORIES

```java
public static final String FRAMEWORK_TRUST_REPOSITORIES =
"org.osgi.framework.trust.repositories"
```

Specifies the trust repositories used by the framework. The value is a java.io.File.pathSeparator separated list of valid file paths to files that contain key stores of type JKS. The framework will use the key stores as trust repositories to authenticate certificates of trusted signers. The key stores are only used as read-only trust repositories to access public keys. No passwords are required to access the key stores' public keys.

Note that framework implementations are allowed to use other trust repositories in addition to the trust repositories specified by this property. How these other trust repositories are configured and populated is implementation specific.

**Since:** 1.5

### FRAMEWORK_WINDOWSYSTEM

```java
public static final String FRAMEWORK_WINDOWSYSTEM = "org.osgi.framework.windowsystem"
```

Specifies the current windowing system. The framework should provide a reasonable default if this is not set.

**Since:** 1.5

### FRAMEWORK_BEGINNING_STARTLEVEL

```java
public static final String FRAMEWORK_BEGINNING_STARTLEVEL =
"org.osgi.framework.startlevel.beginning"
```

Specifies the beginning start level of the framework.

**Since:** 1.5

**See Also:** "Core Specification, section 8.2.3."

### FRAMEWORK_BUNDLE_PARENT

```java
public static final String FRAMEWORK_BUNDLE_PARENT = "org.osgi.framework.bundle.parent"
```

Specifies the parent class loader type for all bundle class loaders. Default value is **boot**.
Interface Constants

Since: 1.5
See Also: FRAMEWORK_BUNDLE_PARENT_BOOT, FRAMEWORK_BUNDLE_PARENT_EXT,
FRAMEWORK_BUNDLE_PARENT_APP, FRAMEWORK_BUNDLE_PARENT_FRAMEWORK

FRAMEWORK_BUNDLE_PARENT_BOOT

public static final String FRAMEWORK_Bundle_PARENT_BOOT = "boot"

Specifies to use of the boot class loader as the parent class loader for all bundle class loaders.

Since: 1.5
See Also: FRAMEWORK_BUNDLE_PARENT

FRAMEWORK_BUNDLE_PARENT_EXT

public static final String FRAMEWORK_BUNDLE_PARENT_EXT = "ext"

Specifies to use the extension class loader as the parent class loader for all bundle class loaders.

Since: 1.5
See Also: FRAMEWORK_BUNDLE_PARENT

FRAMEWORK_BUNDLE_PARENT_APP

public static final String FRAMEWORK_BUNDLE_PARENT_APP = "app"

Specifies to use the application class loader as the parent class loader for all bundle class loaders. Depending on how the framework is launched, this may refer to the same class loader as

Since: 1.5
See Also: FRAMEWORK_BUNDLE_PARENT

FRAMEWORK_BUNDLE_PARENT_FRAMEWORK

public static final String FRAMEWORK_BUNDLE_PARENT_FRAMEWORK = "framework"

Specifies to use the framework class loader as the parent class loader for all bundle class loaders. The framework class loader is the class loader used to load the framework implementation. Depending on how the framework is launched, this may refer to the same class loader as

Since: 1.5
See Also: FRAMEWORK_BUNDLE_PARENT
**OBJECTCLASS**

```java
public static final String OBJECTCLASS = "objectClass"
```

Service property identifying all of the class names under which a service was registered in the Framework. The value of this property must be of type `String[]`. This property is set by the Framework when a service is registered.

**SERVICE_ID**

```java
public static final String SERVICE_ID = "service.id"
```

Service property identifying a service's registration number. The value of this property must be of type `Long`. The value of this property is assigned by the Framework when a service is registered. The Framework assigns a unique value that is larger than all previously assigned values since the Framework was started. These values are NOT persistent across restarts of the Framework.

**SERVICE_PID**

```java
public static final String SERVICE_PID = "service.pid"
```

Service property identifying a service's persistent identifier. This property may be supplied in the `properties Dictionary` object passed to the `BundleContext.registerService` method. The value of this property must be of type `String`, `String[]`, or `Collection` of `String`. A service's persistent identifier uniquely identifies the service and persists across multiple Framework invocations.

By convention, every bundle has its own unique namespace, starting with the bundle's identifier (see `Bundle.getBundleId()`) and followed by a dot (.). A bundle may use this as the prefix of the persistent identifiers for the services it registers.

**SERVICE_RANKING**

```java
public static final String SERVICE_RANKING = "service.ranking"
```

Service property identifying a service's ranking number. This property may be supplied in the `properties Dictionary` object passed to the `BundleContext.registerService` method. The value of this property must be of type `Integer`.

The service ranking is used by the Framework to determine the natural order of services, see `ServiceReference.compareTo()`, and the default service to be returned from a call to the `BundleContext.getServiceReference()` method. The default ranking is zero (0). A service with a ranking of `Integer.MAX_VALUE` is very likely to be returned as the default service, whereas a service with a ranking of `Integer.MIN_VALUE` is very unlikely to be returned.

If the supplied property value is not of type `Integer`, it is deemed to have a ranking value of zero.
**SERVICE_VENDOR**

```java
public static final String SERVICE_VENDOR = "service.vendor"
```

Service property identifying a service's vendor.

This property may be supplied in the properties Dictionary object passed to the BundleContext.registerService method.

**SERVICE_DESCRIPTION**

```java
public static final String SERVICE_DESCRIPTION = "service.description"
```

Service property identifying a service's description.

This property may be supplied in the properties Dictionary object passed to the BundleContext.registerService method.

**FRAMEWORK_UUID**

```java
public static final String FRAMEWORK_UUID = "org.osgi.framework.uuid"
```

Framework environment property identifying the Framework's universally unique identifier (UUID). A UUID represents a 128-bit value. See the toString method of java.util.UUID for the format of this string.

The value of this property may be retrieved by calling the BundleContext.getProperty method.

Since: 1.6

**REMOTE_CONFIGS_SUPPORTED**

```java
public static final String REMOTE_CONFIGS_SUPPORTED = "remote.configs.supported"
```

Service property identifying the configuration types supported by a distribution provider. Registered by the distribution provider on one of its services to indicate the supported configuration types.

The value of this property must be of type String, String[], or Collection of String.

Since: 1.6

See Also:
"Remote Services Specification"

**REMOTE_INTENTS_SUPPORTED**

```java
public static final String REMOTE_INTENTS_SUPPORTED = "remote.intents.supported"
```

Service property identifying the intents supported by a distribution provider. Registered by the distribution provider on one of its services to indicate the vocabulary of implemented intents.

The value of this property must be of type String, String[], or Collection of String.

Since: 1.6
Interface Constants

See Also:
"Remote Services Specification"

SERVICE_EXPORTED_CONFIGS

public static final String SERVICE_EXPORTED_CONFIGS = "service.exported.configs"

Service property identifying the configuration types that should be used to export the service. Each configuration type represents the configuration parameters for an endpoint. A distribution provider should create an endpoint for each configuration type that it supports.

This property may be supplied in the properties Dictionary object passed to the BundleContext.registerService method. The value of this property must be of type String, String[], or Collection of String.

Since: 1.6
See Also: "Remote Services Specification"

SERVICE_EXPORTED_INTENTS

public static final String SERVICE_EXPORTED_INTENTS = "service.exported.intents"

Service property identifying the intents that the distribution provider must implement to distribute the service. Intents listed in this property are reserved for intents that are critical for the code to function correctly, for example, ordering of messages. These intents should not be configurable.

This property may be supplied in the properties Dictionary object passed to the BundleContext.registerService method. The value of this property must be of type String, String[], or Collection of String.

Since: 1.6
See Also: "Remote Services Specification"

SERVICE_EXPORTED_INTENTS_EXTRA

public static final String SERVICE_EXPORTED_INTENTS_EXTRA = "service.exported.intents.extra"

Service property identifying the extra intents that the distribution provider must implement to distribute the service. This property is merged with the service.exported.intents property before the distribution provider interprets the listed intents; it has therefore the same semantics but the property should be configurable so the administrator can choose the intents based on the topology. Bundles should therefore make this property configurable, for example through the Configuration Admin service.

This property may be supplied in the properties Dictionary object passed to the BundleContext.registerService method. The value of this property must be of type String, String[], or Collection of String.

Since: 1.6
See Also: "Remote Services Specification"
## Interface Constants

### SERVICE_EXPORTED_INTERFACES

```java
public static final String SERVICE_EXPORTED_INTERFACES = "service.exported.interfaces"
```

Service property marking the service for export. It defines the interfaces under which this service can be exported. This list must be a subset of the types under which the service was registered. The single value of an asterisk ("*", `\u002A`) indicates all the interface types under which the service was registered excluding the non-interface types. It is strongly recommended to only export interface types and not concrete classes due to the complexity of creating proxies for some type of concrete classes.

This property may be supplied in the `properties` `Dictionary` object passed to the `BundleContext.registerService` method. The value of this property must be of type `String`, `String[]`, or `Collection` of `String`.

Since: 1.6

See Also: "Remote Services Specification"

### SERVICE_IMPORTED

```java
public static final String SERVICE_IMPORTED = "service.imported"
```

Service property identifying the service as imported. This service property must be set by a distribution provider to any value when it registers the endpoint proxy as an imported service. A bundle can use this property to filter out imported services.

The value of this property may be of any type.

Since: 1.6

See Also: "Remote Services Specification"

### SERVICE_IMPORTED_CONFIGS

```java
public static final String SERVICE_IMPORTED_CONFIGS = "service.imported.configs"
```

Service property identifying the configuration types used to import the service. Any associated properties for this configuration types must be properly mapped to the importing system. For example, a URL in these properties must point to a valid resource when used in the importing framework. If multiple configuration types are listed in this property, then they must be synonyms for exactly the same remote endpoint that is used to export this service.

The value of this property must be of type `String`, `String[]`, or `Collection` of `String`.

Since: 1.6

See Also: "Remote Services Specification", `SERVICE_EXPORTED_CONFIGS`

### SERVICE_INTENTS

```java
public static final String SERVICE_INTENTS = "service.intents"
```

Service property identifying the intents that this service implement. This property has a dual purpose:

- A bundle can use this service property to notify the distribution provider that these intents are already implemented by the exported service object.
A distribution provider must use this property to convey the combined intents of:
- The exporting service, and
- the intents that the exporting distribution provider adds, and
- the intents that the importing distribution provider adds.

To export a service, a distribution provider must expand any qualified intents. Both the exporting and importing distribution providers must recognize all intents before a service can be distributed.

The value of this property must be of type String, String[], or Collection of String.

Since: 1.6

See Also: "Remote Services Specification"
public interface Filter

An RFC 1960-based Filter.

Filters can be created by calling BundleContext.createFilter() or FrameworkUtil.createFilter() with a filter string.

A Filter can be used numerous times to determine if the match argument matches the filter string that was used to create the Filter.

Some examples of LDAP filters are:

"(cn=Babs Jensen)"
"(!(cn=Tim Howes))"
"&((" + Constants.OBJECTCLASS + "=Person) (| (sn=Jensen) (cn=Babs *))"
"(o=univ*of*mich*)"

Since: 1.1
Version: $Revision: 8299 $
See Also: "Core Specification, section 5.5, for a description of the filter string syntax."
ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(Object obj)</td>
<td>102</td>
</tr>
<tr>
<td>Compares this Filter to another Filter.</td>
<td></td>
</tr>
<tr>
<td>int hashCode()</td>
<td>102</td>
</tr>
<tr>
<td>Returns the hashCode for this Filter.</td>
<td></td>
</tr>
<tr>
<td>boolean match(Dictionary&lt;String, Object&gt; dictionary)</td>
<td>102</td>
</tr>
<tr>
<td>Filter using a Dictionary.</td>
<td></td>
</tr>
<tr>
<td>boolean match(ServiceReference&lt;?&gt; reference)</td>
<td>101</td>
</tr>
<tr>
<td>Filter using a service's properties.</td>
<td></td>
</tr>
<tr>
<td>boolean matchCase(Dictionary&lt;String, Object&gt; dictionary)</td>
<td>103</td>
</tr>
<tr>
<td>Filter with case sensitivity using a Dictionary.</td>
<td></td>
</tr>
<tr>
<td>String toString()</td>
<td>102</td>
</tr>
<tr>
<td>Returns this Filter's filter string.</td>
<td></td>
</tr>
</tbody>
</table>

Method Detail

match

boolean match(ServiceReference<?> reference)

Filter using a service's properties.

This Filter is executed using the keys and values of the referenced service's properties. The keys are case insensitively matched with this Filter.

Parameters:
reference - The reference to the service whose properties are used in the match.
**Interface Filter**

Returns:
true if the service's properties match this Filter; false otherwise.

**match**

boolean `match(Dictionary<String, Object> dictionary)`

Filter using a Dictionary. This Filter is executed using the specified Dictionary's keys and values. The keys are case insensitively matched with this Filter.

Parameters:
- `dictionary` - The Dictionary whose keys are used in the match.

Returns:
true if the Dictionary's keys and values match this filter; false otherwise.

Throws:
- `IllegalArgumentException` - If `dictionary` contains case variants of the same key name.

**toString**

String `toString()`

Returns this Filter's filter string.

The filter string is normalized by removing whitespace which does not affect the meaning of the filter.

Overrides:
- `toString` in class `Object`

Returns:
This Filter's filter string.

**equals**

boolean `equals(Object obj)`

Compares this Filter to another Filter.

This implementation returns the result of calling `this.toString().equals(obj.toString())`.

Overrides:
- `equals` in class `Object`

Parameters:
- `obj` - The object to compare against this Filter.

Returns:
- If the other object is a Filter object, then returns the result of calling `this.toString().equals(obj.toString())`; false otherwise.

**hashCode**

int `hashCode()`

Returns the hashCode for this Filter.

This implementation returns the result of calling `this.toString().hashCode()`.
Overrides:
  hashCode in class Object

Returns:
  The hashCode of this Filter.

matchCase

boolean matchCase(Dictionary<String, Object> dictionary)

Filter with case sensitivity using a Dictionary. This Filter is executed using the specified Dictionary's keys and values. The keys are case sensitively matched with this Filter.

Parameters:
  dictionary - The Dictionary whose keys are used in the match.

Returns:
  true if the Dictionary's keys and values match this filter; false otherwise.

Since:
  1.3
Class FrameworkEvent

org.osgi.framework

java.lang.Object
   java.util.EventObject
      org.osgi.framework.FrameworkEvent

All Implemented Interfaces:
   Serializable

public class FrameworkEvent
   extends EventObject

A general event from the Framework.

FrameworkEvent objects are delivered to FrameworkListeners when a general event occurs within the OSGi environment. A type code is used to identify the event type for future extendability.

OSGi Alliance reserves the right to extend the set of event types.

Version:
   $Revision: 8271 $

See Also:
   FrameworkListener

Immutable

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int ERROR</td>
<td>105</td>
</tr>
<tr>
<td>static int INFO</td>
<td>106</td>
</tr>
<tr>
<td>static int PACKAGES_REFRESHED</td>
<td>105</td>
</tr>
<tr>
<td>static int STARTED</td>
<td>105</td>
</tr>
<tr>
<td>static int STARTLEVEL_CHANGED</td>
<td>105</td>
</tr>
<tr>
<td>static int STOPPED</td>
<td>106</td>
</tr>
<tr>
<td>static int STOPPED_BOOTCLASSPATH_MODIFIED</td>
<td>106</td>
</tr>
<tr>
<td>static int STOPPED_UPDATE</td>
<td>106</td>
</tr>
<tr>
<td>static int WAIT_TIMEOUT</td>
<td>107</td>
</tr>
<tr>
<td>static int WARNING</td>
<td>106</td>
</tr>
</tbody>
</table>

   An error has occurred.

   An informational event has occurred.

   A PackageAdmin.refreshPackage operation has completed.

   The Framework has started.

   A StartLevel.setStartLevel operation has completed.

   The Framework has stopped.

   The Framework has stopped and the boot class path has changed.

   The Framework has stopped during update.

   The Framework did not stop before the wait timeout expired.

   A warning has occurred.

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameworkEvent(int type, Object source)</td>
<td>107</td>
</tr>
<tr>
<td>Deprecated. As of 1.2.</td>
<td></td>
</tr>
<tr>
<td>FrameworkEvent(int type, Bundle bundle, Throwable throwable)</td>
<td>107</td>
</tr>
<tr>
<td>Creates a Framework event regarding the specified bundle.</td>
<td></td>
</tr>
</tbody>
</table>
**Method Summary**

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bundle</td>
<td>getBundle()</td>
<td>Returns the bundle associated with the event.</td>
</tr>
<tr>
<td>Throwable</td>
<td>getThrowable()</td>
<td>Returns the exception related to this event.</td>
</tr>
<tr>
<td>int</td>
<td>getType()</td>
<td>Returns the type of framework event.</td>
</tr>
</tbody>
</table>

**Field Detail**

**STARTED**

```java
public static final int STARTED = 1
```

The Framework has started.

This event is fired when the Framework has started after all installed bundles that are marked to be started have been started and the Framework has reached the initial start level. The source of this event is the System Bundle.

See Also:

"The Start Level Service"

**ERROR**

```java
public static final int ERROR = 2
```

An error has occurred.

There was an error associated with a bundle.

**PACKAGES_REFRESHED**

```java
public static final int PACKAGES_REFRESHED = 4
```

A PackageAdmin.refreshPackage operation has completed.

This event is fired when the Framework has completed the refresh packages operation initiated by a call to the PackageAdmin.refreshPackages method. The source of this event is the System Bundle.

Since:

1.2

See Also:

"PackageAdmin.refreshPackages"

**STARTLEVEL_CHANGED**

```java
public static final int STARTLEVEL_CHANGED = 8
```

A StartLevel.setStartLevel operation has completed.

This event is fired when the Framework has completed changing the active start level initiated by a call to the StartLevel.setStartLevel method. The source of this event is the System Bundle.
WARNING

public static final int WARNING = 16

A warning has occurred.
There was a warning associated with a bundle.

Since:
1.3

INFO

public static final int INFO = 32

An informational event has occurred.
There was an informational event associated with a bundle.

Since:
1.3

STOPPED

public static final int STOPPED = 64

The Framework has stopped.
This event is fired when the Framework has been stopped because of a stop operation on the system bundle. The source of this event is the System Bundle.

Since:
1.5

STOPPED_UPDATE

public static final int STOPPED_UPDATE = 128

The Framework has stopped during update.
This event is fired when the Framework has been stopped because of an update operation on the system bundle. The Framework will be restarted after this event is fired. The source of this event is the System Bundle.

Since:
1.5

STOPPED_BOOTCLASSPATH_MODIFIED

public static final int STOPPED_BOOTCLASSPATH_MODIFIED = 256
Class FrameworkEvent

The Framework has stopped and the boot class path has changed.

This event is fired when the Framework has been stopped because of a stop operation on the system bundle and a bootclasspath extension bundle has been installed or updated. The source of this event is the System Bundle.

Since: 1.5

WAIT_TIMEDOUT

public static final int WAIT_TIMEDOUT = 512

The Framework did not stop before the wait timeout expired.

This event is fired when the Framework did not stop before the wait timeout expired. The source of this event is the System Bundle.

Since: 1.5

Constructor Detail

FrameworkEvent

public FrameworkEvent(int type,
Object source)

Deprecated.

Creates a Framework event.

Parameters:
  type - The event type.
  source - The event source object. This may not be null.

FrameworkEvent

public FrameworkEvent(int type,
Bundle bundle,
Throwable throwable)

Creates a Framework event regarding the specified bundle.

Parameters:
  type - The event type.
  bundle - The event source.
  throwable - The related exception. This argument may be null if there is no related exception.

Method Detail

getThrowable

public Throwable getThrowable()

Returns the exception related to this event.
Class FrameworkEvent

Returns:
The related exception or null if none.

getBundle

public Bundle getBundle()

Returns the bundle associated with the event. This bundle is also the source of the event.

Returns:
The bundle associated with the event.

getType

public int getType()

Returns the type of framework event.

The type values are:

- STARTED
- ERROR
- WARNING
- INFO
- PACKAGES_REFRESHED
- STARTLEVEL_CHANGED
- STOPPED
- STOPPED_BOOTCLASSPATH_MODIFIED
- STOPPED_UPDATE
- WAIT_TIMEDOUT

Returns:
The type of state change.
public interface FrameworkListener
extends EventListener

A FrameworkEvent listener, FrameworkListener is a listener interface that may be implemented by a bundle
developer. When a FrameworkEvent is fired, it is asynchronously delivered to a FrameworkListener. The
Framework delivers FrameworkEvent objects to a FrameworkListener in order and must not concurrently call a
FrameworkListener.

A FrameworkListener object is registered with the Framework using the
BundleContext.addFrameworkListener() method. FrameworkListener objects are called with a
FrameworkEvent objects when the Framework starts and when asynchronous errors occur.

Version: $Revision: 5673 $
See Also: FrameworkEvent
NotThreadSafe

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void frameworkEvent(FrameworkEvent event)</td>
<td>109</td>
</tr>
</tbody>
</table>

Receives notification of a general FrameworkEvent object.

### Method Detail

**frameworkEvent**

void frameworkEvent(FrameworkEvent event)

Receives notification of a general FrameworkEvent object.

**Parameters:**

event - The FrameworkEvent object.
public class FrameworkUtil
extends Object

Framework Utility class.

This class contains utility methods which access Framework functions that may be useful to bundles.

Since:
1.3
Version:
$Revision: 8344 $
ThreadSafe

Method Summary

| static Filter createFilter(String filter) | 110 |
| static Bundle getBundle(Class<?> classFromBundle) | 112 |
| static boolean matchDistinguishedNameChain(String matchPattern, List<String> dnChain) | 111 |

Method Detail

createFilter

public static Filter createFilter(String filter)
    throws InvalidSyntaxException

Creates a Filter object. This Filter object may be used to match a ServiceReference object or a Dictionary object.

If the filter cannot be parsed, an InvalidSyntaxException will be thrown with a human readable message where the filter became unparsable.

This method returns a Filter implementation which may not perform as well as the framework implementation-specific Filter implementation returned by BundleContext.createFilter(String).

Parameters:
filter - The filter string.

Returns:
A Filter object encapsulating the filter string.

Throws:
InvalidSyntaxException - If filter contains an invalid filter string that cannot be parsed.
NullPointerException - If filter is null.

See Also:
Filter
matchDistinguishedNameChain

```java
public static boolean matchDistinguishedNameChain(String matchPattern,
                       List<String> dnChain)

Match a Distinguished Name (DN) chain against a pattern. DNs can be matched using wildcards. A wildcard ("*" or \u002A) replaces all possible values. Due to the structure of the DN, the comparison is more complicated than string-based wildcard matching.

A wildcard can stand for zero or more DNs in a chain, a number of relative distinguished names (RDNs) within a DN, or the value of a single RDN. The DNs in the chain and the matching pattern are canonicalized before processing. This means, among other things, that spaces must be ignored, except in values.

The format of a wildcard match pattern is:

```
matchPattern ::= dn-match (';' dn-match ) *

dn-match ::= ( '*' | rdn-match ) ( ',', rdn-match ) * | '-'

rdsn-match ::= name '=' value-match

value-match ::= '*' | value-star

value-star ::= < value, requires escaped '*' and '-' >
```

The most simple case is a single wildcard; it must match any DN. A wildcard can also replace the first list of RDNs of a DN. The first RDNs are the least significant. Such lists of matched RDNs can be empty.

For example, a match pattern with a wildcard that matches all all DNs that end with RDNs of o=ACME and c=US would look like this:

`*, o=ACME, c=US`

This match pattern would match the following DNs:

```
cn = Bugs Bunny, o = ACME, c = US
ou = Carrots, cn=Daffy Duck, o=ACME, c=US
street = 9C\, Avenue St. Dr\@z\@ry, o=ACME, c=US
dc=www, dc=acme, dc=com, o=ACME, c=US
o=ACME, c=US
```

The following DNs would not match:

```
street = 9C\, Avenue St. Dr\@z\@ry, o=ACME, c=FR
dc=www, dc=acme, dc=com, c=US
```

If a wildcard is used for a value of an RDN, the value must be exactly ". The wildcard must match any value, and no substring matching must be done. For example:

```
cn=*, o=ACME, c=*
```

This match pattern with wildcard must match the following DNs:

```
cn=Bugs Bunny, o=ACME, c=US
cn = Daffy Duck , o = ACME , c = US
cn=Road Runner, o=ACME, c=NL
```

But not:

```
o=ACME, c=NL
dc=acme.com, cn=Bugs Bunny, o=ACME, c=US
```

A match pattern may contain a chain of DN match patterns. The semicolon (";" or \u003B) must be used to separate DN match patterns in a chain. Wildcards can also be used to match against a complete DN within a chain.

The following example matches a certificate signed by Tweety Inc. in the US.

```
* ; ou=S & V, o=Tweety Inc., c=US
```
The wildcard (*) matches zero or one DN in the chain, however, sometimes it is necessary to match a longer chain. The minus sign (-) represents zero or more DNs, whereas the asterisk only represents a single DN. For example, to match a DN where the Tweety Inc. is in the DN chain, use the following expression:

- ; *, o=Tweety Inc., c=US

**Parameters:**
- matchPattern - The pattern against which to match the DN chain.
- dnChain - The DN chain to match against the specified pattern. Each element of the chain must be of type `String` and use the format defined in RFC 2253.

**Returns:**
- true if the pattern matches the DN chain; otherwise false is returned.

**Throws:**
- `IllegalArgumentException` - If the specified match pattern or DN chain is invalid.

**Since:**
- 1.5

---

### getBundle

```java
public static Bundle getBundle(Class<?> classFromBundle)
```

Return a `Bundle` for the specified bundle class. The returned `Bundle` is the bundle associated with the bundle class loader which defined the specified class.

**Parameters:**
- classFromBundle - A class defined by a bundle class loader.

**Returns:**
- A `Bundle` for the specified bundle class or null if the specified class was not defined by a bundle class loader.

**Since:**
- 1.5
Class InvalidSyntaxException

org.osgi.framework

java.lang.Object
  └─java.lang.Throwable
      └─java.lang.Exception
          └─org.osgi.framework.InvalidSyntaxException

All Implemented Interfaces:
  Serializable

public class InvalidSyntaxException
  extends Exception

A Framework exception used to indicate that a filter string has an invalid syntax.

An InvalidSyntaxException object indicates that a filter string parameter has an invalid syntax and cannot be parsed. See Filter for a description of the filter string syntax.

This exception conforms to the general purpose exception chaining mechanism.

Version:
  $Revision: 8344 $

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>113</td>
</tr>
<tr>
<td>InvalidSyntaxException(String msg, String filter)</td>
<td>Creates an exception of type InvalidSyntaxException.</td>
</tr>
<tr>
<td></td>
<td>114</td>
</tr>
<tr>
<td>InvalidSyntaxException(String msg, String filter, Throwable cause)</td>
<td>Creates an exception of type InvalidSyntaxException.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>114</td>
</tr>
<tr>
<td>Throwable get Cause()</td>
<td>Returns the cause of this exception or null if no cause was set.</td>
</tr>
<tr>
<td>String get Filter()</td>
<td>Returns the filter string that generated the InvalidSyntaxException object.</td>
</tr>
<tr>
<td>Throwable init Cause(Throwable cause)</td>
<td>Initializes the cause of this exception to the specified value.</td>
</tr>
</tbody>
</table>

Constructor Detail

InvalidSyntaxException

public InvalidSyntaxException(String msg, String filter)

Creates an exception of type InvalidSyntaxException.

This method creates an InvalidSyntaxException object with the specified message and the filter string which generated the exception.

Parameters:
  msg - The message.
  filter - The invalid filter string.
InvalidSyntaxException

public InvalidSyntaxException(String msg, String filter, Throwable cause)

Creates an exception of type InvalidSyntaxException.

This method creates an InvalidSyntaxException object with the specified message and the filter string which generated the exception.

Parameters:
  - msg: The message.
  - filter: The invalid filter string.
  - cause: The cause of this exception.

Since: 1.3

Method Detail

getFilter

public String getFilter()

Returns the filter string that generated the InvalidSyntaxException object.

Returns: The invalid filter string.

See Also: BundleContext.getServiceReferences(), BundleContext.addServiceListener(ServiceListener, String)

getCause

public Throwable getCause()

Returns the cause of this exception or null if no cause was set.

Overrides: getCause in class Throwable

Returns: The cause of this exception or null if no cause was set.

Since: 1.3

initCause

public Throwable initCause(Throwable cause)

Initializes the cause of this exception to the specified value.

Overrides: initCause in class Throwable

Parameters: cause - The cause of this exception.

Returns: This exception.
Throws:

- `IllegalArgumentException` - If the specified cause is this exception.
- `IllegalStateException` - If the cause of this exception has already been set.

Since:

1.3
### Class PackagePermission

**org.osgi.framework**

```java
java.lang.Object
  └ java.security.Permission
      └ java.security.BasicPermission
          └ org.osgi.framework.PackagePermission
```

All Implemented Interfaces:
- Guard, Serializable

```java
final public class PackagePermission
  extends BasicPermission
```

A bundle's authority to import or export a package.

A package is a dot-separated string that defines a fully qualified Java package.

For example:

```java
org.osgi.service.http
```

PackagePermission has three actions: `exportonly`, `import` and `export`. The export action, which is deprecated, implies the import action.

Version: $Revision: 8338$

ThreadSafe

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td><strong>EXPORT</strong></td>
<td>Deprecated. <em>Since 1.5.</em></td>
<td>117</td>
</tr>
<tr>
<td>static String</td>
<td><strong>EXPORTONLY</strong></td>
<td>The action string <code>exportonly</code>.</td>
<td>117</td>
</tr>
<tr>
<td>static String</td>
<td><strong>IMPORT</strong></td>
<td>The action string <code>import</code>.</td>
<td>117</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Method</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PackagePermission(String name, String actions)</code></td>
<td>Creates a new PackagePermission object.</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td><code>PackagePermission(String name, Bundle exportingBundle, String actions)</code></td>
<td>Creates a new requested PackagePermission object to be used by code that must perform checkPermission for the import action.</td>
<td>118</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean <code>equals(Object obj)</code></td>
<td>Determines the equality of two PackagePermission objects.</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>String <code>getActions()</code></td>
<td>Returns the canonical string representation of the PackagePermission actions.</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>int <code>hashCode()</code></td>
<td>Returns the hash code value for this object.</td>
<td>119</td>
<td></td>
</tr>
</tbody>
</table>
### Class PackagePermission

The `PackagePermission` class is used to represent permissions associated with packages in an OSGi framework. It includes methods to determine if a specified permission is implied by this object and to create a new `PermissionCollection` suitable for storing `PackagePermission` objects.

#### Method Details

**implies(Permission p)**

Determines if the specified permission is implied by this object.

**newPermissionCollection()**

Returns a new `PermissionCollection` object suitable for storing `PackagePermission` objects.

### Field Detail

**EXPORT**

```java
public static final String EXPORT = "export"
```

Deprecated.

The action string `export`. The `export` action implies the `import` action.

**EXPORTONLY**

```java
public static final String EXPORTONLY = "exportonly"
```

The action string `exportonly`. The `exportonly` action does not imply the `import` action.

Since: 1.5

**IMPORT**

```java
public static final String IMPORT = "import"
```

The action string `import`.

### Constructor Detail

**PackagePermission**

```java
public PackagePermission(String name, String actions)
```

Creates a new `PackagePermission` object.

The name is specified as a normal Java package name: a dot-separated string. Wildcards may be used.

`name ::= <package name> | <package name ending in ".*"> | *

Examples:

- org.osgi.service.http
- javax.servlet.*
- *

For the `import` action, the name can also be a filter expression. The filter gives access to the following attributes:

- signer - A Distinguished Name chain used to sign the exporting bundle. Wildcards in a DN are not matched according to the filter string rules, but according to the rules defined for a DN chain.
Filter attribute names are processed in a case sensitive manner.

Package Permissions are granted over all possible versions of a package. A bundle that needs to export a package must have the appropriate PackagePermission for that package; similarly, a bundle that needs to import a package must have the appropriate PackagePermission for that package.

Permission is granted for both classes and resources.

**Parameters:**

- **name** - Package name or filter expression. A filter expression can only be specified if the specified action is import.
- **actions** - exportonly,import (canonical order).

**Throws:**

- IllegalArgumentException - If the specified name is a filter expression and either the specified action is not import or the filter has an invalid syntax.

### PackagePermission

```java
public PackagePermission(String name,
                          Bundle exportingBundle,
                          String actions)
```

Creates a new requested PackagePermission object to be used by code that must perform checkPermission for the import action. PackagePermission objects created with this constructor cannot be added to a PackagePermission permission collection.

**Parameters:**

- **name** - The name of the requested package to import.
- **exportingBundle** - The bundle exporting the requested package.
- **actions** - The action import.

**Throws:**

- IllegalArgumentException - If the specified action is not import or the name is a filter expression.

**Since:**

1.5

### Method Detail

#### implies

```java
public boolean implies(Permission p)
```

Determines if the specified permission is implied by this object.

This method checks that the package name of the target is implied by the package name of this object. The list of PackagePermission actions must either match or allow for the list of the target object to imply the target PackagePermission action.

The permission to export a package implies the permission to import the named package.

\[
x.y.*,"export" \rightarrow x.y.z,"export" \text{ is true}
\]

\[
x.*,"import" \rightarrow x.y, "import" \text{ is true}
\]

\[
x.*,"export" \rightarrow x.y, "import" \text{ is true}
\]

\[
x.y,"export" \rightarrow x.y.z, "export" \text{ is false}
\]
Class PackagePermission

Overrides:
implies in class BasicPermission

Parameters:
p - The requested permission.

Returns:
true if the specified permission is implied by this object; false otherwise.

goingActions

public String getActions()

Returns the canonical string representation of the PackagePermission actions.
Always returns present PackagePermission actions in the following order: EXPORTONLY, IMPORT.

Overrides:
getActions in class BasicPermission

Returns:
Canonical string representation of the PackagePermission actions.

newPermissionCollection

public PermissionCollection newPermissionCollection()

Returns a new PermissionCollection object suitable for storing PackagePermission objects.

Overrides:
newPermissionCollection in class BasicPermission

Returns:
A new PermissionCollection object.

equals

public boolean equals(Object obj)

Determines the equality of two PackagePermission objects. This method checks that specified package has the same package name and PackagePermission actions as this PackagePermission object.

Overrides:
equals in class BasicPermission

Parameters:
obj - The object to test for equality with this PackagePermission object.

Returns:
true if obj is a PackagePermission, and has the same package name and actions as this PackagePermission object; false otherwise.

hashCode

public int hashCode()

Returns the hash code value for this object.

Overrides:
hashCode in class BasicPermission
Returns:
A hash code value for this object.
Class ServiceEvent

org.osgi.framework

java.lang.Object
   └ java.util.EventObject
       └ org.osgi.framework.ServiceEvent

All Implemented Interfaces:
   Serializable

public class ServiceEvent
extends EventObject

An event from the Framework describing a service lifecycle change.

ServiceEvent objects are delivered to ServiceListeners and AllServiceListeners when a change occurs in
this service's lifecycle. A type code is used to identify the event type for future extendability.

OSGi Alliance reserves the right to extend the set of types.

Version:
$Revision: 8096 $

See Also:
   ServiceListener, AllServiceListener

Immutable

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int MODIFIED</td>
<td>The properties of a registered service have been modified.</td>
<td>122</td>
</tr>
<tr>
<td>static int MODIFIED_ENDMATCH</td>
<td>The properties of a registered service have been modified and the new properties no longer match the listener's filter.</td>
<td>122</td>
</tr>
<tr>
<td>static int REGISTERED</td>
<td>This service has been registered.</td>
<td>121</td>
</tr>
<tr>
<td>static int UNREGISTERING</td>
<td>This service is in the process of being unregistered.</td>
<td>122</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceEvent(int type, ServiceReference&lt;?&gt; reference)</td>
<td>Creates a new service event object.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getServiceReference()</td>
<td>Returns a reference to the service that had a change occur in its lifecycle.</td>
</tr>
<tr>
<td>int getType()</td>
<td>Returns the type of event.</td>
</tr>
</tbody>
</table>

Field Detail

REGISTERED

public static final int REGISTERED = 1
This service has been registered.
This event is synchronously delivered after the service has been registered with the Framework.

See Also: BundleContext.registerService(String[], Object, Dictionary)

MODIFIED

public static final int MODIFIED = 2

The properties of a registered service have been modified.
This event is synchronously delivered after the service properties have been modified.

See Also: ServiceRegistration.setProperties()

UNREGISTERING

public static final int UNREGISTERING = 4

This service is in the process of being unregistered.
This event is synchronously delivered before the service has completed unregistering.
If a bundle is using a service that is UNREGISTERING, the bundle should release its use of the service when it receives this event. If the bundle does not release its use of the service when it receives this event, the Framework will automatically release the bundle's use of the service while completing the service unregistration operation.

See Also: ServiceRegistration.unregister(), BundleContext.ungetService()

MODIFIED_ENDMATCH

public static final int MODIFIED_ENDMATCH = 8

The properties of a registered service have been modified and the new properties no longer match the listener's filter.
This event is synchronously delivered after the service properties have been modified. This event is only delivered to listeners which were added with a non-null filter where the filter matched the service properties prior to the modification but the filter does not match the modified service properties.

Since: 1.5
See Also: ServiceRegistration.setProperties()

Constructor Detail

ServiceEvent

public ServiceEvent(int type, ServiceReference<?> reference)
Class ServiceEvent

Creates a new service event object.

**Parameters:**
- `type` - The event type.
- `reference` - A `ServiceReference` object to the service that had a lifecycle change.

### Method Detail

#### getServiceImplReference

```java
public ServiceReference<? super ServiceReference> getServiceReference()
```

Returns a reference to the service that had a change occur in its lifecycle.

This reference is the source of the event.

**Returns:**
Reference to the service that had a lifecycle change.

#### getType

```java
public int getType()
```

Returns the type of event. The event type values are:

- `REGISTERED`
- `MODIFIED`
- `MODIFIED_ENDMATCH`
- `UNREGISTERING`

**Returns:**
Type of service lifecycle change.
public class ServiceException
extends RuntimeException

A service exception used to indicate that a service problem occurred.

A ServiceException object is created by the Framework or service implementation to denote an exception condition in the service. A type code is used to identify the exception type for future extendability. Service implementations may also create subclasses of ServiceException. When subclassing, the subclass should set the type to SUBCLASSED to indicate that ServiceException has been subclassed.

This exception conforms to the general purpose exception chaining mechanism.

Since: 1.5
Version: $Revision: 6518 $

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int FACTORY_ERROR</td>
<td>125</td>
</tr>
<tr>
<td>The service factory produced an invalid service object.</td>
<td></td>
</tr>
<tr>
<td>static int FACTORY_EXCEPTION</td>
<td>125</td>
</tr>
<tr>
<td>The service factory threw an exception.</td>
<td></td>
</tr>
<tr>
<td>static int REMOTE</td>
<td>125</td>
</tr>
<tr>
<td>An error occurred invoking a remote service.</td>
<td></td>
</tr>
<tr>
<td>static int SUBCLASSED</td>
<td>125</td>
</tr>
<tr>
<td>The exception is a subclass of ServiceException.</td>
<td></td>
</tr>
<tr>
<td>static int UNREGISTERED</td>
<td>125</td>
</tr>
<tr>
<td>The service has been unregistered.</td>
<td></td>
</tr>
<tr>
<td>static int UNSPECIFIED</td>
<td>125</td>
</tr>
<tr>
<td>No exception type is unspecified.</td>
<td></td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceException(String msg)</td>
<td>126</td>
</tr>
<tr>
<td>Creates a ServiceException with the specified message.</td>
<td></td>
</tr>
<tr>
<td>ServiceException(String msg, int type)</td>
<td>126</td>
</tr>
<tr>
<td>Creates a ServiceException with the specified message and type.</td>
<td></td>
</tr>
<tr>
<td>ServiceException(String msg, int type, Throwable cause)</td>
<td>126</td>
</tr>
<tr>
<td>Creates a ServiceException with the specified message, type and exception cause.</td>
<td></td>
</tr>
<tr>
<td>ServiceException(String msg, Throwable cause)</td>
<td>126</td>
</tr>
<tr>
<td>Creates a ServiceException with the specified message and exception cause.</td>
<td></td>
</tr>
</tbody>
</table>
## Field Detail

### UNSPECIFIED

public static final int UNSPECIFIED = 0

No exception type is unspecified.

### UNREGISTERED

public static final int UNREGISTERED = 1

The service has been unregistered.

### FACTORY_ERROR

public static final int FACTORY_ERROR = 2

The service factory produced an invalid service object.

### FACTORY_EXCEPTION

public static final int FACTORY_EXCEPTION = 3

The service factory threw an exception.

### SUBCLASSED

public static final int SUBCLASSED = 4

The exception is a subclass of ServiceException. The subclass should be examined for the type of the exception.

### REMOTE

public static final int REMOTE = 5

An error occurred invoking a remote service.
Class ServiceException

Constructor Detail

ServiceException

public ServiceException(String msg,
                        Throwable cause)

Creates a ServiceException with the specified message and exception cause.

Parameters:
  msg - The associated message.
  cause - The cause of this exception.

ServiceException

public ServiceException(String msg)

Creates a ServiceException with the specified message.

Parameters:
  msg - The message.

ServiceException

public ServiceException(String msg,
                        int type,
                        Throwable cause)

Creates a ServiceException with the specified message, type and exception cause.

Parameters:
  msg - The associated message.
  type - The type for this exception.
  cause - The cause of this exception.

ServiceException

public ServiceException(String msg,
                        int type)

Creates a ServiceException with the specified message and type.

Parameters:
  msg - The message.
  type - The type for this exception.

Method Detail

getType

public int getType()

Returns the type for this exception or UNSPECIFIED if the type was unspecified or unknown.
**Returns:**
The type of this exception.
public interface ServiceFactory

Allows services to provide customized service objects in the OSGi environment.

When registering a service, a ServiceFactory object can be used instead of a service object, so that the bundle
developer can gain control of the specific service object granted to a bundle that is using the service.

When this happens, the BundleContext.getService(ServiceReference) method calls the
ServiceFactory.getService method to create a service object specifically for the requesting bundle. The service
object returned by the ServiceFactory is cached by the Framework until the bundle releases its use of the
service.

When the bundle’s use count for the service equals zero (including the bundle stopping or the service being
unregistered), the ServiceFactory.ungetService method is called.

ServiceFactory objects are only used by the Framework and are not made available to other bundles in the OSGi
environment. The Framework may concurrently call a ServiceFactory.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getService(Bundle bundle, ServiceRegistration&lt;S&gt; registration)</td>
<td>128</td>
</tr>
<tr>
<td>ungetService(Bundle bundle, ServiceRegistration&lt;S&gt; registration, S service)</td>
<td>129</td>
</tr>
</tbody>
</table>

Method Detail

gervice

**get Service** (Bundle bundle, ServiceRegistration<S> registration)

Creates a new service object.

The Framework invokes this method the first time the specified bundle requests a service object using the
BundleContext.getService(ServiceReference) method. The service factory can then return a specific
service object for each bundle.

The Framework caches the value returned (unless it is null), and will return the same service object on
any future call to BundleContext.getService for the same bundle. This means the Framework must not
allow this method to be concurrently called for the same bundle.

The Framework will check if the returned service object is an instance of all the classes named when the
service was registered. If not, then null is returned to the bundle.
interface ServiceFactory

Parameters:
  bundle - The bundle using the service.
  registration - The ServiceRegistration object for the service.

Returns:
  A service object that must be an instance of all the classes named when the service was registered.

See Also:
BundleContext.getService()

ungetService

void ungetService(Bundle bundle, ServiceRegistration<S> registration, S service)

Releases a service object.

The Framework invokes this method when a service has been released by a bundle. The service object may then be destroyed.

Parameters:
  bundle - The bundle releasing the service.
  registration - The ServiceRegistration object for the service.
  service - The service object returned by a previous call to the ServiceFactory.getService method.

See Also:
BundleContext.ungetService()
public interface ServiceListener extends EventListener

A ServiceListener is a listener interface that may be implemented by a bundle developer. When a ServiceEvent is fired, it is synchronously delivered to a ServiceListener. The Framework may deliver ServiceEvent objects to a ServiceListener out of order and may concurrently call and/or reenter a ServiceListener.

A ServiceListener object is registered with the Framework using the BundleContext.addServiceListener method. ServiceListener objects are called with a ServiceEvent object when a service is registered, modified, or is in the process of unregistering.

ServiceEvent object delivery to ServiceListener objects is filtered by the filter specified when the listener was registered. If the Java Runtime Environment supports permissions, then additional filtering is done. ServiceEvent objects are only delivered to the listener if the bundle which defines the listener object’s class has the appropriate ServicePermission to get the service using at least one of the named classes under which the service was registered.

ServiceEvent object delivery to ServiceListener objects is further filtered according to package sources as defined in ServiceReference.isAssignableTo(Bundle, String).

Version: $Revision: 5673 $
See Also: ServiceEvent, ServicePermission

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void serviceChanged(ServiceEvent event)</td>
<td>Receives notification that a service has had a lifecycle change.</td>
</tr>
</tbody>
</table>

### Method Detail

**serviceChanged**

void serviceChanged(ServiceEvent event)

Receives notification that a service has had a lifecycle change.

**Parameters:**

- event - The ServiceEvent object.
Class ServicePermission

org.osgi.framework

java.lang.Object
  ▼ java.security.Permission
    ▼ java.security.BasicPermission
      ▼ org.osgi.framework.ServicePermission

All Implemented Interfaces:
  Guard, Serializable

final public class ServicePermission
extends BasicPermission

A bundle's authority to register or get a service.

- The register action allows a bundle to register a service on the specified names.
- The get action allows a bundle to detect a service and get it.

Permission to get a service is required in order to detect events regarding the service. Untrusted bundles should not be able to detect the presence of certain services unless they have the appropriate ServicePermission to get the specific service.

Version:
$Revision: 8338 $
ThreadSafe

Field Summary

| static String GET | The action string get. | 132 |
| static String REGISTER | The action string register. | 132 |

Constructor Summary

| ServicePermission(String name, String actions) | Create a new ServicePermission. | 132 |
| ServicePermission(ServiceReference<?> reference, String actions) | Creates a new requested ServicePermission object to be used by code that must perform checkPermission for the get action. | 133 |

Method Summary

| boolean equals(Object obj) | Determines the equality of two ServicePermission objects. | 134 |
| String getActions() | Returns the canonical string representation of the actions. | 133 |
| int hashCode() | Returns the hash code value for this object. | 134 |
| boolean implies(Permission p) | Determines if a ServicePermission object "implies" the specified permission. | 133 |
| PermissionCollection newPermissionCollection() | Returns a new PermissionCollection object for storing ServicePermission objects. | 133 |
Class ServicePermission

Field Detail

GET

public static final String GET = "get"

The action string get.

REGISTER

public static final String REGISTER = "register"

The action string register.

Constructor Detail

ServicePermission

public ServicePermission(String name,
String actions)

Create a new ServicePermission.

The name of the service is specified as a fully qualified class name. Wildcards may be used.

name ::= <class name> | <class name ending in "."*> | *

Examples:

org.osgi.service.http.HttpService
org.osgi.service.http.*

For the get action, the name can also be a filter expression. The filter gives access to the service properties as well as the following attributes:

- signer - A Distinguished Name chain used to sign the bundle publishing the service. Wildcards in a DN are not matched according to the filter string rules, but according to the rules defined for a DN chain.
- location - The location of the bundle publishing the service.
- id - The bundle ID of the bundle publishing the service.
- name - The symbolic name of the bundle publishing the service.

Since the above attribute names may conflict with service property names used by a service, you can prefix an attribute name with '@' in the filter expression to match against the service property and not one of the above attributes. Filter attribute names are processed in a case sensitive manner unless the attribute references a service property. Service properties names are case insensitive.

There are two possible actions: get and register. The get permission allows the owner of this permission to obtain a service with this name. The register permission allows the bundle to register a service under that name.

Parameters:

name - The service class name
actions - get, register (canonical order)

Throws:

IllegalArgumentException - If the specified name is a filter expression and either the specified action is not get or the filter has an invalid syntax.
Class ServicePermission

ServicePermission

public ServicePermission(ServiceReference<?> reference, String actions)

Creates a new requested ServicePermission object to be used by code that must perform checkPermission for the get action. ServicePermission objects created with this constructor cannot be added to a ServicePermission permission collection.

Parameters:
- reference - The requested service.
- actions - The action get.

Throws:
- IllegalArgumentException - If the specified action is not get or reference is null.

Since: 1.5

Method Detail

implies

public boolean implies(Permission p)

Determines if a ServicePermission object "implies" the specified permission.

Overrides:
- implies in class BasicPermission

Parameters:
- p - The target permission to check.

Returns:
- true if the specified permission is implied by this object; false otherwise.

getActions

public String getActions()

Returns the canonical string representation of the actions. Always returns present actions in the following order: get, register.

Overrides:
- getActions in class BasicPermission

Returns:
- The canonical string representation of the actions.

newPermissionCollection

public PermissionCollection newPermissionCollection()

Returns a new PermissionCollection object for storing ServicePermission objects.

Overrides:
- newPermissionCollection in class BasicPermission

Returns:
- A new PermissionCollection object suitable for storing ServicePermission objects.
equals

public boolean equals(Object obj)

Determines the equality of two ServicePermission objects. Checks that specified object has the same class name and action as this ServicePermission.

**Overrides:**
equals in class BasicPermission

**Parameters:**
obj - The object to test for equality.

**Returns:**
true if obj is a ServicePermission, and has the same class name and actions as this ServicePermission object; false otherwise.

hashCode

public int hashCode()

Returns the hash code value for this object.

**Overrides:**
hashCode in class BasicPermission

**Returns:**
Hash code value for this object.
public interface ServiceReference
extends Comparable<Object>

A reference to a service.

The Framework returns ServiceReference objects from the BundleContext.getServiceReference and BundleContext.getServiceReferences methods.

A ServiceReference object may be shared between bundles and can be used to examine the properties of the service and to get the service object.

Every service registered in the Framework has a unique ServiceRegistration object and may have multiple, distinct ServiceReference objects referring to it. ServiceReference objects associated with a ServiceRegistration object have the same hashCode and are considered equal (more specifically, their equals() method will return true when compared).

If the same service object is registered multiple times, ServiceReference objects associated with different ServiceRegistration objects are not equal.

Version:
$Revision: 8163 $

See Also:
BundleContext.getServiceReference(), BundleContext.getServiceReferences(), BundleContext.getService()

ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int compareTo (Object reference)</td>
<td>137</td>
</tr>
<tr>
<td>Bundle getBundle ()</td>
<td>136</td>
</tr>
<tr>
<td>Object getProperty (String key)</td>
<td>136</td>
</tr>
<tr>
<td>String[] getPropertyKeys ()</td>
<td>136</td>
</tr>
<tr>
<td>Bundle[] getUsingBundles ()</td>
<td>136</td>
</tr>
<tr>
<td>boolean isAssignableTo (Bundle bundle, String className)</td>
<td>137</td>
</tr>
</tbody>
</table>
Method Detail

getProperty

Object getProperty(String key)

Returns the property value to which the specified property key is mapped in the properties Dictionary object of the service referenced by this ServiceReference object.

Property keys are case-insensitive.

This method must continue to return property values after the service has been unregistered. This is so references to unregistered services (for example, ServiceReference objects stored in the log) can still be interrogated.

Parameters:
key - The property key.

Returns:
The property value to which the key is mapped; null if there is no property named after the key.

getPropertyKeys

String[] getPropertyKeys()

Returns an array of the keys in the properties Dictionary object of the service referenced by this ServiceReference object.

This method will continue to return the keys after the service has been unregistered. This is so references to unregistered services (for example, ServiceReference objects stored in the log) can still be interrogated.

This method is case-preserving; this means that every key in the returned array must have the same case as the corresponding key in the properties Dictionary that was passed to the BundleContext.registerService(String[], Object, Dictionary) or ServiceRegistration.setProperties() methods.

Returns:
An array of property keys.

getBundle

Bundle getBundle()

Returns the bundle that registered the service referenced by this ServiceReference object.

This method must return null when the service has been unregistered. This can be used to determine if the service has been unregistered.

Returns:
The bundle that registered the service referenced by this ServiceReference object; null if that service has already been unregistered.

See Also:
BundleContext.registerService(String[], Object, Dictionary)

getUsingBundles

Bundle[] getUsingBundles()


Returns the bundles that are using the service referenced by this `ServiceReference` object. Specifically, this method returns the bundles whose usage count for that service is greater than zero.

**Returns:**
An array of bundles whose usage count for the service referenced by this `ServiceReference` object is greater than zero; `null` if no bundles are currently using that service.

**Since:**
1.1

### isAssignableTo

```java
boolean isAssignableTo(Bundle bundle, String className)
```

Tests if the bundle that registered the service referenced by this `ServiceReference` and the specified bundle use the same source for the package of the specified class name.

This method performs the following checks:

1. Get the package name from the specified class name.
2. For the bundle that registered the service referenced by this `ServiceReference` (registrant bundle); find the source for the package. If no source is found then return `true` if the registrant bundle is equal to the specified bundle; otherwise return `false`.
3. If the package source of the registrant bundle is equal to the package source of the specified bundle then return `true`; otherwise return `false`.

**Parameters:**
- `bundle` - The `Bundle` object to check.
- `className` - The class name to check.

**Returns:**
- `true` if the bundle which registered the service referenced by this `ServiceReference` and the specified bundle use the same source for the package of the specified class name. Otherwise `false` is returned.

**Throws:**
- `IllegalArgumentException` - If the specified `Bundle` was not created by the same framework instance as this `ServiceReference`.

**Since:**
1.3

### compareTo

```java
int compareTo(Object reference)
```

Compares this `ServiceReference` with the specified `ServiceReference` for order.

If this `ServiceReference` and the specified `ServiceReference` have the same `service_id` they are equal. This `ServiceReference` is less than the specified `ServiceReference` if it has a lower `service_ranking` and greater if it has a higher service ranking. Otherwise, if this `ServiceReference` and the specified `ServiceReference` have the same `service_ranking`, this `ServiceReference` is less than the specified `ServiceReference` if it has a higher `service_id` and greater if it has a lower service id.

**Specified by:**
`compareTo` in interface `Comparable`

**Parameters:**
- `reference` - The `ServiceReference` to be compared.

**Returns:**
- Returns a negative integer, zero, or a positive integer if this `ServiceReference` is less than, equal to, or greater than the specified `ServiceReference`.
Throws:

IllegalArgumentException - If the specified ServiceReference was not created by the same framework instance as this ServiceReference.

Since:

1.4
Interface ServiceRegistration

org.osgi.framework

Type Parameters:
S - Type of Service.

public interface ServiceRegistration

A registered service.

The Framework returns a ServiceRegistration object when a BundleContext.registerService method invocation is successful. The ServiceRegistration object is for the private use of the registering bundle and should not be shared with other bundles.

The ServiceRegistration object may be used to update the properties of the service or to unregister the service.

Version: $Revision: 8096 $
See Also: BundleContext.registerService(String[],Object,Dictionary)

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceReference&lt;S&gt; getReference()</td>
<td>139</td>
</tr>
<tr>
<td>Returns a ServiceReference object for a service being registered.</td>
<td></td>
</tr>
<tr>
<td>void setProperties(Dictionary&lt;String,Object&gt; properties)</td>
<td>139</td>
</tr>
<tr>
<td>Updates the properties associated with a service.</td>
<td></td>
</tr>
<tr>
<td>void unregister()</td>
<td>140</td>
</tr>
<tr>
<td>Unregisters a service.</td>
<td></td>
</tr>
</tbody>
</table>

Method Detail

getAddress

ServiceReference<S> getReference()

Returns a ServiceReference object for a service being registered.

The ServiceReference object may be shared with other bundles.

Returns: ServiceReference object.

Throws: IllegalStateException - If this ServiceRegistration object has already been unregistered.

setProperties

void setProperties(Dictionary<String, Object> properties)

Updates the properties associated with a service.

The Constants.OBJECTCLASS and Constants.SERVICE_ID keys cannot be modified by this method.
These values are set by the Framework when the service is registered in the OSGi environment.
The following steps are required to modify service properties:

1. The service's properties are replaced with the provided properties.
2. A service event of type `ServiceEvent.MODIFIED` is fired.

**Parameters:**

- `properties` - The properties for this service. See Constants for a list of standard service property keys. Changes should not be made to this object after calling this method. To update the service's properties this method should be called again.

**Throws:**

- `IllegalStateException` - If this `ServiceRegistration` object has already been unregistered.
- `IllegalArgumentException` - If `properties` contains case variants of the same key name.

---

**unregister**

```java
void unregister()
```

Unregisters a service. Remove a `ServiceRegistration` object from the Framework service registry. All `ServiceReference` objects associated with this `ServiceRegistration` object can no longer be used to interact with the service once unregistration is complete.

The following steps are required to unregister a service:

1. The service is removed from the Framework service registry so that it can no longer be obtained.
2. A service event of type `ServiceEvent.UNREGISTERING` is fired so that bundles using this service can release their use of the service. Once delivery of the service event is complete, the `ServiceReference` objects for the service may no longer be used to get a service object for the service.
3. For each bundle whose use count for this service is greater than zero:
   - The bundle's use count for this service is set to zero.
   - If the service was registered with a `ServiceFactory` object, the `ServiceFactory.ungetService` method is called to release the service object for the bundle.

**Throws:**

- `IllegalStateException` - If this `ServiceRegistration` object has already been unregistered.

**See Also:**

`BundleContext.ungetService()`, `ServiceFactory.ungetService()`
Interface SynchronousBundleListener

org.osgi.framework

All Superinterfaces:
  BundleListener, EventListener

public interface SynchronousBundleListener
extends BundleListener

A synchronous BundleEvent listener. SynchronousBundleListener is a listener interface that may be implemented by a bundle developer. When a BundleEvent is fired, it is synchronously delivered to a SynchronousBundleListener. The Framework may deliver BundleEvent objects to a SynchronousBundleListener out of order and may concurrently call and/or reenter a SynchronousBundleListener.

A SynchronousBundleListener object is registered with the Framework using the BundleContext.addBundleListener() method. SynchronousBundleListener objects are called with a BundleEvent object when a bundle has been installed, resolved, starting, started, stopping, stopped, updated, unresolved, or uninstalled.

Unlike normal BundleListener objects, SynchronousBundleListener objects are synchronously called during bundle lifecycle processing. The bundle lifecycle processing will not proceed until all SynchronousBundleListener objects have completed. SynchronousBundleListener objects will be called prior to BundleListener objects.

AdminPermission[bundle,LISTENER] is required to add or remove a SynchronousBundleListener object.

Since: 1.1
Version: $Revision: 5673 $
See Also: BundleEvent
ThreadSafe

Methods inherited from interface org.osgi.framework.BundleListener

bundleChanged
public class Version
extends Object
implements Comparable<Version>

Version identifier for bundles and packages.

Version identifiers have four components.

4. Qualifier. A text string. See Version(String) for the format of the qualifier string.

Version objects are immutable.

Since: 1.3
Version: $Revision: 8271 $
Immutable
Class Version

### String

#### getQualifier()

Returns the qualifier component of this version identifier.

#### int

#### hashCode()

Returns a hash code value for the object.

### static

#### Version

#### parseVersion(String version)

Parses a version identifier from the specified string.

### String

#### toString()

Returns the string representation of this version identifier.

---

#### Field Detail

**emptyVersion**

public static final Version emptyVersion

The empty version "0.0.0".

---

#### Constructor Detail

**Version**

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

Creates a version identifier from the specified numerical components.

The qualifier is set to the empty string.

**Parameters:**

- major - Major component of the version identifier.
- minor - Minor component of the version identifier.
- micro - Micro component of the version identifier.

**Throws:**

- IllegalArgumentException - If the numerical components are negative.

---

**Version**

public Version(int major,
   int minor,
   int micro,
   String qualifier)

Creates a version identifier from the specified components.

**Parameters:**

- major - Major component of the version identifier.
- minor - Minor component of the version identifier.
- micro - Micro component of the version identifier.
- qualifier - Qualifier component of the version identifier. If null is specified, then the qualifier will be set to the empty string.

**Throws:**

- IllegalArgumentException - If the numerical components are negative or the qualifier string is invalid.
Version

public Version(String version)

Created a version identifier from the specified string.

Here is the grammar for version strings.

version ::= major('.'minor('.'micro('.'qualifier?)?)?)
major ::= digit+
minor ::= digit+
micro ::= digit+
qualifier ::= (alpha|digit|'|'_'|'-')+
digit ::= [0..9]
alpha ::= [a..zA..Z]

There must be no whitespace in version.

Parameters:
version - String representation of the version identifier.

Throws:
IllegalArgumentException - If version is improperly formatted.

Method Detail

parseVersion

public static Version parseVersion(String version)

Parses a version identifier from the specified string.

See Version(String) for the format of the version string.

Parameters:
version - String representation of the version identifier. Leading and trailing whitespace will be ignored.

Returns:
A Version object representing the version identifier. If version is null or the empty string then emptyVersion will be returned.

Throws:
IllegalArgumentException - If version is improperly formatted.

getMajor

public int getMajor()

Returns the major component of this version identifier.

Returns:
The major component.

getMinor

public int getMinor()

Returns the minor component of this version identifier.

Returns:
The minor component.
getMicro

```java
public int getMicro()

    Returns the micro component of this version identifier.

    Returns: The micro component.
```

getQualifier

```java
public String getQualifier()

    Returns the qualifier component of this version identifier.

    Returns: The qualifier component.
```

toString

```java
public String toString()

    Returns the string representation of this version identifier.

    The format of the version string will be major.minor.micro if qualifier is the empty string or
    major.minor.micro.qualifier otherwise.

    Overrides: toString in class Object
    Returns: The string representation of this version identifier.
```

hashCode

```java
public int hashCode()

    Returns a hash code value for the object.

    Overrides: hashCode in class Object
    Returns: An integer which is a hash code value for this object.
```

equals

```java
public boolean equals(Object object)

    Compares this Version object to another object.

    A version is considered to be equal to another version if the major, minor and micro components are equal
    and the qualifier component is equal (using String.equals).

    Overrides: equals in class Object
```

Class Version

**Parameters:**
- object - The Version object to be compared.

**Returns:**
- true if object is a Version and is equal to this object; false otherwise.

**compareTo**

```java
public int compareTo(Version other)
```

Compares this Version object to another object.

A version is considered to be **less than** another version if its major component is less than the other version's major component, or the major components are equal and its minor component is less than the other version's minor component, or the major and minor components are equal and its micro component is less than the other version's micro component, or the major, minor and micro components are equal and it's qualifier component is less than the other version's qualifier component (using `String.compareTo`).

A version is considered to be **equal to** another version if the major, minor and micro components are equal and the qualifier component is equal (using `String.compareTo`).

**Specified by:**
- compareTo in interface Comparable

**Parameters:**
- other - The Version object to be compared.

**Returns:**
- A negative integer, zero, or a positive integer if this object is less than, equal to, or greater than the specified Version object.

**Throws:**
- ClassCastException - If the specified object is not a Version.
Framework Service Hooks Package Version 1.0.

See: Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHook</td>
<td>148</td>
</tr>
<tr>
<td>FindHook</td>
<td>149</td>
</tr>
<tr>
<td>ListenerHook</td>
<td>150</td>
</tr>
<tr>
<td>ListenerHook.ListenerInfo</td>
<td>151</td>
</tr>
</tbody>
</table>

Package org.osgi.framework.hooks.service Description

Framework Service Hooks Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. For example:

```
Import-Package: org.osgi.framework.hooks.service;version="[1.0,2.0)"
```
public interface EventHook

OSGi Framework Service Event Hook Service.

Bundles registering this service will be called during framework service (register, modify, and unregister service) operations.

Version:

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>event(ServiceEvent event, Collection(BundleContext) contexts)</td>
<td>148</td>
</tr>
</tbody>
</table>

Method Detail

event

void event(ServiceEvent event, 
Collection(BundleContext) contexts)

Event hook method. This method is called prior to service event delivery when a publishing bundle registers, modifies or unregisters a service. This method can filter the bundles which receive the event.

Parameters:

event - The service event to be delivered.
contexts - A Collection of Bundle Contexts for bundles which have listeners to which the specified event will be delivered. The implementation of this method may remove bundle contexts from the collection to prevent the event from being delivered to the associated bundles. The collection supports all the optional Collection operations except add and addAll. Attempting to add to the collection will result in an UnsupportedOperationException. The collection is not synchronized.
public interface FindHook

OSGi Framework Service Find Hook Service.

Bundles registering this service will be called during framework service find (get service references) operations.

Version:
$Revision: 8096 $

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void find(BundleContext context, String name, String filter, boolean allServices, Collection&lt;ServiceReference&lt;?&gt;&gt; references)</td>
</tr>
</tbody>
</table>

Find hook method.

Method Detail

find

void find(BundleContext context, String name, String filter, boolean allServices, Collection<ServiceReference<?>> references)

Find hook method. This method is called during the service find operation (for example, BundleContext.getServiceReferences(String, String)). This method can filter the result of the find operation.

Parameters:
- context - The bundle context of the bundle performing the find operation.
- name - The class name of the services to find or null to find all services.
- filter - The filter criteria of the services to find or null for no filter criteria.
- allServices - true if the find operation is the result of a call to BundleContext.getAllServiceReferences(String, String)
- references - A Collection of Service References to be returned as a result of the find operation. The implementation of this method may remove service references from the collection to prevent the references from being returned to the bundle performing the find operation. The collection supports all the optional Collection operations except add and addAll. Attempting to add to the collection will result in an UnsupportedOperationException. The collection is not synchronized.
public interface ListenerHook

OSGi Framework Service Listener Hook Service.

Bundles registering this service will be called during service listener addition and removal.

Version:
$Revision: 8093 $
ThreadSafe

### Nested Class Summary

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static interface ListenerHook.ListenerInfo</td>
<td>151</td>
</tr>
<tr>
<td>Information about a Service Listener.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void added(Collection&lt;ListenerHook.ListenerInfo&gt; listeners)</td>
<td>150</td>
</tr>
<tr>
<td>Added listeners hook method.</td>
<td></td>
</tr>
<tr>
<td>void removed(Collection&lt;ListenerHook.ListenerInfo&gt; listeners)</td>
<td>150</td>
</tr>
<tr>
<td>Removed listeners hook method.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Detail

**added**

```java
void added(Collection<ListenerHook.ListenerInfo> listeners)
```

Added listeners hook method. This method is called to provide the hook implementation with information on newly added service listeners. This method will be called as service listeners are added while this hook is registered. Also, immediately after registration of this hook, this method will be called to provide the current collection of service listeners which had been added prior to the hook being registered.

**Parameters:**

- `listeners` - A Collection of `ListenerHook.ListenerInfo`s for newly added service listeners which are now listening to service events. Attempting to add to or remove from the collection will result in an `UnsupportedOperationException`. The collection is not synchronized.

**removed**

```java
void removed(Collection<ListenerHook.ListenerInfo> listeners)
```

Removed listeners hook method. This method is called to provide the hook implementation with information on newly removed service listeners. This method will be called as service listeners are removed while this hook is registered.

**Parameters:**

- `listeners` - A Collection of `ListenerHook.ListenerInfo`s for newly removed service listeners which are no longer listening to service events. Attempting to add to or remove from the collection will result in an `UnsupportedOperationException`. The collection is not synchronized.
public static interface ListenerHook.ListenerInfo

Information about a Service Listener. This interface describes the bundle which added the Service Listener and the filter with which it was added.

ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(Object obj)</td>
<td>152</td>
</tr>
<tr>
<td>BundleContext getBundleContext()</td>
<td>151</td>
</tr>
<tr>
<td>String getFilter()</td>
<td>151</td>
</tr>
<tr>
<td>int hashCode()</td>
<td>152</td>
</tr>
<tr>
<td>boolean isRemoved()</td>
<td>151</td>
</tr>
</tbody>
</table>

**Method Detail**

**getBundleContext**

BundleContext getBundleContext()

Return the context of the bundle which added the listener.

**Returns:**

The context of the bundle which added the listener.

**getFilter**

String getFilter()

Return the filter string with which the listener was added.

**Returns:**

The filter string with which the listener was added. This may be null if the listener was added without a filter.

**isRemoved**

boolean isRemoved()
Return the state of the listener for this addition and removal life cycle. Initially this method will return \texttt{false} indicating the listener has been added but has not been removed. After the listener has been removed, this method must always return \texttt{true}.

There is an extremely rare case in which removed notification to \texttt{ListenerHook}S can be made before added notification if two threads are racing to add and remove the same service listener. Because \texttt{ListenerHook}S are called synchronously during service listener addition and removal, the Framework cannot guarantee in-order delivery of added and removed notification for a given service listener. This method can be used to detect this rare occurrence.

**Returns:**
\texttt{false} if the listener has not been been removed, \texttt{true} otherwise.

### equals

\begin{verbatim}
boolean equals(Object obj)
\end{verbatim}

Compares this \texttt{ListenerInfo} to another \texttt{ListenerInfo}. Two \texttt{ListenerInfo}S are equals if they refer to the same listener for a given addition and removal life cycle. If the same listener is added again, it must have a different \texttt{ListenerInfo} which is not equal to this \texttt{ListenerInfo}.

**Overrides:**
\texttt{equals} in class \texttt{Object}

**Parameters:**
\texttt{obj} - The object to compare against this \texttt{ListenerInfo}.

**Returns:**
\texttt{true} if the other object is a \texttt{ListenerInfo} object and both objects refer to the same listener for a given addition and removal life cycle.

### hashCode

\begin{verbatim}
int hashCode()
\end{verbatim}

Returns the hash code for this \texttt{ListenerInfo}.

**Overrides:**
\texttt{hashCode} in class \texttt{Object}

**Returns:**
The hash code of this \texttt{ListenerInfo}. 

Package org.osgi.framework.launch

Framework Launch Package Version 1.0.

See: Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Framework</strong></td>
<td>154</td>
</tr>
<tr>
<td>A Framework instance.</td>
<td></td>
</tr>
<tr>
<td><strong>FrameworkFactory</strong></td>
<td>161</td>
</tr>
<tr>
<td>A factory for creating Framework instances.</td>
<td></td>
</tr>
</tbody>
</table>

Package org.osgi.framework.launch Description

Framework Launch Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. For example:

Import-Package: org.osgi.framework.launch;version="[1.0,2.0)"
Interface Framework

```
org.osgi.framework.launch
```

**All Superinterfaces:** Bundle, Comparable&lt;Bundle&gt;  

```java
public interface Framework
extends Bundle
```

A Framework instance. A Framework is also known as a System Bundle.

Framework instances are created using a FrameworkFactory. The methods of this interface can be used to manage and control the created framework instance.

**Version:** $Revision: 8906 $

**ThreadSafe**

### Fields inherited from interface org.osgi.framework.Bundle

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVE, INSTALLED, RESOLVED, SIGNERS_ALL, SIGNERS_TRUSTED, START_ACTIVATION_POLICY, START_TRANSIENT, STARTING, STOP_TRANSIENT, STOPPING, TYPE_FRAGMENT, UNINSTALLED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
</table>
| Enumeratio
n&lt;URL&gt; findEntries(String path, String filePattern, boolean recurse) | Returns null as a framework implementation does not have a proper bundle from which to return entries. | 160 |
| long getBundleId() | Returns the Framework unique identifier. | 159 |
| URL getEntry(String path) | Returns null as a framework implementation does not have a proper bundle from which to return an entry. | 160 |
| Enumeratio
n&lt;String&gt; getEntryPaths(String path) | Returns null as a framework implementation does not have a proper bundle from which to return entry paths. | 159 |
| String getLocation() | Returns the Framework location identifier. | 159 |
| String getSymbolicName() | Returns the symbolic name of this Framework. | 159 |
| void init() | Initialize this Framework. | 155 |
| void start() | Start this Framework. | 156 |
| void start(int options) | Start this Framework. | 156 |
| void stop() | Stop this Framework. | 157 |
| void stop(int options) | Stop this Framework. | 157 |
| void uninstall() | The Framework cannot be uninstalled. | 158 |
| void update() | Stop and restart this Framework. | 158 |
**Interface Framework**

```
void update(InputStream in)
Stop and restart this Framework.
```

```
FrameworkEvent waitForStop(long timeout)
Wait until this Framework has completely stopped.
```

### Methods inherited from interface org.osgi.framework.Bundle

- `adapt`<br>`getBundleContext`<br>`getHeaders`<br>`getHeaders`<br>`getLastModified`<br>`getRegisteredServices`<br>`getResource`<br>`getResources`<br>`getServicesInUse`<br>`getSignerCertificates`<br>`getState`<br>`getTypes`<br>`getVersion`<br>`hasPermission`<br>`loadClass`

### Method Detail

#### init

```
void init()
throws BundleException
```

Initialize this Framework. After calling this method, this Framework must:

- Be in the `Bundle.STARTING` state.
- Have a valid Bundle Context.
- Be at start level 0.
- Have event handling enabled.
- Have reified Bundle objects for all installed bundles.
- Have registered any framework services. For example, PackageAdmin, ConditionalPermissionAdmin, StartLevel.

This Framework will not actually be started until `start` is called.

This method does nothing if called when this Framework is in the `Bundle.STARTING`, `Bundle.ACTIVE` or `Bundle.STOPPING` states.

**Throws:**

- `BundleException` - If this Framework could not be initialized.
- `SecurityException` - If the Java Runtime Environment supports permissions and the caller does not have the appropriate `AdminPermission[this,EXECUTE]` or if there is a security manager already installed and the `Constants.FRAMEWORK_SECURITY` configuration property is set.

#### waitForStop

```
FrameworkEvent waitForStop(long timeout)
throws InterruptedException
```

Wait until this Framework has completely stopped. The `stop` and `update` methods on a Framework performs an asynchronous stop of the Framework. This method can be used to wait until the asynchronous stop of this Framework has completed. This method will only wait if called when this Framework is in the `Bundle.STARTING`, `Bundle.ACTIVE`, or `Bundle.STOPPING` states. Otherwise it will return immediately.

A Framework Event is returned to indicate why this Framework has stopped.

**Parameters:**

- `timeout` - Maximum number of milliseconds to wait until this Framework has completely stopped. A value of zero will wait indefinitely.

**Returns:**

A Framework Event indicating the reason this method returned. The following `FrameworkEvent` types may be returned by this method.

- `STOPPED` - This Framework has been stopped.
- **STOPPED UPDATE** - This Framework has been updated which has shutdown and will now restart.
- **STOPPED_BOOTCLASSPATH_MODIFIED** - This Framework has been stopped and a bootclasspath extension bundle has been installed or updated. The VM must be restarted in order for the changed boot class path to take affect.
- **ERROR** - The Framework encountered an error while shutting down or an error has occurred which forced the framework to shutdown.
- **WAIT_TIMEDOUT** - This method has timed out and returned before this Framework has stopped.

**Throws:**

- `InterruptedException` - If another thread interrupted the current thread before or while the current thread was waiting for this Framework to completely stop. The `interrupted status` of the current thread is cleared when this exception is thrown.
- `IllegalArgumentException` - If the value of timeout is negative.

```java
void start()
throws BundleException
```

Start this Framework.

The following steps are taken to start this Framework:

1. If this Framework is not in the `Bundle.STARTING` state, `initialize` this Framework.
2. All installed bundles must be started in accordance with each bundle's persistent `autostart setting`. This means some bundles will not be started, some will be started with `eager activation` and some will be started with their `declared activation policy`. If this Framework implements the optional `Start Level Service Specification`, then the start level of this Framework is moved to the start level specified by the `beginning start level` framework property, as described in the `Start Level Service Specification`. If this framework property is not specified, then the start level of this Framework is moved to start level one (1). Any exceptions that occur during bundle starting must be wrapped in a `BundleException` and then published as a framework event of type `FrameworkEvent.ERROR`.
3. This Framework's state is set to `Bundle.ACTIVE`.
4. A framework event of type `FrameworkEvent.STARTED` is fired.

**Specified by:**

`start` in interface `Bundle`

**Throws:**

- `BundleException` - If this Framework could not be started.
- `SecurityException` - If the caller does not have the appropriate `AdminPermission[this,EXECUTE]`, and the Java Runtime Environment supports permissions.

**See Also:**

"Start Level Service Specification"

```java
void start(int options)
throws BundleException
```

Start this Framework.

Calling this method is the same as calling `start()`. There are no start options for the Framework.

**Specified by:**

`start` in interface `Bundle`

**Parameters:**

- `options` - Ignored. There are no start options for the Framework.
Throws:

- BundleException - If this Framework could not be started.
- SecurityException - If the caller does not have the appropriate AdminPermission[this, EXECUTE], and the Java Runtime Environment supports permissions.

See Also:

- `start()`

---

**stop**

```java
void stop()
  throws BundleException
```

Stop this Framework.

The method returns immediately to the caller after initiating the following steps to be taken on another thread.

1. This Framework's state is set to Bundle.STOPPING.
2. All installed bundles must be stopped without changing each bundle's persistent autostart setting. If this Framework implements the optional Start Level Service Specification, then the start level of this Framework is moved to start level zero (0), as described in the Start Level Service Specification. Any exceptions that occur during bundle stopping must be wrapped in a BundleException and then published as a framework event of type FrameworkEvent.ERROR
3. Unregister all services registered by this Framework.
4. Event handling is disabled.
5. This Framework's state is set to Bundle.RESOLVED.
6. All resources held by this Framework are released. This includes threads, bundle class loaders, open files, etc.
7. Notify all threads that are waiting at waitForStop that the stop operation has completed.

After being stopped, this Framework may be discarded, initialized or started.

Specified by:

- `stop` in interface Bundle

Throws:

- BundleException - If stopping this Framework could not be initiated.
- SecurityException - If the caller does not have the appropriate AdminPermission[this, EXECUTE], and the Java Runtime Environment supports permissions.

See Also:

- "Start Level Service Specification"

---

**stop**

```java
void stop(int options)
  throws BundleException
```

Stop this Framework.

Calling this method is the same as calling `stop()`. There are no stop options for the Framework.

Specified by:

- `stop` in interface Bundle

Parameters:

- options - Ignored. There are no stop options for the Framework.

Throws:

- BundleException - If stopping this Framework could not be initiated.
- SecurityException - If the caller does not have the appropriate AdminPermission[this, EXECUTE], and the Java Runtime Environment supports permissions.

See Also:

- `stop()`
### uninstall

```java
void uninstall()
    throws BundleException
```

The Framework cannot be uninstalled.

This method always throws a BundleException.

**Specified by:**
uninstall in interface Bundle

**Throws:**
- BundleException - This Framework cannot be uninstalled.
- SecurityException - If the caller does not have the appropriate AdminPermission[this,LIFECYCLE], and the Java Runtime Environment supports permissions.

### update

```java
void update()
    throws BundleException
```

Stop and restart this Framework.

The method returns immediately to the caller after initiating the following steps to be taken on another thread.

1. Perform the steps in the `stop()` method to stop this Framework.
2. Perform the steps in the `start()` method to start this Framework.

**Specified by:**
update in interface Bundle

**Throws:**
- BundleException - If stopping and restarting this Framework could not be initiated.
- SecurityException - If the caller does not have the appropriate AdminPermission[this,LIFECYCLE], and the Java Runtime Environment supports permissions.

### update

```java
void update(InputStream in)
    throws BundleException
```

Stop and restart this Framework.

Calling this method is the same as calling `update()` except that any provided InputStream is immediately closed.

**Specified by:**
update in interface Bundle

**Parameters:**
in - Any provided InputStream is immediately closed before returning from this method and otherwise ignored.

**Throws:**
- BundleException - If stopping and restarting this Framework could not be initiated.
- SecurityException - If the caller does not have the appropriate AdminPermission[this,LIFECYCLE], and the Java Runtime Environment supports permissions.
getBundleId

long getBundleId()

Returns the Framework unique identifier. This Framework is assigned the unique identifier zero (0) since this Framework is also a System Bundle.

Specified by: getBundleId in interface Bundle
Returns: 0.
See Also: Bundle.getBundleId()

getLocation

String getLocation()

Returns the Framework location identifier. This Framework is assigned the unique location "System Bundle" since this Framework is also a System Bundle.

Specified by: getLocation in interface Bundle
Returns: The string "System Bundle".
Throws: SecurityException - If the caller does not have the appropriate AdminPermission[this,METADATA], and the Java Runtime Environment supports permissions.
See Also: Bundle.getLocation(), Constants.SYSTEM_BUNDLE_LOCATION

getSymbolicName

String getSymbolicName()

Returns the symbolic name of this Framework. The symbolic name is unique for the implementation of the framework. However, the symbolic name "system.bundle" must be recognized as an alias to the implementation-defined symbolic name since this Framework is also a System Bundle.

Specified by: getSymbolicName in interface Bundle
Returns: The symbolic name of this Framework.
See Also: Bundle.getSymbolicName(), Constants.SYSTEM_BUNDLE_SYMBOLICNAME

g ENTRY PATH S

Enumeration<String> getEntryPaths(String path)

Returns null as a framework implementation does not have a proper bundle from which to return entry paths.

Specified by: getEntryPaths in interface Bundle
Parameters: path - Ignored.
Interface Framework

Returns:
null as a framework implementation does not have a proper bundle from which to return entry paths.

getEntry
URL getEntry(String path)

Returns null as a framework implementation does not have a proper bundle from which to return an entry.

Specified by:
getEntry in interface Bundle

Parameters:
path - Ignored.

Returns:
null as a framework implementation does not have a proper bundle from which to return an entry.

findEntries
Enumeration<URL> findEntries(String path,
                               String filePattern,
                               boolean recurse)

Returns null as a framework implementation does not have a proper bundle from which to return entries.

Specified by:
findEntries in interface Bundle

Parameters:
path - Ignored.
filePattern - Ignored.
recurse - Ignored.

Returns:
null as a framework implementation does not have a proper bundle from which to return entries.
public interface FrameworkFactory

A factory for creating Framework instances.

A framework implementation jar must contain the following resource:

/META-INF/services/org.osgi.framework.launch.FrameworkFactory

This UTF-8 encoded resource must contain the name of the framework implementation's FrameworkFactory implementation class. Space and tab characters, including blank lines, in the resource must be ignored. The number sign ("#\u0023") and all characters following it on each line are a comment and must be ignored.

Launchers can find the name of the FrameworkFactory implementation class in the resource and then load and construct a FrameworkFactory object for the framework implementation. The FrameworkFactory implementation class must have a public, no-argument constructor.

Java™ SE 6 introduced the ServiceLoader class which can create a FrameworkFactory instance from the resource.

Version: $Revision: 8093 $
ThreadSafe

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework newFramework(Map&lt;String,String&gt; configuration)</td>
<td>161</td>
</tr>
</tbody>
</table>

Create a new Framework instance.

### Method Detail

**newFramework**

Framework newFramework(Map<String,String> configuration)

Create a new Framework instance.

**Parameters:**

configuration - The framework properties to configure the new framework instance. If framework properties are not provided by the configuration argument, the created framework instance must use some reasonable default configuration appropriate for the current VM. For example, the system packages for the current execution environment should be properly exported. The specified configuration argument may be null. The created framework instance must copy any information needed from the specified configuration argument since the configuration argument can be changed after the framework instance has been created.

**Returns:**

A new, configured Framework instance. The framework instance must be in the Bundle.INSTALLED state.

**Throws:**

SecurityException - If the caller does not have AllPermission, and the Java Runtime Environment supports permissions.
Package org.osgi.service.packageadmin

Package Admin Package Version 1.3.

See:  
Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleInfo</td>
<td>163</td>
</tr>
<tr>
<td>BundlePackageAdmin</td>
<td>164</td>
</tr>
<tr>
<td>BundleWiring</td>
<td>165</td>
</tr>
<tr>
<td>ExportedPackage</td>
<td>169</td>
</tr>
<tr>
<td>PackageAdmin</td>
<td>171</td>
</tr>
<tr>
<td>PackageExport</td>
<td>179</td>
</tr>
<tr>
<td>RequiredBundle</td>
<td>181</td>
</tr>
</tbody>
</table>

### Package org.osgi.service.packageadmin Description

Package Admin Package Version 1.3.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. For example:

```
Import-Package: org.osgi.service.packageadmin; version="[1.3,2.0)"
```
Interface BundleInfo

```java
public interface BundleInfo extends BundleReference
```

Bundle Information. Since a bundle update can change some information of a bundle, different bundle wirings for the same bundle can have different bundle information.

Since: 1.3
Version: $Revision: 8569 $
ThreadSafe

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String <code>getSymbolicName()</code></td>
<td>Returns the symbolic name for this bundle info.</td>
<td>163</td>
</tr>
<tr>
<td>Version <code>getVersion()</code></td>
<td>Returns the version for this bundle info.</td>
<td>163</td>
</tr>
</tbody>
</table>

### Method Detail

#### getSymbolicName

- **String** `getSymbolicName()`
  - Returns the symbolic name for this bundle info.

  **Returns:**
  - The symbolic name for this bundle info.

  **See Also:**
  - `Bundle.getSymbolicName()`

#### getVersion

- **Version** `getVersion()`
  - Returns the version for this bundle info.

  **Returns:**
  - The version for this bundle info, or `Version.emptyVersion` if this bundle info has no version information.

  **See Also:**
  - `Bundle.getVersion()`
public interface BundlePackageAdmin
extends BundleReference

The bundle adapter for the Package Admin service. Package Admin bundle adapters for a bundle can be obtained by calling bundle.adapt(BundlePackageAdmin.class).

Objects implementing this interface are created by the framework.

Since: 1.3
Version: $Revision: 8569 $
ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Collection&lt;?BundleWiring&gt; getWirings()</th>
<th>Page 164</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return the in use wirings for the referenced bundle.</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.BundleReference

getBundle

Method Detail

getWirings

Collection<?BundleWiring> getWirings()

Return the in use wirings for the referenced bundle.

If the referenced bundle is a non-fragment bundle, then the result is a List of wirings where the list is ordered in reverse chronological order such that the first wiring is the current bundle wiring and last wiring is the oldest bundle wiring.

If the referenced bundle is a fragment bundle, then the result is a Collection of wirings which include the referenced fragment bundle.

The collection will only contain in use wirings. Generally the collection will have at least one wiring for the bundle; the current wiring. However, for an uninstalled bundle with no in use wirings or a newly installed bundle which has not been resolved, the collection will be empty.

Returns:

A Collection containing a snapshot of the BundleWirings for the referenced bundle.
public interface BundleWiring extends BundleReference

A wiring for a bundle. Each time a bundle is resolved, a new wiring of the bundle exists. It consists of a bundle and
it attached fragments and represents the dependencies with other bundle wirings.

The bundle wiring for a bundle is the current bundle wiring if the bundle is resolved and the bundle wiring is the
most recent bundle wiring. All bundle wirings for a bundle except the current bundle wiring are considered removal
pending. A bundle wiring is in use if it is the current wiring or if some other bundle wiring is dependent upon it. For
example, wired to a package exported by the bundle wiring or requires the bundle wiring. An in use bundle wiring
generally has an associated class loader. Once a bundle wiring is no longer in use, it is considered stale and is
discarded by the framework.

A list of all in use wirings for a bundle can be obtained by calling bundle.adapt(BundlePackageAdmin.class).
getWirings(). For non-fragment bundles, the first item in the returned list is the current wiring.

The current wiring for a non-fragment bundle can be obtained by calling bundle.adapt(BundleWiring.class). A
fragment bundle does not itself have BundleWiring. So calling bundle.adapt(BundleWiring.class) on a fragment
must return null.

Objects implementing this interface are created by the framework.

Since: 1.3
Version: $Revision: 8666 $
ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleInfo getBundleInfo()</td>
<td>Returns the bundle information for the bundle in this bundle wiring.</td>
</tr>
<tr>
<td>Collection&lt;PackageExport&gt; getExported()</td>
<td>Returns the package exports for packages exported by this bundle wiring.</td>
</tr>
<tr>
<td>List&lt;BundleInfo&gt; getFragmentInfos()</td>
<td>Returns the bundle information for all attached fragments of this bundle wiring.</td>
</tr>
<tr>
<td>Collection&lt;PackageExport&gt; getImported()</td>
<td>Returns the package exports for packages imported by this bundle wiring.</td>
</tr>
<tr>
<td>List&lt;BundleWiring&gt; getRequired()</td>
<td>Returns the bundle wirings that are required by this bundle wiring.</td>
</tr>
<tr>
<td>Collection&lt;BundleWiring&gt; getRequiring()</td>
<td>Returns the bundle wirings that require this bundle wiring.</td>
</tr>
<tr>
<td>boolean isCurrent()</td>
<td>Returns true if this bundle wiring is the current bundle wiring.</td>
</tr>
<tr>
<td>boolean isInUse()</td>
<td>Returns true if this bundle wiring is in use.</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.BundleReference

getBundle
Method Detail

getExported

Collection<PackageExport> getExported()

Returns the package exports for packages exported by this bundle wiring.

If this bundle wiring is for the system bundle (that is, the bundle with id zero), this method returns all the packages known to be exported by the system bundle. This will include the package specified by the Constants.FRAMEWORK_SYSTEMPACKAGES system property as well as any other package exported by the framework implementation.

Returns: A Collection containing the PackageExports, or an empty collection if this bundle wiring exports no packages. If this bundle wiring is not in use, null will be returned.

getImported

Collection<PackageExport> getImported()

Returns the package exports for packages imported by this bundle wiring.

If this bundle wiring requires other bundle wirings then the package exports from the required bundle wirings are included in the result. (See getRequired()).

Since packages can be dynamically imported, the result of this method can change if this bundle wiring dynamically imports another package.

Returns: A Collection containing a snapshot of the PackageExports, or an empty collection if the bundle wiring imports no packages. If this bundle wiring is not in use, null will be returned.

getRequiring

Collection<BundleWiring> getRequiring()

Returns the bundle wirings that require this bundle wiring.

If this bundle wiring is required and then re-exported by another bundle wiring, then the result will include the requiring bundle wirings of the re-exporting bundle wiring.

The result of this method can change if additional bundle wirings require this bundle wiring.

Returns: A Collection containing a snapshot of the BundleWiringS requiring this bundle wiring, or an empty collection if no bundle wirings require this bundle wiring. If this bundle wiring is not in use, null will be returned.

getRequired

List<BundleWiring> getRequired()

Returns the bundle wirings that are required by this bundle wiring.

If a required bundle wiring requires and then re-exports another bundle wiring, then the result will include the re-exported bundle wiring.
The bundle wirings in the list are ordered as specified in the Require-Bundle header.

**Returns:**

A list containing the BundleWiring's required by this bundle wiring, or an empty list if this bundle wiring does not require any bundle. If this bundle wiring is not in use, null will be returned.

---

### isCurrent

**boolean isCurrent()**

Returns **true** if this bundle wiring is the current bundle wiring. The bundle wiring for a bundle is the current bundle wiring if the bundle is resolved and the bundle wiring is the most recent bundle wiring. All bundle wirings for a bundle except the current bundle wiring are considered removal pending.

**Returns:**

**true** if this bundle wiring is the current bundle wiring; **false** otherwise.

---

### isInUse

**boolean isInUse()**

Returns **true** if this bundle wiring is in use. A bundle wiring is in use if it is the current wiring or if some other bundle wiring is dependent upon it. Once a bundle wiring is no longer in use, it is considered stale and is discarded by the framework.

**Returns:**

**true** if this bundle wiring is in use; **false** otherwise.

---

### getBundleInfo

**BundleInfo getBundleInfo()**

Returns the bundle information for the bundle in this bundle wiring. Since a bundle update can change some information of a bundle, different bundle wirings for the same bundle can have different bundle information.

The bundle object referenced by the returned BundleInfo may return different information than the returned BundleInfo since the returned BundleInfo may refer to an older revision of the bundle.

**Returns:**

The bundle information for this bundle wiring.

---

### getFragmentInfos

**List<BundleInfo> getFragmentInfos()**

Returns the bundle information for all attached fragments of this bundle wiring. Since a bundle update can change some information of a bundle, different bundle wirings for the same bundle can have different bundle information.

The bundle infos in the list are ordered in fragment attachment order such that the first info in the list is the first attached fragment and the last info in the list is the last attached fragment.
Returns:
A list containing the BundleInfo objects for all attached fragments attached of this bundle wiring, or an empty list if this bundle wiring does not have any attached fragments. If this bundle wiring is not in use, null will be returned.
public interface ExportedPackage

Deprecated.

An exported package. Objects implementing this interface are created by the Package Admin service.

The term exported package refers to a package that has been exported from a resolved bundle. This package may or may not be currently wired to other bundles.

The information about an exported package provided by this object may change. An ExportedPackage object becomes stale if the package it references has been updated or removed as a result of calling PackageAdmin.refreshPackages(). If this object becomes stale, its getName() and getVersion() methods continue to return their original values, isRemovalPending() returns true, and getExportingBundle() and getImportingBundles() return null.

Version:
$Revision: 8569 $
ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bundle</strong> getExportingBundle()</td>
<td>169</td>
</tr>
<tr>
<td>Returns the bundle exporting the package associated with this exported package.</td>
<td></td>
</tr>
<tr>
<td><strong>Bundle[]</strong> getImportingBundles()</td>
<td>170</td>
</tr>
<tr>
<td>Returns the resolved bundles that are currently wired to this exported package.</td>
<td></td>
</tr>
<tr>
<td><strong>String</strong> getName()</td>
<td>169</td>
</tr>
<tr>
<td>Returns the name of the package associated with this exported package.</td>
<td></td>
</tr>
<tr>
<td><strong>String</strong> getSpecificationVersion()</td>
<td>170</td>
</tr>
<tr>
<td>Deprecated. As of 1.2, replaced by getVersion().</td>
<td></td>
</tr>
<tr>
<td><strong>Version</strong> getVersion()</td>
<td>170</td>
</tr>
<tr>
<td>Returns the version of this exported package.</td>
<td></td>
</tr>
<tr>
<td><strong>boolean</strong> isRemovalPending()</td>
<td>170</td>
</tr>
<tr>
<td>Returns true if the package associated with this ExportedPackage object has been exported by a bundle that has been updated or uninstalled.</td>
<td></td>
</tr>
</tbody>
</table>

**Method Detail**

**getName**

String getName()

Returns the name of the package associated with this exported package.

**Returns:**

The name of this exported package.

**getExportingBundle**

Bundle getExportingBundle()

Returns the bundle exporting the package associated with this exported package.
Interface ExportedPackage

Returns:
The exporting bundle, or null if this ExportedPackage object has become stale.

getImportingBundles

Bundle[] getImportingBundles()

Returns the resolved bundles that are currently wired to this exported package.

Bundles which require the exporting bundle associated with this exported package are considered to be wired to this exported package are included in the returned array. See RequiredBundle.getRequiringBundles().

Returns:
The array of resolved bundles currently wired to this exported package, or null if this ExportedPackage object has become stale. The array will be empty if no bundles are wired to this exported package.

getSpecificationVersion

String getSpecificationVersion()

Deprecated. As of 1.2, replaced by getVersion().

Returns the version of this exported package.

Returns:
The version of this exported package, or null if no version information is available.

getVersion

Version getVersion()

Returns the version of this exported package.

Returns:
The version of this exported package, or Version.emptyVersion if no version information is available.

Since: 1.2

isRemovalPending

boolean isRemovalPending()

Returns true if the package associated with this ExportedPackage object has been exported by a bundle that has been updated or uninstalled.

Returns:
true if the associated package is being exported by a bundle that has been updated or uninstalled, or if this ExportedPackage object has become stale; false otherwise.
public interface PackageAdmin

Framework service which allows bundle programmers to inspect the package wiring state of bundles in the Framework as well as other functions related to the class loader network among bundles.

If present, there will only be a single instance of this service registered with the Framework.

Version:

See Also:

BundlePackageAdmin

ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>BUNDLE_TYPE_FRAGMENT</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.2.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bundle</td>
<td>getBundle(Class clazz)</td>
</tr>
<tr>
<td></td>
<td>Returns the bundle from which the specified class is loaded.</td>
</tr>
<tr>
<td>Bundle[]</td>
<td>getBundles(String symbolicName, String versionRange)</td>
</tr>
<tr>
<td></td>
<td>Returns the bundles with the specified symbolic name whose bundle version is within the specified version range.</td>
</tr>
<tr>
<td>int</td>
<td>getBundleType(Bundle bundle)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>Collection &lt;Bundle&gt;</td>
<td>getDependencyClosure(Collection&lt;Bundle&gt; bundles)</td>
</tr>
<tr>
<td></td>
<td>Return the dependency closure for the specified bundles.</td>
</tr>
<tr>
<td>ExportedPackage</td>
<td>getExportedPackage(String name)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>ExportedPackage[]</td>
<td>getExportedPackages(String name)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>ExportedPackage[]</td>
<td>getExportedPackages(Bundle bundle)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>Bundle[]</td>
<td>getFragments(Bundle bundle)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>Collection &lt;Bundle&gt;</td>
<td>getHosts(Bundle bundle)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>List&lt;PackageExport&gt;</td>
<td>getPackageExports(String name)</td>
</tr>
<tr>
<td></td>
<td>Returns a list of package exports for exported packages with the specified package name.</td>
</tr>
<tr>
<td>Collection &lt;Bundle&gt;</td>
<td>getRemovalPendingBundles()</td>
</tr>
<tr>
<td></td>
<td>Return the bundles that have is non-current, in use BundleWiring S.</td>
</tr>
<tr>
<td>RequiredBundle[]</td>
<td>getRequiredBundles(String symbolicName)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>void</td>
<td>refreshBundles(Collection&lt;Bundle&gt; bundles, FrameworkListener... listeners)</td>
</tr>
<tr>
<td></td>
<td>Refresh the specified bundles.</td>
</tr>
<tr>
<td>void</td>
<td>refreshPackages(Bundle[] bundles)</td>
</tr>
<tr>
<td></td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>boolean</td>
<td>resolveBundles(Collection&lt;Bundle&gt; bundles)</td>
</tr>
<tr>
<td></td>
<td>Resolve the specified bundles.</td>
</tr>
</tbody>
</table>
Field Detail

BUNDLE_TYPE_FRAGMENT

public static final int BUNDLE_TYPE_FRAGMENT = 1

Deprecated.

Bundle type indicating the bundle is a fragment bundle.

Since:
1.2

Method Detail

getPackageExports

List<PackageExport> getPackageExports(String name)

Returns a list of package exports for exported packages with the specified package name.

The package exports in the list are ordered in descending version such that the first package export has the highest version and last package export has the lowest version. The list will only contain package exports of current bundle wirings.

Parameters:
name - The package name of the exported packages to be returned or null to return all exported packages.

Returns:
A List containing a snapshot of PackageExports, or an empty list if no exported packages with the specified name exist.

Since:
1.3

refreshBundles

void refreshBundles(Collection<Bundle> bundles,
FrameworkListener... listeners)

Refresh the specified bundles. This forces the update (replacement) or removal of packages exported by the specified bundles.

The technique by which this is accomplished may vary among different Framework implementations. One permissible implementation is to stop and restart the Framework.

This method returns to the caller immediately and then performs the following steps on a separate thread:

1. Compute the dependency closure of the specified bundles. If no bundles are specified, compute the dependency closure of the bundles returned by getRemovalPendingBundles().
2. Each bundle in the dependency closure that is in the ACTIVE state will be stopped as described in the Bundle.stop method.
3. Each bundle in the dependency closure that is in the RESOLVED state is unresolved and thus moved to the INSTALLED state. The effect of this step is that bundles in the dependency closure are no longer RESOLVED.
4. Each bundle in the dependency closure that is in the UNINSTALLED state is removed from the dependency closure and is now completely removed from the Framework.

5. Each bundle in the dependency closure that was in the ACTIVE state prior to Step 2 is started as described in the Bundle.start method, causing all bundles required for the restart to be resolved. It is possible that, as a result of the previous steps, packages that were previously exported no longer are. Therefore, some bundles may be unresolvable until another bundle satisfying the dependency has been installed in the Framework.

6. A framework event of type FrameworkEvent.PACKAGES_REFRESHED is fired.

For any exceptions that are thrown during any of these steps, a framework event of type FrameworkEvent.ERROR is fired containing the exception. The source bundle for these events should be the specific bundle to which the exception is related. If no specific bundle can be associated with the exception then the System Bundle must be used as the source bundle for the event. All framework events fired by this method are also delivered to the specified FrameworkListeners in the order they are specified.

Parameters:
- bundles: The bundles to be refreshed, or null to refresh the bundles returned by getRemovalPendingBundles().
- listeners: Zero or more listeners to be notified when the refresh bundles has been completed. The specified listeners do not need to be otherwise registered with the framework. If a specified listener is already registered with the framework, it will be notified twice.

Throws:
- SecurityException: If the caller does not have AdminPermission[System Bundle,RESOLVE] and the Java runtime environment supports permissions.
- IllegalArgumentException: If the specified bundles were not created by the same framework instance that registered this PackageAdmin service.

Since: 1.3

resolveBundles

boolean resolveBundles(Collection<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.

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.

Throws:
- SecurityException: If the caller does not have AdminPermission[System Bundle,RESOLVE] and the Java runtime environment supports permissions.
- IllegalArgumentException: If the specified bundles were not created by the same framework instance that registered this PackageAdmin service.

Since: 1.3

gerBundles

Bundle[] getBundles(String symbolicName, String versionRange)
Interface PackageAdmin

Returns the bundles with the specified symbolic name whose bundle version is 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 whose bundle versions 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

getBundle

Bundle getBundle(Class clazz)

Returns the bundle from which the specified class is loaded. The class loader of the returned bundle must have been used to load the specified class. If the class was not loaded by a bundle class loader then null is returned.

Parameters:
- clazz - The class object from which to locate the bundle.

Returns:
The bundle from which the specified class is loaded or null if the class was not loaded by a bundle class loader created by the same framework instance that registered this PackageAdmin service.

Since: 1.2

getRemovalPendingBundles

Collection<Bundle> getRemovalPendingBundles()

Return the bundles that have is non-current, in use BundleWiring s. This is typically the bundles which have been updated or uninstalled since the last call to refreshBundles(Collection, FrameworkListener...).

Returns:
A Collection containing a snapshot of the Bundle s which have non-current, in use BundleWiring s, or an empty collection if there are no such bundles.

Since: 1.3

getDependencyClosure

Collection<Bundle> getDependencyClosure(Collection<Bundle> bundles)

Return the dependency closure for the specified bundles.

A graph of bundles is computed starting with the specified bundles. The graph is expanded by adding any bundle that is either wired to a package that is currently exported by a bundle in the graph or requires a
bundle in the graph. The graph is fully constructed when there is no bundle outside the graph that is wired to a bundle in the graph. The graph may contain \texttt{UNINSTALLED} bundles that are removal pending.

\textbf{Parameters:}
- \texttt{bundles} - The initial bundles for which to generate the dependency closure.

\textbf{Returns:}
- A \texttt{Collection} containing a snapshot of the dependency closure of the specified bundles, or an empty collection if there were no specified bundles.

\textbf{Throws:}
- \texttt{IllegalArgumentException} - If the specified \texttt{Bundle}s were not created by the same framework instance that registered this \texttt{PackageAdmin} service.

\textbf{Since:} 1.3

\textbf{getExportedPackages}

\texttt{ExportedPackage[]} \texttt{getExportedPackages(Bundle} \texttt{bundle)}

\textbf{Deprecated. As of 1.3. Replaced by bundle.adapt(BundleWiring.class). getExported()}

Gets the exported packages for the specified bundle.

\textbf{Parameters:}
- \texttt{bundle} - The bundle whose exported packages are to be returned, or \texttt{null} if all exported packages are to be returned. If the specified bundle is the system bundle (that is, the bundle with id zero), this method returns all the packages known to be exported by the system bundle. This will include the package specified by the \texttt{Constants.FRAMEWORK_SYSTEMPACKAGES} system property as well as any other package exported by the framework implementation.

\textbf{Returns:}
- An array of exported packages, or \texttt{null} if the specified bundle has no exported packages.

\textbf{Throws:}
- \texttt{IllegalArgumentException} - If the specified \texttt{Bundle} was not created by the same framework instance that registered this \texttt{PackageAdmin} service.

\textbf{getExportedPackages}

\texttt{ExportedPackage[]} \texttt{getExportedPackages(String} \texttt{name)}

\textbf{Deprecated. As of 1.3. Replaced by getPackageExports(String).}

Gets the exported packages for the specified package name.

\textbf{Parameters:}
- \texttt{name} - The name of the exported packages to be returned, or \texttt{null} if all exported packages are to be returned.

\textbf{Returns:}
- An array of the exported packages, or \texttt{null} if no exported packages with the specified name exist.

\textbf{Since:} 1.2

\textbf{getExportedPackage}

\texttt{ExportedPackage} \texttt{getExportedPackage(String} \texttt{name)}

\textbf{Deprecated. As of 1.3. Replaced by getPackageExports(String).\texttt{get(0)}.}

Gets the exported package for the specified package name.
If there are multiple exported packages with specified name, the exported package with the highest version will be returned.

Parameters:
- name - The name of the exported package to be returned.

Returns:
The exported package, or null if no exported package with the specified name exists.

---

**refreshPackages**

```java
void refreshPackages(Bundle[] bundles)
```

*Deprecated.* As of 1.3. Replaced by `refreshBundles(Collection, FrameworkListener...)`

Forces the update (replacement) or removal of packages exported by the specified bundles.

Parameters:
- bundles - The bundles whose exported packages are to be updated or removed, or null for all bundles updated or uninstalled since the last call to this method.

Throws:
- SecurityException - If the caller does not have AdminPermission[System Bundle,RESOLVE] and the Java runtime environment supports permissions.
- IllegalArgumentException - If the specified Bundle(s) were not created by the same framework instance that registered this PackageAdmin service.

---

**resolveBundles**

```java
boolean resolveBundles(Bundle[] bundles)
```

*Deprecated.* As of 1.3. Replaced by `resolveBundles(Collection)`

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.

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.

Throws:
- SecurityException - If the caller does not have AdminPermission[System Bundle,RESOLVE] and the Java runtime environment supports permissions.
- IllegalArgumentException - If the specified Bundle(s) were not created by the same framework instance that registered this PackageAdmin service.

*Since:* 1.2

---

**getRequiredBundles**

```java
RequiredBundle[] getRequiredBundles(String symbolicName)
```
**Interface PackageAdmin**

**getRequired**

```java
Bundle[] getRequired(Bundle bundle)
```

*Deprecated. As of 1.3. Use `getBundles(String, String)` to get Bundle objects for the symbolic name and then call `bundle.adapt(BundleWiring.class).getRequired()`.*

Returns an array of required bundles having the specified symbolic name.

If `null` is specified, then all required bundles will be returned.

**Parameters:**

- `symbolicName` - The bundle symbolic name or `null` for all required bundles.

**Returns:**

- An array of required bundles or `null` if no required bundles exist for the specified symbolic name.

**Since:**

1.2

---

**getFragments**

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

*Deprecated. As of 1.3. Replaced by `bundle.adapt(BundleWiring.class).getFragmentInfos()`.*

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.

This method does not attempt to resolve the specified bundle. If the specified bundle is not resolved then `null` is returned.

**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 or the bundle is not resolved.

**Throws:**

- `IllegalArgumentException` - If the specified `Bundle` was not created by the same framework instance that registered this `PackageAdmin` service.

**Since:**

1.2

---

**getHosts**

```java
Bundle[] getHosts(Bundle bundle)
```

*Deprecated. As of 1.3. Use `bundle.adapt(BundlePackageAdmin.class).getWirings()` to get the bundle wirings in which the fragment participates.*

Returns the host bundles to which the specified fragment bundle is attached.

**Parameters:**

- `bundle` - The fragment bundle whose host bundles are to be returned.

**Returns:**

- An array containing the host bundles to which the specified fragment is attached or `null` if the specified bundle is not a fragment or is not attached to any host bundles.

**Throws:**

- `IllegalArgumentException` - If the specified `Bundle` was not created by the same framework instance that registered this `PackageAdmin` service.

**Since:**

1.2
getBundleType

int getBundleType(Bundle bundle)

Deprecated. As of 1.3. Replaced by Bundle.getTypes().

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

- **BUNDLE_TYPE_FRAGMENT**

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.

**Parameters:**
- bundle - The bundle for which to return the special type.

**Returns:**
- The special type of the bundle.

**Throws:**
- IllegalArgumentException - If the specified Bundle was not created by the same framework instance that registered this PackageAdmin service.

**Since:**
- 1.2
public interface PackageExport

A package that has been exported from a bundle wiring. This package may or may not be imported by any bundle wiring.

Objects implementing this interface are created by the framework.

Since: 1.6
Version: $Revision$
ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleWiring getExporter()</td>
<td>179</td>
</tr>
<tr>
<td>Returns the bundle wiring exporting the package.</td>
<td></td>
</tr>
<tr>
<td>Collection&lt;?BundleWiring&gt; getImporters()</td>
<td>180</td>
</tr>
<tr>
<td>Returns the bundle wirings that are importing the package.</td>
<td></td>
</tr>
<tr>
<td>String getName()</td>
<td>179</td>
</tr>
<tr>
<td>Returns the name of the package.</td>
<td></td>
</tr>
<tr>
<td>Version getVersion()</td>
<td>179</td>
</tr>
<tr>
<td>Returns the version of the package.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Detail

**getName**

String getName()

Returns the name of the package.

**getVersion**

Version getVersion()

Returns the version of the package.

**getExporter**

BundleWiring getExporter()

Returns the bundle wiring exporting the package.
Returns:
The bundle wiring exporting the package. If the bundle wiring is not in use, null will be returned.

getImporters

Collection<BundleWiring> getImporters()

Returns the bundle wirings that are importing the package.

Bundle wirings which require the bundle wiring exporting the package are considered to be importing the package and are included in the result.

The result of this method can change if additional bundle wirings import the package.

Returns:
A Collection containing a snapshot of bundle wirings currently wired to this package, or an empty collection if no bundles are wired to this package. If the exporter's bundle wiring is not in use, null will be returned.
public interface RequiredBundle

Deprecated.

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

The term \textit{required bundle} refers to a resolved bundle that has a bundle symbolic name and is not a fragment. That is, a bundle that may be required by other bundles. This bundle may or may not be currently required by other bundles.

The information about a required bundle provided by this object may change. A \texttt{RequiredBundle} object becomes stale if an exported package of the bundle it references has been updated or removed as a result of calling \texttt{PackageAdmin.refreshPackages()}. If this object becomes stale, its \texttt{getSymbolicName()} and \texttt{getVersion()} methods continue to return their original values, \texttt{isRemovalPending()} returns true, and \texttt{getBundle()} and \texttt{getRequiringBundles()} return null.

\textbf{Since:} 1.2

\textbf{Version:} $Revision: 8569$

\textbf{ThreadSafe}

\begin{tabular}{|l|l|}
\hline
\textbf{Method Summary} & \textbf{Page} \\
\hline
\texttt{Bundle} & \texttt{getBundle()} & 181 \\
 & \hspace{1cm} Returns the bundle associated with this required bundle. & \\
\hline
\texttt{Bundle[]} & \texttt{getRequiringBundles()} & 182 \\
 & \hspace{1cm} Returns the bundles that currently require this required bundle. & \\
\hline
\texttt{String} & \texttt{getSymbolicName()} & 181 \\
 & \hspace{1cm} Returns the symbolic name of this required bundle. & \\
\hline
\texttt{Version} & \texttt{getVersion()} & 182 \\
 & \hspace{1cm} Returns the version of this required bundle. & \\
\hline
\texttt{boolean} & \texttt{isRemovalPending()} & 182 \\
 & \hspace{1cm} Returns true if the bundle associated with this \texttt{RequiredBundle} object has been updated or uninstalled. & \\
\hline
\end{tabular}

\section*{Method Detail}

\subsection*{getSymbolicName}

\texttt{String \texttt{getSymbolicName}()}

\texttt{Returns the symbolic name of this required bundle.}

\textbf{Returns:}

\texttt{The symbolic name of this required bundle.}

\subsection*{getBundle}

\texttt{Bundle \texttt{getBundle}()}

\texttt{Returns the bundle associated with this required bundle.}
Returns:
The bundle, or null if this RequiredBundle object has become stale.

getRequiringBundles

Bundle[] getRequiringBundles()

Returns the bundles that currently require this required bundle.

If this required bundle is required and then 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 bundles currently requiring this required bundle, or null if this RequiredBundle object has become stale. The array will be empty if no bundles require this required package.

getVersion

Version getVersion()

Returns the version of this required bundle.

Returns:
The version of this required bundle, or Version.emptyVersion if no version information is available.

isRemovalPending

boolean isRemovalPending()

Returns true if the bundle associated with this RequiredBundle object has been updated or uninstalled.

Returns:
true if the required bundle has been updated or uninstalled, or if the RequiredBundle object has become stale; false otherwise.
Package org.osgi.service.startlevel

Start Level Package Version 1.2.

See: Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleStartLevel</td>
<td>184</td>
</tr>
<tr>
<td>The bundle adapter for the Start Level service.</td>
<td></td>
</tr>
<tr>
<td>StartLevel</td>
<td>186</td>
</tr>
<tr>
<td>The StartLevel service allows management agents to</td>
<td></td>
</tr>
<tr>
<td>manage a start level assigned to each bundle and the</td>
<td></td>
</tr>
<tr>
<td>active start level of the Framework.</td>
<td></td>
</tr>
</tbody>
</table>

Package org.osgi.service.startlevel Description

Start Level Package Version 1.2.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. For example:

Import-Package: org.osgi.service.startlevel; version="[1.2,2.0)"
Interface BundleStartLevel

org.osgi.service.startlevel

All Superinterfaces:
   BundleReference

public interface BundleStartLevel
extends BundleReference

The bundle adapter for the Start Level service. Start level bundle adapters for a bundle can be obtained by calling
bundle.adapt(BundleStartLevel.class).

Since: 1.2
Version: $Revision: 8569 $
ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getStartLevel()</td>
<td>184</td>
</tr>
<tr>
<td>boolean isActivationPolicyUsed()</td>
<td>185</td>
</tr>
<tr>
<td>boolean isPersistentlyStarted()</td>
<td>185</td>
</tr>
<tr>
<td>void setStartLevel(int startlevel)</td>
<td>184</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.BundleReference
getBundle

Method Detail

getStartLevel

int getStartLevel()

Return the assigned start level value for the bundle.

Returns: The start level value of the bundle.

Throws: IllegalStateException - If the bundle has been uninstalled.

See Also:
   setStartLevel(int)

setStartLevel

void setStartLevel(int startlevel)

Assign a start level value to the bundle.
The bundle will be assigned the specified start level. The start level value assigned to the bundle will be persistently recorded by the Framework.

If the new start level for the bundle is lower than or equal to the active start level of the Framework and the bundle's autostart setting indicates this bundle must be started, the Framework will start the bundle as described in the `Bundle.start(int)` method using the `Bundle.START_TRANSIENT` option. The `Bundle.START_ACTIVATION_POLICY` option must also be used if `isActivationPolicyUsed()` returns true. The actual starting of the bundle must occur asynchronously.

If the new start level for the bundle is higher than the active start level of the Framework, the Framework will stop the bundle as described in the `Bundle.stop(int)` method using the `Bundle.STOP_TRANSIENT` option. The actual stopping of the bundle must occur asynchronously.

**Parameters:**
- `startlevel` - The new start level for the bundle.

**Throws:**
- `IllegalArgumentException` - If the specified start level is less than or equal to zero, or if the bundle is the system bundle.
- `IllegalStateException` - If the bundle has been uninstalled.
- `SecurityException` - If the caller does not have `AdminPermission[bundle,EXECUTE]` and the Java runtime environment supports permissions.

### isPersistentlyStarted

**boolean isPersistentlyStarted()**

Returns whether the bundle's autostart setting indicates it must be started.

The autostart setting of a bundle indicates whether the bundle is to be started when its start level is reached.

**Returns:**
- `true` if the autostart setting of the bundle indicates it is to be started. `false` otherwise.

**Throws:**
- `IllegalStateException` - If this bundle has been uninstalled.

**See Also:**
- `Bundle.START_TRANSIENT`

### isActivationPolicyUsed

**boolean isActivationPolicyUsed()**

Returns whether the bundle's autostart setting indicates that the activation policy declared in the bundle manifest must be used.

The autostart setting of a bundle indicates whether the bundle's declared activation policy is to be used when the bundle is started.

**Returns:**
- `true` if the bundle's autostart setting indicates the activation policy declared in the manifest must be used. `false` if the bundle must be eagerly activated.

**Throws:**
- `IllegalStateException` - If the bundle has been uninstalled.

**See Also:**
- `Bundle.START_ACTIVATION_POLICY`
public interface StartLevel

The StartLevel service allows management agents to manage a start level assigned to each bundle and the active start level of the Framework. There is at most one StartLevel service present in the OSGi environment.

A start level is defined to be a state of execution in which the Framework exists. StartLevel values are defined as unsigned integers with 0 (zero) being the state where the Framework is not launched. Progressively higher integral values represent progressively higher start levels. e.g. 2 is a higher start level than 1.

Access to the StartLevel service is protected by corresponding ServicePermission. In addition AdminPermission is required to actually modify start level information.

Start Level support in the Framework includes the ability to control the beginning start level of the Framework, to modify the active start level of the Framework and to assign a specific start level to a bundle. How the beginning start level of a Framework is specified is implementation dependent. It may be a command line argument when invoking the Framework implementation.

When the Framework is first started it must be at start level zero. In this state, no bundles are running. This is the initial state of the Framework before it is launched. When the Framework is launched, the Framework will enter start level one and all bundles which are assigned to start level one and whose autostart setting indicates the bundle should be started are started as described in the Bundle.start method. The Framework will continue to increase the start level, starting bundles at each start level, until the Framework has reached a beginning start level. At this point the Framework has completed starting bundles and will then fire a Framework event of type FrameworkEvent.STARTED to announce it has completed its launch.

Within a start level, bundles may be started in an order defined by the Framework implementation. This may be something like ascending Bundle.getBundleId order or an order based upon dependencies between bundles. A similar but reversed order may be used when stopping bundles within a start level.

The StartLevel service can be used by management bundles to alter the active start level of the framework.

Version:
$Revision: 8569$
ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getBundleStartLevel(Bundle bundle)</td>
<td>188</td>
</tr>
<tr>
<td>Deprecated. As of 1.2.</td>
<td></td>
</tr>
<tr>
<td>int getInitialBundleStartLevel()</td>
<td>189</td>
</tr>
<tr>
<td>Return the initial start level value that is assigned to a Bundle when it is first installed.</td>
<td></td>
</tr>
<tr>
<td>int getStartLevel()</td>
<td>187</td>
</tr>
<tr>
<td>Return the active start level value of the Framework.</td>
<td></td>
</tr>
<tr>
<td>boolean isBundleActivationPolicyUsed(Bundle bundle)</td>
<td>190</td>
</tr>
<tr>
<td>Deprecated. As of 1.2.</td>
<td></td>
</tr>
<tr>
<td>boolean isBundlePersistentlyStarted(Bundle bundle)</td>
<td>189</td>
</tr>
<tr>
<td>Deprecated. As of 1.2.</td>
<td></td>
</tr>
<tr>
<td>void setBundleStartLevel(Bundle bundle, int startlevel)</td>
<td>188</td>
</tr>
<tr>
<td>Deprecated. As of 1.2.</td>
<td></td>
</tr>
<tr>
<td>void setInitialBundleStartLevel(int startlevel)</td>
<td>189</td>
</tr>
<tr>
<td>Set the initial start level value that is assigned to a Bundle when it is first installed.</td>
<td></td>
</tr>
<tr>
<td>void setStartLevel(int startlevel)</td>
<td>187</td>
</tr>
<tr>
<td>Modify the active start level of the Framework.</td>
<td></td>
</tr>
<tr>
<td>void setStartLevel(int startlevel, FrameworkListener... listeners)</td>
<td>187</td>
</tr>
<tr>
<td>Modify the active start level of the Framework and notify when complete.</td>
<td></td>
</tr>
</tbody>
</table>
### Method Detail

#### getStartLevel

```java
int getStartLevel()
```

Return the active start level value of the Framework. If the Framework is in the process of changing the start level this method must return the active start level if this differs from the requested start level.

**Returns:**

The active start level value of the Framework.

#### setStartLevel

```java
void setStartLevel(int startlevel)
```

Modify the active start level of the Framework.

The Framework will move to the requested start level. This method will return immediately to the caller and the start level change will occur asynchronously on another thread.

If the specified start level is higher than the active start level, the Framework will continue to increase the start level until the Framework has reached the specified start level. At each intermediate start level value on the way to and including the target start level, the Framework must:

1. Change the active start level to the intermediate start level value.
2. Start bundles at the intermediate start level whose autostart setting indicate they must be started. They are started as described in the `Bundle.start(int)` method using the `Bundle.START_TRANSIENT` option. The `Bundle.START_ACTIVATION_POLICY` option must also be used if `isBundleActivationPolicyUsed(Bundle)` returns `true` for the bundle.

When this process completes after the specified start level is reached, the Framework will fire a Framework event of type `FrameworkEvent.STARTLEVEL_CHANGED` to announce it has moved to the specified start level.

If the specified start level is lower than the active start level, the Framework will continue to decrease the start level until the Framework has reached the specified start level. At each intermediate start level value on the way to and including the specified start level, the framework must:

1. Stop bundles at the intermediate start level as described in the `Bundle.stop(int)` method using the `Bundle.STOP_TRANSIENT` option.
2. Change the active start level to the intermediate start level value.

When this process completes after the specified start level is reached, the Framework will fire a Framework event of type `FrameworkEvent.STARTLEVEL_CHANGED` to announce it has moved to the specified start level.

If the specified start level is equal to the active start level, then no bundles are started or stopped, however, the Framework must fire a Framework event of type `FrameworkEvent.STARTLEVEL_CHANGED` to announce it has finished moving to the specified start level. This event may arrive before this method return.

**Parameters:**

- `startlevel` - The requested start level for the Framework.

**Throws:**

- `IllegalArgumentException` - If the specified start level is less than or equal to zero.
- `SecurityException` - If the caller does not have `AdminPermission[System Bundle,STARTLEVEL]` and the Java runtime environment supports permissions.

#### setStartLevel

```java
void setStartLevel(int startlevel,
                   FrameworkListener... listeners)
```

Modify the active start level of the Framework.
Interface StartLevel

Modify the active start level of the Framework and notify when complete.

This method performs the same actions as `setStartLevel(int)` and notifies the specified `FrameworkListener`s, in the order specified, when the start level change is complete. When the start level change completes normally, each specified `FrameworkListener` will be called with a Framework event of type `FrameworkEvent.STARTLEVEL_CHANGED`. If the start level change does not complete normally, each specified `FrameworkListener` will be called with a Framework event of type `FrameworkEvent.ERROR`.

Parameters:
- `startlevel` - The requested start level for the Framework.
- `listeners` - Zero or more listeners to be notified when the start level change has been completed. The specified listeners do not need to be otherwise registered with the framework. If a specified listener is already registered with the framework, it will be notified twice.

Throws:
- `IllegalArgumentException` - If the specified start level is less than or equal to zero.
- `SecurityException` - If the caller does not have `AdminPermission[System Bundle,STARTLEVEL]` and the Java runtime environment supports permissions.

Since: 1.2

See Also: `setStartLevel(int)`

getBundleStartLevel

int `getBundleStartLevel(Bundle bundle)`

**Deprecated. As of 1.2. Replaced by `bundle.adapt(BundleStartLevel.class).getStartLevel()`**

Return the assigned start level value for the specified Bundle.

Parameters:
- `bundle` - The target bundle.

Returns: The start level value of the specified Bundle.

Throws:
- `IllegalArgumentException` - If the specified bundle has been uninstalled or if the specified bundle was not created by the same framework instance that registered this `StartLevel` service.

setBundleStartLevel

void `setBundleStartLevel(Bundle bundle, int startlevel)`

**Deprecated. As of 1.2. Replaced by `bundle.adapt(BundleStartLevel.class).setStartLevel(int)`**

Assign a start level value to the specified Bundle.

The specified bundle will be assigned the specified start level. The start level value assigned to the bundle will be persistently recorded by the Framework.

If the new start level for the bundle is lower than or equal to the active start level of the Framework and the bundle's autostart setting indicates the bundle must be started, the Framework will start the specified bundle as described in the `Bundle.start(int)` method using the `Bundle.START_TRANSIENT` option. The `Bundle.START_ACTIVATION_POLICY` option must also be used if `isBundleActivationPolicyUsed(Bundle)` returns `true` for the bundle. The actual starting of this bundle must occur asynchronously.

If the new start level for the bundle is higher than the active start level of the Framework, the Framework will stop the specified bundle as described in the `Bundle.stop(int)` method using the `Bundle.STOP_TRANSIENT` option. The actual stopping of this bundle must occur asynchronously.
Interface StartLevel

**Parameters:**
- bundle - The target bundle.
- startlevel - The new start level for the specified Bundle.

**Throws:**
- IllegalArgumentException - If the specified bundle has been uninstalled, or if the specified start level is less than or equal to zero, or if the specified bundle is the system bundle, or if the specified bundle was not created by the same framework instance that registered this StartLevel service.
- SecurityException - If the caller does not have AdminPermission[bundle,EXECUTE] and the Java runtime environment supports permissions.

### getInitialBundleStartLevel

```java
int getInitialBundleStartLevel()
```

Return the initial start level value that is assigned to a Bundle when it is first installed.

**Returns:** The initial start level value for Bundles.

**See Also:** `setInitialBundleStartLevel()`

### setInitialBundleStartLevel

```java
void setInitialBundleStartLevel(int startlevel)
```

Set the initial start level value that is assigned to a Bundle when it is first installed.

The initial bundle start level will be set to the specified start level. The initial bundle start level value will be persistently recorded by the Framework.

When a Bundle is installed via `BundleContext.installBundle`, it is assigned the initial bundle start level value.

The default initial bundle start level value is 1 unless this method has been called to assign a different initial bundle start level value.

This method does not change the start level values of installed bundles.

**Parameters:**
- startlevel - The initial start level for newly installed bundles.

**Throws:**
- IllegalArgumentException - If the specified start level is less than or equal to zero.
- SecurityException - If the caller does not have AdminPermission[System Bundle,STARTLEVEL] and the Java runtime environment supports permissions.

### isBundlePersistentlyStarted

```java
boolean isBundlePersistentlyStarted(Bundle bundle)
```

Deprecated. As of 1.2. Replaced by `bundle.adapt(BundleStartLevel.class).isPersistentlyStarted()`

Returns whether the specified bundle's autostart setting indicates the bundle must be started.

The autostart setting of a bundle indicates whether the bundle is to be started when its start level is reached.
Interface StartLevel

Parameters:
bundle - The bundle whose autostart setting is to be examined.

Returns:
true if the autostart setting of the bundle indicates the bundle is to be started. false otherwise.

Throws:
IllegalArgumentException - If the specified bundle has been uninstalled or if the specified bundle was not created by the same framework instance that registered this StartLevel service.

See Also:
Bundle.START_TRANSIENT

isBundleActivationPolicyUsed

boolean isBundleActivationPolicyUsed(Bundle bundle)

Deprecated. As of 1.2. Replaced by
bundle.adapt(BundleStartLevel.class).isActivationPolicyUsed()

Returns whether the specified bundle's autostart setting indicates that the activation policy declared in the bundle's manifest must be used.

The autostart setting of a bundle indicates whether the bundle's declared activation policy is to be used when the bundle is started.

Parameters:
bundle - The bundle whose autostart setting is to be examined.

Returns:
true if the bundle's autostart setting indicates the activation policy declared in the manifest must be used. false if the bundle must be eagerly activated.

Throws:
IllegalArgumentException - If the specified bundle has been uninstalled or if the specified bundle was not created by the same framework instance that registered this StartLevel service.

Since: 1.1

See Also:
Bundle.START_ACTIVATION_POLICY


Package org.osgi.util.tracker

Tracker Package Version 1.5.

See: Description

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleTracker Customizer</td>
<td>The BundleTrackerCustomizer interface allows a BundleTracker to customize the Bundles that are tracked.</td>
<td>197</td>
</tr>
<tr>
<td>ServiceTracker Customizer</td>
<td>The ServiceTrackerCustomizer interface allows a ServiceTracker to customize the service objects that are tracked.</td>
<td>207</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleTracker</td>
<td>The BundleTracker class simplifies tracking bundles much like the ServiceTracker simplifies tracking services.</td>
<td>192</td>
</tr>
<tr>
<td>ServiceTracker</td>
<td>The ServiceTracker class simplifies using services from the Framework's service registry.</td>
<td>199</td>
</tr>
</tbody>
</table>

Package org.osgi.util.tracker Description

Tracker Package Version 1.5.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. For example:

Import-Package: org.osgi.util.tracker; version="[1.5,2.0)"
The BundleTracker class simplifies tracking bundles much like the ServiceTracker simplifies tracking services.

A BundleTracker is constructed with state criteria and a BundleTrackerCustomizer object. A BundleTracker can use the BundleTrackerCustomizer to select which bundles are tracked and to create a customized object to be tracked with the bundle. The BundleTracker can then be opened to begin tracking all bundles whose state matches the specified state criteria.

The getBundles method can be called to get the Bundle objects of the bundles being tracked. The getObject method can be called to get the customized object for a tracked bundle.

The BundleTracker class is thread-safe. It does not call a BundleTrackerCustomizer while holding any locks. BundleTrackerCustomizer implementations must also be thread-safe.

Since: 1.4
Version: $Revision: 8320 $
### Field Detail

**context**

protected final BundleContext context

The Bundle Context used by this BundleTracker.

### Constructor Detail

**BundleTracker**

public BundleTracker(BundleContext context,  
int stateMask,  
BundleTrackerCustomizer<T> customizer)

Create a BundleTracker for bundles whose state is present in the specified state mask.

Bundles whose state is present on the specified state mask will be tracked by this BundleTracker.

**Parameters:**

- context - The BundleContext against which the tracking is done.
- stateMask - The bit mask of the ORing of the bundle states to be tracked.
- customizer - The customizer object to call when bundles are added, modified, or removed in this BundleTracker. If customizer is null, then this BundleTracker will be used as the BundleTrackerCustomizer and this BundleTracker will call the BundleTrackerCustomizer methods on itself.

**See Also:**

Bundle.getState()

### Method Detail

**open**

public void open()
Open this BundleTracker and begin tracking bundles.

Bundle which match the state criteria specified when this BundleTracker was created are now tracked by this BundleTracker.

Throws:
- `IllegalStateException` - If the BundleContext with which this BundleTracker was created is no longer valid.
- `SecurityException` - If the caller and this class do not have the appropriate AdminPermission[context bundle,LISTENER], and the Java Runtime Environment supports permissions.

---

close

```java
public void close()
```

Close this BundleTracker.

This method should be called when this BundleTracker should end the tracking of bundles.

This implementation calls `getBundles()` to get the list of tracked bundles to remove.

---

addingBundle

```java
public T addingBundle(Bundle bundle,
                      BundleEvent event)
```

Default implementation of the BundleTrackerCustomizer.addingBundle method.

This method is only called when this BundleTracker has been constructed with a null BundleTrackerCustomizer argument.

This implementation simply returns the specified Bundle.

This method can be overridden in a subclass to customize the object to be tracked for the bundle being added.

Specified by:
- `addingBundle` in interface `BundleTrackerCustomizer`

Parameters:
- `bundle` - The Bundle being added to this BundleTracker object.
- `event` - The bundle event which caused this customizer method to be called or null if there is no bundle event associated with the call to this method.

Returns:
The specified bundle.

See Also:
- `BundleTrackerCustomizer.addingBundle(Bundle, BundleEvent)`

---

modifiedBundle

```java
public void modifiedBundle(Bundle bundle,
                           BundleEvent event,
                           T object)
```

Default implementation of the BundleTrackerCustomizer.modifiedBundle method.
This method is only called when this BundleTracker has been constructed with a null BundleTrackerCustomizer argument.

This implementation does nothing.

**Specified by:**
modifiedBundle in interface BundleTrackerCustomizer

**Parameters:**
- bundle - The Bundle whose state has been modified.
- event - The bundle event which caused this customizer method to be called or null if there is no bundle event associated with the call to this method.
- object - The customized object for the specified Bundle.

**See Also:**
BundleTrackerCustomizer.modifiedBundle(Bundle, BundleEvent, Object)

---

**removedBundle**

```java
public void removedBundle(Bundle bundle, BundleEvent event, T object)
```

Default implementation of the BundleTrackerCustomizer.removedBundle method.

This method is only called when this BundleTracker has been constructed with a null BundleTrackerCustomizer argument.

This implementation does nothing.

**Specified by:**
removedBundle in interface BundleTrackerCustomizer

**Parameters:**
- bundle - The Bundle being removed.
- event - The bundle event which caused this customizer method to be called or null if there is no bundle event associated with the call to this method.
- object - The customized object for the specified bundle.

**See Also:**
BundleTrackerCustomizer.removedBundle(Bundle, BundleEvent, Object)

---

**getBundles**

```java
public Bundle[] getBundles()
```

Return an array of Bundles for all bundles being tracked by this BundleTracker.

**Returns:**
An array of Bundles or null if no bundles are being tracked.

---

**getObject**

```java
public T getObject(Bundle bundle)
```

Returns the customized object for the specified Bundle if the specified bundle is being tracked by this BundleTracker.

**Parameters:**
- bundle - The Bundle being tracked.
Class BundleTracker

Returns:
The customized object for the specified Bundle or null if the specified Bundle is not being tracked.

remove

public void remove(Bundle bundle)

Remove a bundle from this BundleTracker. The specified bundle will be removed from this BundleTracker. If the specified bundle was being tracked then the BundleTrackerCustomizer.removedBundle method will be called for that bundle.

Parameters:
   bundle - The Bundle to be removed.

size

public int size()

Return the number of bundles being tracked by this BundleTracker.

Returns:
The number of bundles being tracked.

getTrackingCount

public int getTrackingCount()

Returns the tracking count for this BundleTracker. The tracking count is initialized to 0 when this BundleTracker is opened. Every time a bundle is added, modified or removed from this BundleTracker the tracking count is incremented.

The tracking count can be used to determine if this BundleTracker has added, modified or removed a bundle by comparing a tracking count value previously collected with the current tracking count value. If the value has not changed, then no bundle has been added, modified or removed from this BundleTracker since the previous tracking count was collected.

Returns:
The tracking count for this BundleTracker or -1 if this BundleTracker is not open.

getTracked

public Map<Bundle, T> getTracked()

Return a Map with the BundleS and customized objects for all bundles being tracked by this BundleTracker.

Returns:
A Map with the BundleS and customized objects for all services being tracked by this BundleTracker. If no bundles are being tracked, then the returned map is empty.

Since:
1.5
Interface BundleTrackerCustomizer

org.osgi.util.tracker

Type Parameters:
  T - The type of the tracked object.

All Known Implementing Classes:
  BundleTracker

public interface BundleTrackerCustomizer

The BundleTrackerCustomizer interface allows a BundleTracker to customize the Bundles that are tracked. A BundleTrackerCustomizer is called when a bundle is being added to a BundleTracker. The BundleTrackerCustomizer can then return an object for the tracked bundle. A BundleTrackerCustomizer is also called when a tracked bundle is modified or has been removed from a BundleTracker.

The methods in this interface may be called as the result of a BundleEvent being received by a BundleTracker. Since BundleEvents are received synchronously by the BundleTracker, it is highly recommended that implementations of these methods do not alter bundle states while being synchronized on any object.

The BundleTracker class is thread-safe. It does not call a BundleTrackerCustomizer while holding any locks. BundleTrackerCustomizer implementations must also be thread-safe.

Since: 1.4
Version: $Revision: 8177 $
ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>T addingBundle(Bundle bundle, BundleEvent event)</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td>A bundle is being added to the BundleTracker.</td>
</tr>
<tr>
<td>void modifiedBundle(Bundle bundle, BundleEvent event, T object)</td>
<td>198</td>
</tr>
<tr>
<td></td>
<td>A bundle tracked by the BundleTracker has been modified.</td>
</tr>
<tr>
<td>void removedBundle(Bundle bundle, BundleEvent event, T object)</td>
<td>198</td>
</tr>
<tr>
<td></td>
<td>A bundle tracked by the BundleTracker has been removed.</td>
</tr>
</tbody>
</table>

Method Detail

addingBundle

T addingBundle(Bundle bundle, BundleEvent event)

A bundle is being added to the BundleTracker.

This method is called before a bundle which matched the search parameters of the BundleTracker is added to the BundleTracker. This method should return the object to be tracked for the specified Bundle. The returned object is stored in the BundleTracker and is available from the getObject method.

Parameters:
  bundle - The Bundle being added to the BundleTracker.
  event - The bundle event which caused this customizer method to be called or null if there is no bundle event associated with the call to this method.
Interface BundleTrackerCustomizer

Returns:
The object to be tracked for the specified Bundle object or null if the specified Bundle object should not be tracked.

modifiedBundle

void modifiedBundle(Bundle bundle,
BundleEvent event,
T object)

A bundle tracked by the BundleTracker has been modified.

This method is called when a bundle being tracked by the BundleTracker has had its state modified.

Parameters:
bundle - The Bundle whose state has been modified.
event - The bundle event which caused this customizer method to be called or null if there is no bundle event associated with the call to this method.
object - The tracked object for the specified bundle.

removedBundle

void removedBundle(Bundle bundle,
BundleEvent event,
T object)

A bundle tracked by the BundleTracker has been removed.

This method is called after a bundle is no longer being tracked by the BundleTracker.

Parameters:
bundle - The Bundle that has been removed.
event - The bundle event which caused this customizer method to be called or null if there is no bundle event associated with the call to this method.
object - The tracked object for the specified bundle.
Class ServiceTracker

org.osgi.util.tracker

java.lang.Object
  org.osgi.util.tracker.ServiceTracker

Type Parameters:
  s - The type of the service being tracked.
  t - The type of the tracked object.

All Implemented Interfaces:
  ServiceTrackerCustomizer<S,T>

public class ServiceTracker
  extends Object
  implements ServiceTrackerCustomizer<S,T>

The ServiceTracker class simplifies using services from the Framework's service registry.

A ServiceTracker object is constructed with search criteria and a ServiceTrackerCustomizer object. A ServiceTracker can use a ServiceTrackerCustomizer to customize the service objects to be tracked. The ServiceTracker can then be opened to begin tracking all services in the Framework's service registry that match the specified search criteria. The ServiceTracker correctly handles all of the details of listening to ServiceEvent$ and getting and ungetting services.

The getServiceReferences method can be called to get references to the services being tracked. The getService and getServices methods can be called to get the service objects for the tracked service.

The ServiceTracker class is thread-safe. It does not call a ServiceTrackerCustomizer while holding any locks. ServiceTrackerCustomizer implementations must also be thread-safe.

Version:
  $Revision: 8344 $

ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>protected BundleContext context</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Bundle Context used by this ServiceTracker.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected Filter filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Filter used by this ServiceTracker which specifies the search criteria for the services to track.</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>ServiceTracker(BundleContext context, Class&lt;s&gt; clazz, ServiceTrackerCustomizer&lt;s,t&gt; customizer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a ServiceTracker on the specified class.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ServiceTracker(BundleContext context, String clazz, ServiceTrackerCustomizer&lt;s,t&gt; customizer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a ServiceTracker on the specified class name.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ServiceTracker(BundleContext context, Filter filter, ServiceTrackerCustomizer&lt;s,t&gt; customizer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a ServiceTracker on the specified Filter object.</td>
</tr>
</tbody>
</table>
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addingService(ServiceReference&lt;S&gt; reference)</code></td>
<td>Default implementation of the <code>ServiceTrackerCustomizer.addingService</code> method.</td>
</tr>
<tr>
<td><code>close()</code></td>
<td>Close this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>getService()</code></td>
<td>Returns a service object for one of the services being tracked by this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>getService(ServiceReference&lt;S&gt; reference)</code></td>
<td>Returns the service object for the specified <code>ServiceReference</code> if the specified referenced service is being tracked by this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>getServiceReference()</code></td>
<td>Returns a <code>ServiceReference</code> for one of the services being tracked by this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>getServiceReferences()</code></td>
<td>Return an array of <code>ServiceReference</code> for all services being tracked by this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>getServices()</code></td>
<td>Return an array of service objects for all services being tracked by this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>getTracked()</code></td>
<td>Return a <code>SortedMap</code> of the <code>ServiceReference</code>s and service objects for all services being tracked by this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>getTrackingCount()</code></td>
<td>Returns the tracking count for this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>modifiedService(ServiceReference&lt;S&gt; reference, T service)</code></td>
<td>Default implementation of the <code>ServiceTrackerCustomizer.modifiedService</code> method.</td>
</tr>
<tr>
<td><code>open()</code></td>
<td>Open this <code>ServiceTracker</code> and begin tracking services.</td>
</tr>
<tr>
<td><code>open(boolean trackAllServices)</code></td>
<td>Open this <code>ServiceTracker</code> and begin tracking services.</td>
</tr>
<tr>
<td><code>remove(ServiceReference&lt;S&gt; reference)</code></td>
<td>Remove a service from this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>removedService(ServiceReference&lt;S&gt; reference, T service)</code></td>
<td>Default implementation of the <code>ServiceTrackerCustomizer.removedService</code> method.</td>
</tr>
<tr>
<td><code>size()</code></td>
<td>Return the number of services being tracked by this <code>ServiceTracker</code>.</td>
</tr>
<tr>
<td><code>waitForService(long timeout)</code></td>
<td>Wait for at least one service to be tracked by this <code>ServiceTracker</code>.</td>
</tr>
</tbody>
</table>

### Field Detail

**context**

protected final `BundleContext` context

The Bundle Context used by this `ServiceTracker`. 
Class ServiceTracker

protected final Filter filter

The Filter used by this ServiceTracker which specifies the search criteria for the services to track.

Since: 1.1

Constructor Detail

ServiceTracker

public ServiceTracker(BundleContext context,
   ServiceReference<S> reference,
   ServiceTrackerCustomizer<S, T> customizer)

Create a ServiceTracker on the specified ServiceReference.

The service referenced by the specified ServiceReference will be tracked by this ServiceTracker.

Parameters:
   context - The BundleContext against which the tracking is done.
   reference - The ServiceReference for the service to be tracked.
   customizer - The customizer object to call when services are added, modified, or removed in this ServiceTracker. If customizer is null, then this ServiceTracker will be used as the ServiceTrackerCustomizer and this ServiceTracker will call the ServiceTrackerCustomizer methods on itself.

ServiceTracker

public ServiceTracker(BundleContext context,
   String clazz,
   ServiceTrackerCustomizer<S, T> customizer)

Create a ServiceTracker on the specified class name.

Services registered under the specified class name will be tracked by this ServiceTracker.

Parameters:
   context - The BundleContext against which the tracking is done.
   clazz - The class name of the services to be tracked.
   customizer - The customizer object to call when services are added, modified, or removed in this ServiceTracker. If customizer is null, then this ServiceTracker will be used as the ServiceTrackerCustomizer and this ServiceTracker will call the ServiceTrackerCustomizer methods on itself.

ServiceTracker

public ServiceTracker(BundleContext context,
   Filter filter,
   ServiceTrackerCustomizer<S, T> customizer)

Create a ServiceTracker on the specified Filter object.

Services which match the specified Filter object will be tracked by this ServiceTracker.
Class ServiceTracker

Parameters:
- context - The BundleContext against which the tracking is done.
- filter - The Filter to select the services to be tracked.
- customizer - The customizer object to call when services are added, modified, or removed in this ServiceTracker. If customizer is null, then this ServiceTracker will be used as the ServiceTrackerCustomizer and this ServiceTracker will call the ServiceTrackerCustomizer methods on itself.

Since:
1.1

ServiceTracker

public ServiceTracker(BundleContext context,
                      Class<S> clazz,
                      ServiceTrackerCustomizer<S,T> customizer)

Create a ServiceTracker on the specified class.

Services registered under the name of the specified class will be tracked by this ServiceTracker.

Parameters:
- context - The BundleContext against which the tracking is done.
- clazz - The class of the services to be tracked.
- customizer - The customizer object to call when services are added, modified, or removed in this ServiceTracker. If customizer is null, then this ServiceTracker will be used as the ServiceTrackerCustomizer and this ServiceTracker will call the ServiceTrackerCustomizer methods on itself.

Since:
1.5

Method Detail

open

public void open()

Open this ServiceTracker and begin tracking services.

This implementation calls open(false).

Throws:
- IllegalStateException - If the BundleContext with which this ServiceTracker was created is no longer valid.

See Also:
- open(boolean)

open

public void open(boolean trackAllServices)

Open this ServiceTracker and begin tracking services.

Services which match the search criteria specified when this ServiceTracker was created are now tracked by this ServiceTracker.

Parameters:
- trackAllServices - If true, then this ServiceTracker will track all matching services regardless of class loader accessibility. If false, then this ServiceTracker will only track matching services
which are class loader accessible to the bundle whose BundleContext is used by this ServiceTracker.

**Throws:**
- `IllegalStateException` - If the BundleContext with which this ServiceTracker was created is no longer valid.

**Since:**
1.3

---

### close

**public void close()**

Close this ServiceTracker.

This method should be called when this ServiceTracker should end the tracking of services.

This implementation calls `getserviceReferences()` to get the list of tracked services to remove.

### addingService

**public T addingService(ServiceReference<S> reference)**

Default implementation of the ServiceTrackerCustomizer.addingService method.

This method is only called when this ServiceTracker has been constructed with a null ServiceTrackerCustomizer argument.

This implementation returns the result of calling `getService` on the BundleContext with which this ServiceTracker was created passing the specified ServiceReference.

This method can be overridden in a subclass to customize the service object to be tracked for the service being added. In that case, take care not to rely on the default implementation of `removedService` to unget the service.

**Specified by:**
- `addingService` in interface `ServiceTrackerCustomizer`

**Parameters:**
- `reference` - The reference to the service being added to this ServiceTracker.

**Returns:**
- The service object to be tracked for the service added to this ServiceTracker.

**See Also:**
- `ServiceTrackerCustomizer.addingService(ServiceReference)`

---

### modifiedService

**public void modifiedService(ServiceReference<S> reference, T service)**

Default implementation of the ServiceTrackerCustomizer.modifiedService method.

This method is only called when this ServiceTracker has been constructed with a null ServiceTrackerCustomizer argument.

This implementation does nothing.

**Specified by:**
- `modifiedService` in interface `ServiceTrackerCustomizer`
Parameters:
- reference - The reference to modified service.
- service - The service object for the modified service.

See Also:
- ServiceTrackerCustomizer.modifiedService(ServiceReference, Object)

removedService

public void removedService(ServiceReference<S> reference, T service)

Default implementation of the ServiceTrackerCustomizer.removedService method.

This method is only called when this ServiceTracker has been constructed with a null ServiceTrackerCustomizer argument.

This implementation calls ungetService, on the BundleContext with which this ServiceTracker was created, passing the specified ServiceReference.

This method can be overridden in a subclass. If the default implementation of addingService method was used, this method must unget the service.

Specified by:
- removedService in interface ServiceTrackerCustomizer

Parameters:
- reference - The reference to removed service.
- service - The service object for the removed service.

See Also:
- ServiceTrackerCustomizer.removedService(ServiceReference, Object)

waitForService

public T waitForService(long timeout) throws InterruptedException

Wait for at least one service to be tracked by this ServiceTracker. This method will also return when this ServiceTracker is closed.

It is strongly recommended that waitForService is not used during the calling of the BundleActivator methods. BundleActivator methods are expected to complete in a short period of time.

This implementation calls getService() to determine if a service is being tracked.

Parameters:
- timeout - The time interval in milliseconds to wait. If zero, the method will wait indefinitely.

Returns:
- Returns the result of getService().

Throws:
- InterruptedException - If another thread has interrupted the current thread.
- IllegalArgumentException - If the value of timeout is negative.

getServiceReferences

public ServiceReference<S>[] getServiceReferences()

Return an array of ServiceReferences for all services being tracked by this ServiceTracker.
Returns:
Array of ServiceReferences or null if no services are being tracked.

---

**getServiceReference**

```java
class ServiceTracker

public ServiceReference<\> getServiceReference()
```

Returns a ServiceReference for one of the services being tracked by this ServiceTracker.

If multiple services are being tracked, the service with the highest ranking (as specified in its service.ranking property) is returned. If there is a tie in ranking, the service with the lowest service ID (as specified in its service.id property); that is, the service that was registered first is returned. This is the same algorithm used by BundleContext.getServiceReference.

This implementation calls `getServiceReferences()` to get the list of references for the tracked services.

Returns:
A ServiceReference or null if no services are being tracked.

Since:
1.1

---

**getService**

```java
class ServiceTracker

public \ getService(ServiceReference<\> reference)
```

Returns the service object for the specified ServiceReference if the specified referenced service is being tracked by this ServiceTracker.

Parameters:
reference - The reference to the desired service.

Returns:
A service object or null if the service referenced by the specified ServiceReference is not being tracked.

---

**getServices**

```java
class ServiceTracker

public Object[] getServices()
```

Return an array of service objects for all services being tracked by this ServiceTracker.

This implementation calls `getServiceReferences()` to get the list of references for the tracked services and then calls `getService(ServiceReference)` for each reference to get the tracked service object.

Returns:
An array of service objects or null if no services are being tracked.

---

**getService**

```java
class ServiceTracker

public \ getService()
```

Returns a service object for one of the services being tracked by this ServiceTracker.

If any services are being tracked, this implementation returns the result of calling `getService(getServiceReference())`. 
Returns:
A service object or null if no services are being tracked.

remove
public void remove(ServiceReference<S> reference)
Remove a service from this ServiceTracker. The specified service will be removed from this
ServiceTracker. If the specified service was being tracked then the
ServiceTrackerCustomizer.removedService method will be called for that service.

Parameters:
reference - The reference to the service to be removed.

size
public int size()
Return the number of services being tracked by this ServiceTracker.

Returns:
The number of services being tracked.

getTrackingCount
public int getTrackingCount()
Returns the tracking count for this ServiceTracker. The tracking count is initialized to 0 when this
ServiceTracker is opened. Every time a service is added, modified or removed from this
ServiceTracker, the tracking count is incremented.

The tracking count can be used to determine if this ServiceTracker has added, modified or removed a
service by comparing a tracking count value previously collected with the current tracking count value. If
the value has not changed, then no service has been added, modified or removed from this
ServiceTracker since the previous tracking count was collected.

Returns:
The tracking count for this ServiceTracker or -1 if this ServiceTracker is not open.
Since:
1.2

getTracked
public SortedMap<ServiceReference<S>,T> getTracked()
Return a SortedMap of the ServiceReferences and service objects for all services being tracked by this
ServiceTracker. The map is sorted in reverse natural order of ServiceReference. That is, the first entry
is the service with the highest ranking and the lowest service id.

Returns:
A SortedMap with the ServiceReferences and service objects for all services being tracked by this
ServiceTracker. If no services are being tracked, then the returned map is empty.
Since:
1.5
Interface ServiceTrackerCustomizer

Type Parameters:

S - The type of the service being tracked.
T - The type of the tracked object.

All Known Implementing Classes:

ServiceTracker

public interface ServiceTrackerCustomizer

The ServiceTrackerCustomizer interface allows a ServiceTracker to customize the service objects that are tracked. A ServiceTrackerCustomizer is called when a service is being added to a ServiceTracker. The ServiceTrackerCustomizer can then return an object for the tracked service. A ServiceTrackerCustomizer is also called when a tracked service is modified or has been removed from a ServiceTracker.

The methods in this interface may be called as the result of a ServiceEvent being received by a ServiceTracker. Since ServiceEvents are synchronously delivered by the Framework, it is highly recommended that implementations of these methods do not register (BundleContext.registerService), modify (ServiceRegistration.setProperties) or unregister (ServiceRegistration.unregister) a service while being synchronized on any object.

The ServiceTracker class is thread-safe. It does not call a ServiceTrackerCustomizer while holding any locks. ServiceTrackerCustomizer implementations must also be thread-safe.

Version: $Revision: 8177 $

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addingService</td>
<td>207</td>
</tr>
<tr>
<td>modifiedService</td>
<td>208</td>
</tr>
<tr>
<td>removedService</td>
<td>208</td>
</tr>
</tbody>
</table>

Method Detail

addingService

addingService(ServiceReference<S> reference)

A service is being added to the ServiceTracker.

This method is called before a service which matched the search parameters of the ServiceTracker is added to the ServiceTracker. This method should return the service object to be tracked for the specified ServiceReference. The returned service object is stored in the ServiceTracker and is available from the getService and getServices methods.

Parameters:

reference - The reference to the service being added to the ServiceTracker.
**Interface ServiceTrackerCustomizer**

**Returns:**
The service object to be tracked for the specified referenced service or null if the specified referenced service should not be tracked.

---

**modifiedService**

```java
void modifiedService(ServiceReference<S> reference,
                      T service)
```

A service tracked by the ServiceTracker has been modified.

This method is called when a service being tracked by the ServiceTracker has had its properties modified.

**Parameters:**
- reference: The reference to the service that has been modified.
- service: The service object for the specified referenced service.

---

**removedService**

```java
void removedService(ServiceReference<S> reference,
                     T service)
```

A service tracked by the ServiceTracker has been removed.

This method is called after a service is no longer being tracked by the ServiceTracker.

**Parameters:**
- reference: The reference to the service that has been removed.
- service: The service object for the specified referenced service.

---

## 7 Considered Alternatives

### 7.1 Version 2 API

Originally the plan was to do version 2 of the API with breaking API changes. This would have included use of enums and potentially annotations. The JavaOne 2009 presentation was predicated on this idea and proposed a thunking layer to simultaneously support both API versions. Subsequent soul searching and discussions lead to the conclusion that it would be very difficult to move the bundles over to use the version 2 API. We would be far better off with a more limited exploitation of Java 5 language features and preserving backwards compatibility as well as continuing support for embedded users.

### 7.2 Retroweaving

Retroweaving was also considered as a means to provide support for embedded users while exploiting Java 5 language features. However, retroweaving has several issues. Many of the Java 5 language features require class library support. So retroweavers must supply alternate class libraries and modify the woven code to access their
class libraries. These class libraries would then need to be made available at runtime in the OSGi environment which would necessitate proper configuration of things like bootclasspath and bootdelegation. We would also be at the mercy of the correctness of these implementations.

We would also be in the position of supporting/supplying 2 versions of the companion code jars. One compiled for Java 1.4 and one compiled for Java 5.

### 7.3 Moving all the PackageAdmin and StartLevel API into the Framework API.

The initial draft of the RFC has all the PackageAdmin and StartLevel function moved into the Framework API. This approach was rejected at the Portland f2f and the adapt approach was agreed to instead.

### 7.4 Removed BundleAdapter interface

After discussion at the Southampton f2f, we agreed to remove the BundleAdapter interface from the Bundle.adapt signature. This allows Bundle to be adapt to other types which do not extend BundleAdapter. For example, bundle.adapt(ProtectionDomain.class).

---

### 8 Security Considerations

There are no additional security implications raised by this RFC.

---

### 9 Document Support

#### 9.1 References


#### 9.2 Author’s Address
<table>
<thead>
<tr>
<th>Name</th>
<th>BJ Hargrave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>IBM Corporation</td>
</tr>
<tr>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>Voice</td>
<td></td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:hargrave@us.ibm.com">hargrave@us.ibm.com</a></td>
</tr>
</tbody>
</table>

### 9.3 Acronyms and Abbreviations

### 9.4 End of Document
RFC 154 - Generic Capabilities & Requirements

Abstract
This document proposes an approach for bundles to declare generic capabilities and for other bundles specify requirements on those capabilities.
0 Document Information

0.1 Table of Contents

0 Document Information

0.1 Table of Contents.................................................................................................................. 2
0.2 Terminology and Document Conventions............................................................................. 3
0.3 Revision History..................................................................................................................... 3

1 Introduction.............................................................................................................................. 3

2 Application Domain.................................................................................................................. 4

3 Problem Description................................................................................................................ 4

3.1 Extender bundle dependencies............................................................................................ 4
3.2 Operating environment dependencies (Bug 282).................................................................... 4
3.3 Limitations of Bundle-RequiredExecutionEnvironment.......................................................... 5
3.3.1 Inability to capture java.* dependencies (Bug 390)........................................................... 5
3.3.2 Lack of expressiveness (Bug 1000)...................................................................................... 5
3.4 Native code dependencies (Bug 1013).................................................................................... 5

4 Requirements............................................................................................................................ 6

5 Technical Solution..................................................................................................................... 6

5.1 General approach................................................................................................................... 6
5.2 Optionality.............................................................................................................................. 7
5.3 Effective time......................................................................................................................... 7
5.4 Management Agents............................................................................................................. 7
5.5 Resolution result................................................................................................................... 8
5.6 Multiple matching providers.................................................................................................. 8
5.7 Cardinality of requirements................................................................................................. 8
5.8 Capability/Requirement verification...................................................................................... 9
5.9 Host/fragment conflicts......................................................................................................... 9

6 Considered Alternatives........................................................................................................... 9

6.1 Discovery................................................................................................................................ 9
6.2 Uses constraints...................................................................................................................... 9
6.3 Singleton capabilities.......................................................................................................... 9
6.4 Mappings for existing headers............................................................................................ 9
6.5 Bundle state related capabilities.......................................................................................... 10
### 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 8.1.

*Source code is shown in this typeface.*

### 0.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>Dec. 14th, 2009</td>
<td>Adopted typed attribute approach. Resolve result is now observable with lifecycle coupling. Added wording on multiple matching providers, capability/requirement verification, and existing header mappings.</td>
</tr>
<tr>
<td>Southampton F2F feedback</td>
<td>Mar. 1st, 2010</td>
<td>Added framework property that controls which capability namespaces get processed by the framework and commentary on bundle state related capabilities.</td>
</tr>
<tr>
<td>Bundle state feedback</td>
<td>Mar 16th 2010</td>
<td>Added framework property that controls which capability namespaces get processed by the framework and commentary on bundle state related capabilities.</td>
</tr>
</tbody>
</table>

---

Copyright © Sun Microsystems 2010  All Rights Reserved
1 Introduction

The OSGi specification defines a handful of different dependency types that may exist among bundles, such as package, bundle, and fragment dependencies. All of these dependencies relate to defining code visibility for the involved bundles. The OSGi framework is able to consistently resolve these dependencies to ensure a given bundle’s code dependencies are satisfied so it can be used successfully. While this approach is certainly important and worthwhile, it does have one significant drawback.

Not all dependencies among bundles are related to code visibility. A prime example are bundles requiring extenders. Typically, they have no code dependency on the extender bundle, but they do require the extender bundle to be present in order to function. Currently, there is no proper way to express such dependencies. It would be worthwhile if bundles could declare that they provide some arbitrary capability and other bundles could declare requirements on those capabilities.

This RFC investigates this issue and proposes a simple manifest-based metadata approach for declaring dependencies among bundles for reasons other than code visibility.

2 Application Domain

This section should be copied from the appropriate RFP(s). It is repeated here so it can be extended while the RFC authors learn more subtle details.

3 Problem Description

While bundles are able to express a variety of code-related dependencies, not all bundle dependencies are related to code. Ultimately, this leads to situations where a bundle is unable to properly function, even though its code-related dependencies have been properly satisfied. Defining a standard way to express non-code-related dependencies will help eliminate such situations. The following use cases illustrate where such a mechanism could be applicable.
3.1 Extender bundle dependencies

The extender pattern is a common OSGi pattern for providing some functionality or performing some task for unknowing bundles. An extender bundle listens for bundle lifecycle events, probes the associated bundles for metadata, then performs some activity on behalf of the bundle or some other purpose. The extendee bundle may have been written to explicitly expect the processing of the extender and may not be able to properly function without it. If the extendee bundle does not have a code dependency on the extender, then it has no way to express this requirement.

3.2 Operating environment dependencies (Bug 282)

It is often required to have one configuration of bundles run in multiple different operating environments. Here an operating environment is comprised largely of the OS, windowing system, locale, and processor architecture. In certain situations, such as on Unix systems that support GTK and Motif or when the user wants to run the same configuration on a different machine, these capabilities can vary greatly.

Currently, bundles are unable to state the operating environment in which they are valid, this is left to an external management agent to uninstall/reinstall bundles as appropriate. This sets up a need to communicate somehow with an agent on startup or requires managing multiple configurations. It also makes it difficult for provisioning agents to know which bundles are valid for which platforms.

3.3 Limitations of Bundle-RequiredExecutionEnvironment

3.3.1 Inability to capture java.* dependencies (Bug 390)

Sometimes bundle execution environment dependencies do not fall neatly into the defined set of execution environments. For example, consider the follow situations: a bundle that bundle limits itself to the APIs available in the Foundation 1.0 execution environment, but it finds it also needs the java.beans package. While it is possible to install a boot class path extension bundle to supply java.beans, the bundle in question has no way to indicate that it needs the extension bundle in the first place.

3.3.2 Lack of expressiveness (Bug 1000)

In some cases it may make sense to prohibit a bundle on a higher level VM. Currently, this is not possible with BREE because it specifies a list of EEs that the bundle is compatible with. For example:

    Bundle-RequiredExecutionEnvironment: J2SE-1.4

This means the bundle requires an EE that is compatible with J2SE-1.4. This includes J2SE-1.5, JavaSE-1.6 and all future VMs that will be compatible with J2SE-1.4. In some cases, it may make sense to provide a range limit to the EEs that the bundle can execute on. Something like this:

    Bundle-RequiredExecutionEnvironment: [J2SE-1.3, J2SE-1.5)

This would specify that the bundle can execute on an EE that is compatible with J2SE-1.3, J2SE-1.4 but can not execute on an EE that is compatible with J2SE-1.5 or higher.

3.4 Native code dependencies (Bug 1013)

The OSGi framework supports direct bundle dependencies on native code, but it is unable to deal with dependencies from one native library to another. While there is not a perfect solution to this issue, one approach to lessen the impact would be to devise a new system bundle fragment which could install native libraries in the LD_LIBRARY_PATH so that dependencies between native libraries could be resolved. However, a bundle would
need some way to express a dependency on such native system bundle fragments if it were required for it to function properly.

## 4 Requirements

*This section should be copied from the appropriate RFP(s)*

## 5 Technical Solution

This RFC proposes additional manifest headers for bundles to declare arbitrary provided capabilities and requirements on them. In the abstract, these headers impact resolution of the bundles involved, but have no other impact on them. To state this in another way using an analogy, while the resolver can be seen as injecting code visibility wires into a bundle or a service dependency framework can be seen as injecting services into a component, the mechanism proposed in this document does not have any associated “injection” phase. It is predicated solely on the existence of declared capabilities to satisfy declared requirements and nothing more.

### 5.1 General approach

The general approach uses manifest-based syntax to allow a bundle to declare arbitrary capabilities and requirements. A capability is a set of attributes associated with a given namespace, where the namespace is simply a string name used to scope attribute names to avoid name clashes and convey semantics. For example, in OBR's use of generic capabilities and requirements, it uses the “package” and “bundle” namespaces to map package and bundle code dependencies, respectively.

The approach for declaring a dependency uses the common OSGi manifest header syntax, since it avoids the need for new parsing code. To declare a capability, the `Provide-Capability` header is used, which has the following syntax:

```
Provide-Capability ::= capability ( ',' capability ) *
capability ::= namespace ( ';' typed-attr ) * ( ';' untyped-attr ) *
typed-attr ::= token ':' attr-type '=' argument
untyped-attr ::= token '=' argument
attr-type ::= ['string' | 'version' | 'long' | 'double']
```

With an example capability taking the form of:

```
Provide-Capability: foo-extender; version:version="1.1.0"; secure=false
```
An untyped attribute defaults to string type.  

To require a generic capability, the Require-Capability header is used, which has the following syntax:

\[
\text{Require-Capability} \ ::= \ \text{requirement} \ (',', \ \text{requirement}) \ *
\]
\[
\text{requirement} \ ::= \ \text{namespace} \ (';' \ \text{resolution-directive})
\]
\[
\text{resolution-directive} \ ::= \ 'resolution' \ ':=' \ (\ 'mandatory' \mid \ 'optional')
\]
\[
\text{filter-attribute} \ ::= \ 'filter' \ '=' \ ldap-filter
\]

For an example, consider a requirement for some imaginary extender bundle:

\[
\text{Require-Capability}: \text{foo-extender};
\]
\[
\text{filter}="(\&(version>=1.0.0)(!(version>=2.0.0))(secure=true))"
\]

For this imaginary example, the requirement would not match the capability. Even though the extender is within the required version range, it does not provide a secure capability.

### 5.2 Optionality

By default, all requirements are mandatory. However, Require-Capability does support the resolution directive which can be used to declare a requirement as optional.

### 5.3 Effective time

The effective time of generic requirement resolution is at bundle resolve time, the same as for any of the existing code dependencies. The resolver algorithm only runs at resolve time, except in the cases of dynamically imported packages. In the case of generic capabilities and requirements, there does not appear to be an analogue to dynamically imported packages.

In reality, some provided and required capabilities only apply to bundles in the active state. Use case 3.1, concerning extender capabilities is such an example. This is because an extendee bundle only requires an extender capability when it is active. Similarly an extender bundle only provides an extender capability when it is active. No attempt is made to address this at framework level because to do so would greatly increase the scope and complexity of the framework resolver.

The main benefit of resolving this sort of dependency at bundle resolve time is that resolution will fail with a meaningful error message if there is no provider of the required capability.

On the other hand, this approach has some unfortunate limitations. For example, in the extender example:

- An extender bundle will only resolve if the extender and all of its dependencies can also be resolved, even though it has no need for the extender at this time. In the case where the extendee is only being resolved to get access to some package it exports the resolution of its extender is not just early, but unnecessary.

- After resolving an extender bundle the extender bundle will be present, but not active, so it won't be providing the extender capability. Before it can do so the operator will have to figure out which bundle is the extender and start it.
5.4 Management Agents

As mentioned in section 5.3, for the framework, the effective time of all capabilities is bundle resolution time, and this has some downsides for capabilities where bundle state matters.

Although addressing this in a standardized way at framework level would be complex, or at least premature, management agents that deal with bundle provisioning should not be prevented from innovating in this area.

For example, a management agent may know that a resolved extendee bundle has no need for a resolved extender bundle, and may also know that before the extendee is activated the extender should also be active.

In this case the management agent will want to take responsibility for handling the extender capability and would like the framework to ignore it. To this end the solution will make use of a framework property that tells the framework which capability namespaces it should ignore, as follows.

\[ \text{org.osgi.framework.processcapabilites} = '-*'|*|(-)?namespace(,namespace)* \]

Where namespace is a generic capability namespace.

If the value is * then the framework will process all namespaces. This is the default.

In the absence of a leading '-', the list comprises the namespaces that the framework should process. All others should be ignored.

If the value is '-*' then the framework should ignore all namespaces.

If the value has a leading '-' then the listed namespaces should be ignored by the framework, but all others should be processed.

5.5 Resolution result

The result of resolving generic capabilities and requirements is a set of wires. A wire connects a given requirement to its satisfying provider(s). The wiring result does not imply any sort of additional visibility or grant any new functionality between the provider and requirers. It does, however, tie their life cycles together like normal package wiring. This means if the provider is refreshed, then requiring bundles are refreshed as well.

### We will need to propose some sort of service API for inspecting wiring

5.6 Multiple matching providers

If multiple providers match a given requirement, then similar rules as for multiple packages providers are followed. For example:

1. A resolved provider is preferred over an unresolved one.
2. Lower bundle identifier is preferred over a higher one.

A difference is that we do not assume anything about versioning for generic providers.

5.7 Cardinality of requirements

All generic requirements are assumed to match only a single provider.
### Since the resolver result is observable now, are we going to allow aggregate cardinality?

#### 5.8 Capability/Requirement verification

Unlike other headers (e.g., `Import-Package` and `Export-Package`) that have various verification rules associated with them (i.e., cannot duplicate or have certain attributes), generic capabilities/requirements have no such verification phase. So, for example, there is no way to say a bundle can only provide a given capability once or require overlapping providers.

#### 5.9 Host/fragment conflicts

For code-related dependencies, fragment capabilities and requirements are merged with the host as long as they do not conflict. It is not clear how to detect a conflict for generic capabilities/requirements; thus, it is assumed that generic capabilities/requirements supplied by a fragment never conflict with those of a host.

### In the observable result, is the end point the host or the fragment for fragment-supplied capabilities?

---

### 6 Considered Alternatives

#### 6.1 Discovery

OBR has demonstrated the potential usefulness of using generic requirements to aide in the discovery of interesting or related functionality. OBR refers to this as an "extends" relationship, which indicates that a given resource extends another target resource, although the target resource is unaware of it. Currently in OBR, this is simply represented as a requirement with a boolean "extends" flag. Having such a concept makes it possible for an OBR resolver implementation to proactively discover related resources and to implement policies around such extensions. It would be possible to support such a concept here with a directive on `require-capability`.

#### 6.2 Uses constraints

It is not clear if "uses" constraints can be applied in a generic context.

#### 6.3 Singleton capabilities

Yuck. We don't support them.

#### 6.4 Mappings for existing headers

The existing bundle manifest headers could easily be mapped to this generic approach and in the case of OBR, they will be. Currently, the framework will not consider such replacements for the existing headers.

### Should a standard mapping be defined here?
6.5 **Bundle state related capabilities.**

Framework level handling of bundle state issues was ruled out on complexity grounds. Whether it is possible and desirable is an open question.

One option was to define a set of well known capability attributes that indicate the bundle state in which capability is available.

One option was to drop active state related capabilities from this specification altogether and make a statement that it should not be used to model these. The view seems to be that the informative resolution time error message makes modeling them in this way worthwhile.

---

7 **Security Considerations**

Description of all known vulnerabilities this may either introduce or address as well as scenarios of how the weaknesses could be circumvented.

---

8 **Document Support**

8.1 **References**


Add references simply by adding new items. You can then cross-refer to them by choosing `<Insert><Cross Reference><Numbered Item>` and then selecting the paragraph. **STATIC REFERENCES (I.E. BODGED) ARE NOT ACCEPTABLE, SOMEONE WILL HAVE TO UPDATE THEM LATER, SO DO IT PROPERLY NOW.**

8.2 **Author's Address**
8.3 Acronyms and Abbreviations

8.4 End of Document
The OSGi Alliance and its members specify, create, advance, and promote wide industry adoption of an open delivery and management platform for application services in home, commercial buildings, automotive and industrial environments. The OSGi Alliance serves as the focal point for a collaborative ecosystem of service providers, developers, manufacturers, and consumers. The OSGi specifications define a standardized, component oriented, computing environment for networked services. OSGi technology is currently being delivered in products and services shipping from several Fortune 100 companies. The OSGi Alliance’s horizontal software integration platform is ideal for both vertical and cross-industry business models within home, vehicle, mobile and industrial environments. As an independent non-profit corporation, the OSGi Alliance also provides for the fair and uniform creation and distribution of relevant intellectual property – including specifications, reference implementations, and test suites – to all its members.

HOW TO REACH US:

OSGi Alliance
Bishop Ranch 6
2400 Camino Ramon, Suite 375
San Ramon, CA 94583 USA

Phone: +1.925.275.6625
E-mail: marketinginfo@osgi.org
Web: http://www.osgi.org

OSGi is a trademark of the OSGi Alliance in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

All other marks are trademarks of their respective companies.