DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

DATE OF DISTRIBUTION: 31 August 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 to not 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 2 of the OSGi Service Platform Release 4 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 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
Abstract

This RFC describes the means for a bundle to hook into the resolver, bundle events and accessing bundles with a bundle context.
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.................................................................................................................................. 4

2 Application Domain..................................................................................................................... 5

3 Problem Description.................................................................................................................... 5

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

5 Technical Solution....................................................................................................................... 6
  5.1 Terminology......................................................................................................................... 7
  5.2 Synopsis.............................................................................................................................. 7
  5.3 Bundle Event Hook.............................................................................................................. 8
  5.4 Bundle Find Hook................................................................................................................ 8
  5.5 Resolver Hook..................................................................................................................... 9
    5.5.1 Resolution Process Begins........................................................................................ 9
    5.5.2 Resolution Process Ends........................................................................................... 10
    5.5.3 Hide Capabilities......................................................................................................... 10
    5.5.4 Effect of Singleton Bundles........................................................................................ 10
    5.5.5 Limit the Set of Resolvable Bundles.......................................................................... 11
    5.5.6 The Resolve Process................................................................................................. 11
  5.6 Bundle Symbolic Name and Version................................................................................... 12

6 Java Doc....................................................................................................................................... 13
  6.1 org.osgi.framework.hooks.bundle
    Interface EventHook.................................................................................................................. 13
      6.1.1 event........................................................................................................................... 13
  6.2 org.osgi.framework.hooks.bundle
    Interface FindHook................................................................................................................ 14
      6.2.1 find.............................................................................................................................. 14
  6.3 org.osgi.framework.hooks.resolver
    Interface ResolverHook.......................................................................................................... 15
      6.3.1 begin.......................................................................................................................... 16
      6.3.2 filterResolvable........................................................................................................ 16
      6.3.3 filterSingletonCollisions......................................................................................... 17
      6.3.4 filterMatches............................................................................................................. 17
6.4  org.osgi.framework.wiring

Interface BundleRevision................................................................. 18
  6.4.1 TYPE_FRAGMENT...................................................................... 19
  6.4.2 getSymbolicName.................................................................... 19
  6.4.3 getVersion................................................................................. 19
  6.4.4 getDeclaredCapabilities....................................................... 19
  6.4.5 getTypes................................................................................ 20

6.5  org.osgi.framework.wiring

Interface BundleWiring................................................................. 20
  6.5.1 FINDENTIRES_RECURSE............................................................ 21
  6.5.2 LISTRESOURCES_RECURSE...................................................... 22
  6.5.3 LISTRESOURCES_LOCAL........................................................... 22
  6.5.4 isCurrent................................................................................ 22
  6.5.5 isInUse..................................................................................... 22
  6.5.6 getProvidedCapabilities....................................................... 23
  6.5.7 getRequiredCapabilities...................................................... 23
  6.5.8 getBundleRevision.............................................................. 23
  6.5.9 getFragmentRevisions.......................................................... 24
  6.5.10 getClassLoader...................................................................... 24
  6.5.11 findEntries............................................................................. 24
  6.5.12 listResources...................................................................... 25

6.6  org.osgi.framework.wiring

Interface BundleWirings................................................................... 26
  6.6.1 getWirings.............................................................................. 26

6.7  org.osgi.framework.wiring

Interface Capability.......................................................................... 27
  6.7.1 PACKAGE_CAPABILITY.............................................................. 28
  6.7.2 BUNDLE_CAPABILITY............................................................. 28
  6.7.3 getNamespace........................................................................ 28
  6.7.4 getDirectives......................................................................... 29
  6.7.5 getAttributes......................................................................... 29
  6.7.6 getProviderRevision............................................................ 29

6.8  org.osgi.framework.wiring

Interface FrameworkWiring............................................................. 29
  6.8.1 refreshBundles....................................................................... 30
  6.8.2 resolveBundles...................................................................... 31
  6.8.3 getRemovalPendingBundles.................................................. 32
  6.8.4 getDependencyClosure.......................................................... 32

6.9  org.osgi.framework.wiring

Interface WiredCapability............................................................... 32
  6.9.1 getProviderWiring.............................................................. 33
  6.9.2 getRequirerWirings.............................................................. 33

7 Command Line API......................................................................... 34

8 JMX API.......................................................................................... 34

9 Initial Spec Chapter........................................................................ 34

10 Considered Alternatives............................................................. 35

11 Security Considerations............................................................... 35

12 Document Support........................................................................ 35
  12.1 References.............................................................................. 35
  12.2 Author’s Address.................................................................... 36
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 12.1.

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>08/02/10</td>
<td>At the F2F in Ottawa it was decided to remove the concept of composite from the core framework. This RFC now is used to specify a set of core framework hooks that can be used to build isolation models on top of the framework. Thomas Watson, IBM <a href="mailto:tjwatson@us.ibm.com">tjwatson@us.ibm.com</a></td>
</tr>
</tbody>
</table>
|          | 08/18/10 | - Bundle EventHook takes a collection of bundle context now
- Bundle EventHook must be called once and only once per BundleEvent
- The Bundle FindHook only affects the behavior of BundleContext.getBundles() not BundleContext.getBundle(long)
- Describe dynamic resolution process.
- Add begin/end to ResolverHook
- Provide details on the resolve process and the interaction with the resolver hooks
- Specify a failure when a resolver hook attempts to start a nested resolve process.
- Imported javadoc. |
1 Introduction

This RFC details how a bundle can hook into the various operations with in the module and lifecycle layers of the core Framework to influence the resolve operations and influence access to bundles through bundle events and the bundle context.

2 Application Domain

This design is targeted at bundles which need to observe and manipulate select resolution operations and access to bundles through bundle events and the bundle context. In general these will be highly specialized bundles written by systems programmers. The design is not intended to be used by so-called “normal” application bundles.

3 Problem Description

One of the key features of the module layer is to perform the resolution process which “wires” requirements (Import-Package, Require-Bundle etc.) to capabilities (Export-Package, Bundle-SymbolicName/Bundle-Version etc.). The module layer provides no means for a bundle (external to the framework) to observe or manipulate the resolution process as it occurs. Certain specialized bundles need to be able to alter the output results of the resolution process. Such purposes may include things like bundle grouping or scoping for isolation, etc.

The lifecycle layer provides no means for a bundle (external to the framework) to observe or manipulate access to bundles. Certain specialized bundles need to be able to alter the output results of the lifecycle layer with respect to accessing bundle objects with the bundle event delivery operations and accessing bundle objects with the bundle context. Such purposes may include things like bundle grouping or scoping for isolation, etc.
4 Requirements

RFP 100 contains a number of requirements used for this RFC. Additional requirements have been added or reworded to address some of the issues that have been observed since the original RFP.

- The solution MUST work with the current lifecycle layer.
- The solution MUST work with the current module layer.
- The solution MUST allow certain bundles to reduce the list of candidates (exported packages) available to resolve import package constraint from a given bundle.
- The solution MUST allow certain bundles to reduce the list of candidates (bundles) available to resolve require bundle constraints from a given bundle.
- The solution MUST allow certain bundles to reduce the list of candidates (host bundles) available to resolve fragment host constraints from a given fragment.
- The solution MUST allow certain bundles to reduce the list of generic capability (RFC 154) candidates available to resolve generic requirements from a given bundle.
- The solution MUST allow multiple bundles with identical symbolic-name and version to be installed without error.
- The solution MUST allow certain bundles to reduce the list of singleton bundles which influence the resolvability of a given singleton bundle.
- The solution MUST allow certain bundles to reduce the list of bundles a bundle event is delivered to.
- The solution MUST allow certain bundles to reduce the list of bundles returned by a find bundle operation (BundleContext.getBundles()).
- The solution MUST be secured when java permissions are in effect.
- Security MUST not be used to provide the means to the solution. The solution MUST work without a security manager.
5 Technical Solution

The OSGi framework has built-in support for bundle resolution primitives and bundle lifecycle primitives. The resolution primitives are quite complex as well as powerful. The resolver built into the framework provides the basis for which the complete module layer is built upon. However, the resolver operates on information that is not completely visible to the bundles and in most cases bundles cannot effect the results of the resolver. For example, it is not possible for a bundle outside of the framework reduce the set of exported packages available to resolve a particular import package from a certain bundle.

Additionally, it is also not possible to allow bundles to influence the access to other bundle objects from a certain bundle. For example, when a find bundle operation is performed using a bundle context (getBundles() method) it is not possible to reduce the collection of bundles returned. Also, it is not possible to hide bundle events from certain bundles.

Therefore, this framework hook specification provides a number of new mechanisms that closely interact with the built-in resolver and lifecycle layer of the framework. These interactions are not intended for use by application bundles. Modifying the behavior of the module and lifecycle layer requires developers to closely follow the semantics of the OSGi module and lifecycle model and this is often hard, requiring a significant amount of code.

5.1 Terminology

- **Client** – The bundle that finds bundles or receives events about bundles. Also, the bundle which declares requirements which need to be resolved.

- **Handler** – The bundle that registers a hook service uses this to:
  - view or modify the resolution process
  - view or modify find bundle operations
  - view or modify bundle event delivery

- **Capability** – Some feature that is declared with meta-data. For example, Export-Package.

- **Provider** – A bundle that provides a capability

- **Consumer** – A client bundle that requires a capability

- **Resolver** – The internal framework machinery that wires (resolves) constraints declared by a consumer to capabilities declared by a provider.

- **Bundle Event Hook** – A bundle event hook intercepts bundle events before they are delivered to the client. The hook can select to remove events for specific bundles, which effectively allows the hook to hide bundle events from a bundle.

- **Bundle Find Hook** – A bundle find hook intercepts the getBundle(s) call just before it returns the results to the client. The results can be influenced by removing bundle entries. The bundle find hook can be used to hide specific bundles for specific bundles.
• Resolver Hook – A resolver hook intercepts the resolve process in the following ways:
  ◦ Remove specific capabilities available to resolve a specific client bundle
  ◦ Remove specific singleton bundles from influencing the resolvability of another specific singleton bundle.
  ◦ Remove specific bundles from a list of resolvable bundles to prevent specific bundles from resolving during a single resolution process.

5.2 Synopsis
A bundle that needs to hide specific capabilities available to wire to from other bundles can register a resolver hook by registering a Resolver Hook service with the framework. If a resolve operation is performed it will pass this resolve hook information about the resolve process as it is occurring. The resolver hook can then inspect the arguments and optionally remove candidates available to a consumer to influence the resolver solution.

A bundle that needs to hide bundle objects from other bundles can register a bundle find or event hook by registering a Bundle Find Hook service or a Bundle Event Hook service with the framework. If a bundle event is generated, it will pass this event to the bundle event hook. The event hook method can then inspect the arguments and optionally remove bundles from the event delivery list.

When a bundle uses the Bundle Context getBundle(long) or getBundles method, the Bundle Find Hook is notified with a list of discovered bundles. The hook can then remove any bundles it wants to hide from the target bundle.

5.3 Bundle Event Hook
To intercept events being delivered to bundles, a handler must register a bundle EventHook object as a service with the framework. The framework must then send all bundle events to all registered hooks. The calling order of the hooks is defined by the reversed compareTo ordering of their Service Reference objects. That is, the service with the highest ranking number is called first. Bundle Event hooks are called after the event is generated but before they are delivered to any of the registered bundle listeners. Before the return, the handler can remove bundles from the given list. This allows a bundle event hook to hide bundle events for specific bundles.

An event hook receives all events, INSTALLED, RESOLVED, STARTING, STARTED, STOPPING, STOPPED, UNRESOLVED, UPDATED, and UNINSTALLED, if and only if there one or more bundle listeners registered with the framework.

The bundle EventHook class has a single method:

• event(BundleEvent, Collection<BundleContext>) – A bundle event has been generated. The implementer of this method can optionally shrink the given collection of target bundles.

One of the parameters of the event method is a collection of target bundle contexts. The handler can shrink this list by removing bundle contexts. The collection and its iterator must therefore implement the remove method.

Removing a bundle context from the list of target bundle contexts will effectively hide the bundle event from the target bundle context and therefore any bundle listeners registered with that bundle context. The target bundle context can still get the bundle for the event with the BundleContext.getBundle(s) methods, though the Bundle Find Hook can be used to block this access to the bundle.

The event method must be called one and only one time for each bundle event generated, this included bundle events which are generated when there are no bundle listeners registered. The event method must be called on
the same thread that is performing the action which generated the specified event. The specified collection includes bundle contexts with synchronous and asynchronous bundle listeners registered with them.

Implementations of the Bundle Event Hook must ensure that target bundles continue to see a consistent set of bundle events. Bundle events can be used in a stat machine. Such state machines can get confused if some events are missed. For example, if a Bundle Tracker sees a STARTED event but is hidden from the corresponding STOPPED event then the tracker will still think the bundle is active. A simple solution is to stop the target bundle when the filter is put in place. However, when the bundle that implements the bundle event hook is stopped, it will of course no longer be able to filter events and the target bundle might see bundle events for bundles it was not aware of. As a best practice a bundle event hook should not hide an event from the bundle which the event is for.

5.4 Bundle Find Hook

The Bundle Find Hook is called when a target bundle searches the framework with the getBundles() methods on BundleContext. A registered Bundle Find Hook service gets a chance to inspect the returned set of bundles and can optionally shrink the set of returned bundles. The order in which the bundle find hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first.

The bundle FindHook class has a single method:

- find(BundleContext, Collection<Bundle>) - The callback when a bundle calls the getBundle(long) or getBundles() methods on BundleContext. As parameters, it gets the bundle context of the calling bundle, and a set of bundles that will be returned. This list can be shortened by removing bundles from the given list.

The purpose of the Bundle Find Hook, is to limit the visibility of bundles to selected target bundles. For this reason the hook implementation can remove selected bundles from the result collection. The collection and its iterator must therefor implement the remove method. As a best practice a bundle find hook should not hide a bundle from itself.

5.5 Resolver Hook

The Resolver Hook is called during a resolution process. A registered Resolver Hook gets a chance to influence the outcome of a resolution process in the following ways.

- Limit the visibility of a capability to selected target bundles.
- Limit the effect a singleton bundle has on the resolvability of selected target bundles.
- Limit the set of bundles that will be available for resolution to a set of selected target bundles.

There are types of resolve processes that can be initiated.

1. A static bundle resolution operation. This resolve process is necessary any time one or more bundles transitions from the INSTALLED state to the RESOLVED state. During this resolve process the framework attempts to resolve static requirements specified by the bundles

2. A dynamic import resolution operation. This resolve process is necessary any time a request is made to wire a dynamic import for a bundle.
The framework must, at most, have one resolve process running at any given point in time. A resolver hook may influence the outcome of a resolve process by removing entries from shrinkable collections that are passed to the hook during a resolve process. A shrinkable collection is a collection that supports all remove operations. Any other attempts to modify a shrinkable collection will result in an UnsupportedOperationException being thrown.

5.5.1 Resolution Process Begins

A resolver hook is informed about a resolution process beginning with the following method:

- **begin()** - This method is called once at the beginning of the resolve process before any other methods are called on the hook.

The order in which the resolver hook begin methods are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first.

TJW – We may want to consider adding parameters to begin which indicate why the resolve process is beginning. For example, is this a static resolution process or a dynamic package import process (perhaps an (int type) param). This is also the method which we could add the initial set of bundles being requested to resolve (bug 1703)

5.5.2 Resolution Process Ends

A resolver hook is informed about a resolution process ending with the following method:

- **end()** - This method is called once at the end of the resolve process. After the end method is called the resolve process has ended. No methods will be called on the hook except after the begin() method is called again to indicate a new resolve process is beginning.

The order in which the resolver hook end methods are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first.

5.5.3 Hide Capabilities

A Resolver Hook can hide capabilities using the following method:

- **filterMatches(BundleRevision requirer, Collection<Capability> candidates)** – A bundle (requirer) has a constraint which can be satisfied (matched) with the supplied collection of candidates (capabilities). The implementor of this method can optionally shrink the list of candidate capabilities.

One of the parameters of the filterCandidates method is a list of capabilities. The handler can shrink this collection by removing capabilities. The collection and its iterator must therefore implement the remove method. Removing a capability from the list of candidates will effectively hide the capability from the target bundle. This will prevent the target bundle from getting wired to the capability.

The order in which the resolver hook filterMatches methods are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first.

5.5.4 Effect of Singleton Bundles

A Resolver Hook can change the effect of singleton bundles using the following method:

- **filterSingletonCollisions(Capability singleton, Collection<Capability> collisionCandidates)** – An osgi.bundle singleton capability has the same symbolic-name as the given collection of osgi.bundle capability
candidates. The implementor of this method can optionally shrink the list of collision candidate capabilities.

One of the parameters of the filterSingletonCollisions method is a set of capabilities. The handler can shrink this collection by removing capabilities. The collection and its iterator must therefore implement the remove method. Removing a capability from the list of collision candidates will effectively hide the collision candidate from the target singleton bundle. This will allow the target singleton bundle to resolve regardless of the resolution state of the collision candidate.

The framework may call this method multiple times for the same singleton capability. For example, as a first pass a framework may want to determine if a singleton bundle is resolvable first based on the presence of other already resolved singleton capabilities. Later the framework may call this method again to determine which singleton capabilities from unresolved bundles to allow to resolve.

The order in which the resolver hook filterSingletonCollisions methods are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first.

### 5.5.5 Limit the Set of Resolvable Bundles

A Resolver hook can limit the set of bundles that will be allowed to resolve for a single resolve operation with the following method:

- `filterResolvable(Collection<BundleRevision> candidates)` - A resolve operation has been started. The given collection of candidate bundles are available to resolve (i.e. they are currently unresolved). The implementor of this method can optionally shrink the collection of candidate bundles.

The candidates parameters of the filterResolvable method is a collection of bundle revisions. The handler can shrink this collection by removing candidate bundle revisions. The collection and its iterator must therefore implement the remove method. Removing a candidate bundle from the collection of candidates will effectively prevent the bundle from resolving for the current resolve operation.

The order in which the resolver hook filterResolvable methods are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first.

For a dynamic import resolution process it is acceptable for the framework to pass an empty collection of resolvable candidates. This indicates that the framework will not cause any bundles to transition from INSTALLED to RESOLVED during a dynamic import package resolution.

### 5.5.6 The Resolve Process

The following steps outline the way a framework uses the resolver hooks during a resolve process.

1. Collect a snapshot of registered resolver hooks that will be called during the current resolve process. Any hooks registered after the snapshot is taken must not be called during the current resolve process. A resolver hook contained in the snapshot may become unregistered during the resolve process. The framework should handle this and stop calling the unregistered hook for the remainder of the resolve process. Each registered resolver hook service in the snapshot will be obtained by the framework (by calling BundleContext.getService method using the system bundle context).

2. For each registered hook call the begin() method to inform the hooks about a resolve process beginning.

3. Determine the collection of unresolved bundle revisions that may be considered for resolution during the current resolution process and place each of the bundle revisions in a shrinkable collection `R`.

Copyright © IBM Corporation 2010 All Rights Reserved
a) For each registered hook call the filterResolveable method with the shrinkable collection \( R \).

4. The shrinkable collection \( R \) now contains all the unresolved bundle revisions that may end up as resolved at the end of the current resolve process. Any other bundle revisions that got removed from the shrinkable collection \( R \) must not end up as resolved at the end of the current resolve process.

5. For each bundle revision \( B \) left in the shrinkable collection \( R \) that represents a singleton bundle do the following:

   a) Determine the collection of available capabilities that have a name space of osgi.bundle, are singletons, and have the same symbolic name as the singleton bundle revision \( B \) and place each of the matching capabilities into a shrinkable collection \( S \).

   b) Remove the osgi.bundle capability provided by the bundle revision \( B \) from the shrinkable collection \( S \). A singleton bundle cannot collide with itself.

   c) For each registered hook call the filterSingletonCollisions method with the osgi.bundle capability provided by bundle revision \( B \) and the shrinkable collection \( S \).

   d) The shrinkable collection \( S \) now contains all the singleton osgi.bundle capabilities that can influence the ability of bundle revision \( B \) to resolve.

6. During a resolve process a framework is free to attempt to resolve any or all bundles contained in the shrinkable collection \( R \). For each bundle revision \( B \) left in the shrinkable collection \( R \) which the framework attempts to resolve the following steps must be followed:

   a) For each requirement \( T \) specified by bundle revision \( B \) determine the collection of capabilities that satisfy (or match) the requirement and place each matching capability into a shrinkable collection \( C \).

   b) For each registered hook call the filterMatches method with the bundle revision \( B \) and the shrinkable collection \( C \).

   c) The shrinkable collection \( C \) now contains all the capabilities that may be used to satisfy the requirement \( T \). Any other capabilities that got removed from the shrinkable collection \( C \) must not be used to satisfy requirement \( T \).

7. For each registered hook call the end method to inform the hook about a resolve process ending.

8. For each hook obtained by the framework in the first step; unget the service instance (by calling BundleContext.ungetService using the system bundle context).

In cases where the a shrinkable collection becomes empty the framework is required to call the remaining hooks.

The above steps are meant to illustrate how the resolve hooks are used by the framework. They are not normative. The nature of the resolve process and the resolve algorithm can require some backtracking by the framework implementation. Because of this it is acceptable for the framework to call methods on the ResolverHook multiple times with similar or identical parameters during a single resolve process. This is true for all methods except the begin and end methods. The begin and end methods must be called once and only once for each resolve process.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another resolve process (e.g. by calling Bundle.start() or FrameworkWiring.resolveBundles). The framework must detect this and throw an IllegalStateException from the resolve process. In cases where a BundleException
can be thrown (e.g. Bundle.start()) the IllegalStateException must be the cause of the BundleException and the BundleException must be of type BundleException.RESOLVE_ERROR. In cases where an exception cannot be propagated to a caller (e.g. dynamic import resolution) a FrameworkEvent of type ERROR must be published.

### 5.6 Bundle Symbolic Name and Version

In R4.0 the concept of a bundle symbolic name and version were formalized. The unique content of a bundle can be identified by its unique bundle symbolic name and version. The R4.0 specification also caused an installation exception if the same bundle (with identical bundle symbolic name and version) was installed multiple times. This restriction is being lifted in this release to allow the same bundle to be installed multiple times (using unique location strings).

TJW – There have been concerns about loosening this install time check. We may need to consider adding a new method that takes options at install or update.

### 6 Java Doc

<table>
<thead>
<tr>
<th>Package Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.osgi.framework.hooks.bundle</td>
<td>14</td>
</tr>
<tr>
<td>org.osgi.framework.hooks.resolver</td>
<td>17</td>
</tr>
<tr>
<td>org.osgi.framework.wiring</td>
<td>21</td>
</tr>
</tbody>
</table>

Framework Bundle Hooks Package Version 1.0.
Framework Wiring Hooks Package Version 1.0.
Framework Wiring Package Version 1.0.
Package org.osgi.framework.hooks.bundle

Framework Bundle 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>15</td>
</tr>
<tr>
<td>FindHook</td>
<td>16</td>
</tr>
</tbody>
</table>

Package org.osgi.framework.hooks.bundle Description

Framework Bundle 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.

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.framework.hooks.bundle; version="[1.0,1.1)"
```
public interface EventHook

OSGi Framework Bundle Event Hook Service.

Bundles registering this service will be called during framework lifecycle (install, start, stop, update, and uninstall bundle) operations.

Version: $Id: d9e163e0ed32cf77eadd254a24c910a8882a517 $
ThreadSafe

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>

### Method Detail

**event**

```java
void event (org.osgi.framework.BundleEvent event, Collection<org.osgi.framework.BundleContext> contexts)
```

Bundle event hook method. This method is called prior to bundle event delivery when a bundle is installed, resolved, started, stopped, unresolved, or uninstalled. This method can filter the bundles which receive the event.

This method must be called by the framework one and only one time for each bundle event generated, this included bundle events which are generated when there are no bundle listeners registered. This method must be called on the same thread that is performing the action which generated the specified event. The specified collection includes bundle contexts with synchronous and asynchronous bundle listeners registered with them.

**Parameters:**
- `event` - The bundle 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 UnsupportedOperation exception. The collection is not synchronized.
public interface FindHook

OSGi Framework Bundle Context Hook Service.

Bundles registering this service will be called during framework bundle find (get bundles) operations.

Version:
$Id: ade0301a4edaa5d4c756fc785e136756cd8647b5 $

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
</table>

Method Detail

find

void find (org.osgi.framework.BundleContext context, Collection<org.osgi.framework.Bundle> bundles)

Find hook method. This method is called during the bundle find operation (for example, getBundle and org.osgi.framework.BundleContext.getBundles() methods). This method can filter the result of the find operation.

Parameters:
- context - The bundle context of the bundle performing the find operation.
- bundles - A collection of Bundles to be returned as a result of the find operation. The implementation of this method may remove bundles from the collection to prevent the bundles 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.
Package org.osgi.framework.hooks.resolver

Framework Wiring Hooks Package Version 1.0.

See:  
  Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResolverHook</td>
<td>18</td>
</tr>
</tbody>
</table>

OSGi Framework Resolver Hook Service.

Package org.osgi.framework.hooks.resolver Description

Framework Wiring 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.

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.framework.hooks.wiring; version="[1.0,1.1)"
```
public interface ResolverHook

OSGi Framework Resolver Hook Service.

Services registered with this service interface will be called by the framework during a resolve process. The framework must, at most, have one resolve process running at any given point in time. A resolver hook may influence the outcome of a resolve process by removing entries from shrinkable collections that are passed to the hook during a resolve process. A shrinkable collection is a `Collection` that supports all remove operations. Any other attempts to modify a shrinkable collection will result in an `UnsupportedOperationException` being thrown.

The following steps outline the way a framework uses the resolver hooks during a resolve process.

1. Collect a snapshot of registered resolver hooks that will be called during the current resolve process. Any hooks registered after the snapshot is taken must not be called during the current resolve process. A resolver hook contained in the snapshot may become unregistered during the resolve process. The framework should handle this and stop calling the unregistered hook for the remainder of the resolve process.
2. For each registered hook call the `begin()` method to inform the hooks about a resolve process beginning.
3. Determine the collection of unresolved bundle revisions that may be considered for resolution during the current resolution process and place each of the bundle revisions in a shrinkable collection `R`.
   a. For each registered hook call the `filterResolvable(Collection)` method with the shrinkable collection `R`.
4. The shrinkable collection `R` now contains all the unresolved bundle revisions that may end up as resolved at the end of the current resolve process. Any other bundle revisions that got removed from the shrinkable collection `R` must not end up as resolved at the end of the current resolve process.
5. For each bundle revision `B` left in the shrinkable collection `R` that represents a singleton bundle do the following:
   a. Determine the collection of available capabilities that have a name space of `osgi.bundle`, are singletons, and have the same symbolic name as the singleton bundle revision `B` and place each of the matching capabilities into a shrinkable collection `S`.
   b. Remove the `osgi.bundle` capability provided by bundle revision `B` from shrinkable collection `S`. A singleton bundle cannot collide with itself.
   c. For each registered hook call the `filterSingletonCollisions(Capability, Collection)` with the `osgi.bundle` capability provided by bundle revision `B` and the shrinkable collection `S`.
   d. The shrinkable collection `S` now contains all singleton `osgi.bundle` capabilities that can influence the ability of bundle revision `B` to resolve.
6. During a resolve process a framework is free to attempt to resolve any or all bundles contained in shrinkable collection `R`. For each bundle revision `B` left in the shrinkable collection `R` which the framework attempts to resolve the following steps must be followed:
   a. For each requirement `T` specified by bundle revision `B` determine the collection of capabilities that satisfy (or match) the requirement and place each matching capability into a shrinkable collection `C`.
   b. For each registered hook call the `filterMatches(BundleRevision, Collection)` with the bundle revision `B` and the shrinkable collection `C`.
   c. The shrinkable collection `C` now contains all the capabilities that may be used to satisfy the requirement `T`. Any other capabilities that got removed from the shrinkable collection `C` must not be used to satisfy requirement `T`.
7. For each registered hook call the `end()` method to inform the hooks about a resolve process ending.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.

In all cases, the order in which the resolver hooks are called is the reverse compareTo ordering of their Service References. That is, the service with the highest ranking number must be called first. In cases where a shrinkable collection becomes empty the framework is required to call the remaining registered hooks.

Resolver hooks are low level. Implementations of the resolver hook must be careful not to create an unresolvable state which is very hard for a developer or a provisioner to diagnose. Resolver hooks also must not be allowed to start another synchronous resolve process (e.g. by calling `org.osgi.framework.Bundle.start()` or `FrameworkWiring.resolveBundles(Collection)`). The framework must detect this and throw an `IllegalStateException`.
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>begin()</code></td>
<td>19</td>
</tr>
<tr>
<td>This method is called once at the beginning of the resolve process before any other methods are called on this hook.</td>
<td></td>
</tr>
<tr>
<td>void <code>end()</code></td>
<td>20</td>
</tr>
<tr>
<td>This method is called once at the end of the resolve process.</td>
<td></td>
</tr>
<tr>
<td>void <code>filterMatches(BundleRevision requirer, Collection&lt;Capability&gt; candidates)</code></td>
<td>20</td>
</tr>
<tr>
<td>Filter matches hook method.</td>
<td></td>
</tr>
<tr>
<td>void <code>filterResolvable(Collection&lt;BundleRevision&gt; candidates)</code></td>
<td>19</td>
</tr>
<tr>
<td>Filter resolvable candidates hook method.</td>
<td></td>
</tr>
<tr>
<td>void <code>filterSingletonCollisions(Capability singleton, Collection&lt;Capability&gt; collisionCandidates)</code></td>
<td>19</td>
</tr>
<tr>
<td>Filter singleton collisions hook method.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Detail

**begin**

void `begin()`

This method is called once at the beginning of the resolve process before any other methods are called on this hook.

**filterResolvable**

void `filterResolvable(Collection<BundleRevision> candidates)`

Filter resolvable candidates hook method. This method may be called multiple times during a single resolve process. This method can filter the collection of candidates by removing potential candidates. Removing a candidate will prevent the candidate from resolving during the current resolve process.

**Parameters:**

- `candidates` - the collection of resolvable candidates available during a resolve process.

**filterSingletonCollisions**

void `filterSingletonCollisions(Capability singleton, Collection<Capability> collisionCandidates)`

Filter singleton collisions hook method. This method is called during the resolve process for the specified singleton. The specified singleton represents a singleton capability and the specified collection represent a collection of singleton capabilities which are considered collision candidates. The singleton capability and the collection of collision candidates must all use the same name space.

Currently only capabilities with the name space of `osgi.bundle` can be singletons. In that case all the collision candidates have the name space of `osgi.bundle`, are singletons, and have the same symbolic name as the specified singleton capability.

In the future, capabilities in other name spaces may support the singleton concept. Hook implementations should be prepared to receive calls to this method for capabilities in name spaces other than `osgi.bundle`.
Interface ResolverHook

This method can filter the list of collision candidates by removing potential collisions. Removing a collision candidate will allow the specified singleton to resolve regardless of the resolution state of the removed collision candidate.

**Parameters:**
- `singleton` - the singleton involved in a resolve process
- `collisionCandidates` - a collection of singleton collision candidates

---

**filterMatches**

```java
void filterMatches(BundleRevision requirer,
                   Collection<Capability> candidates)
```

Filter matches hook method. This method is called during the resolve process for the specified requirer. The collection of candidates match a requirement for the requirer. This method can filter the collection of matching candidates by removing candidates from the collection. Removing a candidate will prevent the resolve process from choosing the removed candidate to satisfy a requirement for the requirer.

All of the candidates will have the same name space and will match a requirement of the requirer.

**Parameters:**
- `requirer` - the bundle revision which contains a requirement
- `candidates` - a collection of candidates that match a requirement of the requirer

---

**end**

```java
void end()
```

This method is called once at the end of the resolve process. After the end method is called the resolve process has ended. No methods will be called on this hook except after the `begin` method is called again to indicate a new resolve process is beginning.
Package org.osgi.framework.wiring

Framework Wiring Package Version 1.0.

See:

Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleRevision</td>
<td>22</td>
</tr>
<tr>
<td>BundleWiring</td>
<td>24</td>
</tr>
<tr>
<td>BundleWirings</td>
<td>29</td>
</tr>
<tr>
<td>Capability</td>
<td>30</td>
</tr>
<tr>
<td>FrameworkWiring</td>
<td>33</td>
</tr>
<tr>
<td>WiredCapability</td>
<td>36</td>
</tr>
</tbody>
</table>

### Package org.osgi.framework.wiring Description

Framework Wiring 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.wiring; version="[1.0,2.0)"
```
Interface BundleRevision

org.osgi.framework.wiring

All Superinterfaces:
org.osgi.framework.BundleReference

public interface BundleRevision
extends org.osgi.framework.BundleReference

Bundle Revision. Since a bundle update can change the entries in a bundle, different bundle wirings for the same bundle can be associated with different bundle revisions.

The current bundle revision for a bundle can be obtained by calling bundle.adapt(BundleRevision.class).

Version:
$Id: f5d8487c04841817478513898635620ca788e84f $
ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int TYPE_FRAGMENT</td>
<td>22</td>
</tr>
<tr>
<td>Bundle revision type indicating the bundle revision is a fragment.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;Capability&gt; getDeclaredCapabilities(String namespace)</td>
<td>23</td>
</tr>
<tr>
<td>Returns the capabilities declared by this bundle revision.</td>
<td></td>
</tr>
<tr>
<td>String getSymbolicName()</td>
<td>22</td>
</tr>
<tr>
<td>Returns the symbolic name for this bundle revision.</td>
<td></td>
</tr>
<tr>
<td>int getTypes()</td>
<td>23</td>
</tr>
<tr>
<td>Returns the special types of this bundle revision.</td>
<td></td>
</tr>
<tr>
<td>org.osgi.framework.Version getVersion()</td>
<td>23</td>
</tr>
<tr>
<td>Returns the version for this bundle revision.</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.BundleReference

getBundle

Field Detail

TYPE_FRAGMENT

public static final int TYPE_FRAGMENT = 1

Bundle revision type indicating the bundle revision is a fragment.

See Also:
getTypes()

Method Detail

getSymbolicName

String getSymbolicName()
Interface BundleRevision

Returns the symbolic name for this bundle revision.

*Returns:* The symbolic name for this bundle revision.

*See Also:* org.osgi.framework.Bundle.getSymbolicName()

getVersion

org.osgi.framework.Version getVersion()

Returns the version for this bundle revision.

*Returns:* The version for this bundle revision, or org.osgi.framework.Version.emptyVersion if this bundle revision has no version information.

*See Also:* org.osgi.framework.Bundle.getVersion()

getDeclaredCapabilities

List<Capability> getDeclaredCapabilities(String namespace)

Returns the capabilities declared by this bundle revision.

*Parameters:* namespace - The name space of the declared capabilities to return or null to return the provided capabilities from all name spaces.

*Returns:* A list containing a snapshot of the declared Capabilities, or an empty list if this bundle revision declares no capabilities in the specified name space. The list contains the provided capabilities in the order they are specified in the manifest.

getTypes

int getTypes()

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

- TYPE_FRAGMENT

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

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

*Returns:* The special types of this bundle revision. The type values are ORed together.
public interface BundleWiring
extends org.osgi.framework.BundleReference

A wiring for a bundle. Each time a bundle is resolved, a new bundle wiring for the bundle is created. A bundle wiring 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 bundles with non-current, in use bundle wirings are considered removal pending. A bundle wiring is in use if it is the current wiring or if some other in use 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 has a 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 bundle wirings for a bundle can be obtained by calling bundle.adapt(BundleWirings.class).getWirings(). For non-fragment bundles, the first item in the returned list is the current bundle wiring.

The current bundle wiring for a non-fragment bundle can be obtained by calling bundle.adapt(BundleWiring.class). A fragment bundle does not itself have bundle wirings. So calling bundle.adapt(BundleWiring.class) on a fragment must return null.

Version: $Id: 23b89b31d22b6a907db410fcd822f7b55cd7568$
ThreadSafe
Interface BundleWiring

boolean isCurrent()
Returns true if this bundle wiring is the current bundle wiring.

boolean isInUse()
Returns true if this bundle wiring is in use.

List<String> listResources(String path, String filePattern, int options)
Returns the names of resources visible to this bundle wiring's class loader.

Methods inherited from interface org.osgi.framework.BundleReference
getBundle

Field Detail

FINDENTRIES_RECURSE

public static final int FINDENTRIES_RECURSE = 1

The find entries operation must recurse into subdirectories.

This bit may be set when calling findEntries(String, String, int) to specify the result must include
the matching entries from the specified path and its subdirectories. If this bit is not set, then the result must
only include matching entries from the specified path.

See Also:
findEntries(String, String, int)

LISTRESOURCES_RECURSE

public static final int LISTRESOURCES_RECURSE = 1

The list resource names operation must recurse into subdirectories.

This bit may be set when calling listResources(String, String, int) to specify the result must include
the names of matching resources from the specified path and its subdirectories. If this bit is not set, then the result must
only include names of matching resources from the specified path.

See Also:
listResources(String, String, int)

LISTRESOURCES_LOCAL

public static final int LISTRESOURCES_LOCAL = 2

The list resource names operation must limit the result to the names of matching resources contained in
this bundle wiring's bundle revision and its attached fragment revisions.

This bit may be set when calling listResources(String, String, int) to specify the result must only include
the names of matching resources contained in this bundle wiring's bundle revision and its attached
fragment revisions. If this bit is not set, then the result must include the names of matching resources
reachable from this bundle wiring's class loader which may include the names of matching resources
contained in imported packages and required bundles.

See Also:
listResources(String, String, int)
Method Detail

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 bundles with non-current, in use bundle wirings 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 in use 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.

getProvidedCapabilities

List<WiredCapability> getProvidedCapabilities(String capabilityNamespace)

Returns the capabilities provided by this bundle wiring.

Parameters:
  capabilityNamespace - The name space of the provided capabilities to return or null to return the provided capabilities from all name spaces.

Returns: A list containing a snapshot of the WiredCapabilitys, or an empty list if this bundle wiring provides no capabilities in the specified name space. If this bundle wiring is not in use, null will be returned. The list contains the provided capabilities in the order they are specified in the manifest.

getRequiredCapabilities

List<WiredCapability> getRequiredCapabilities(String capabilityNamespace)

Returns the required capabilities used by this bundle wiring.

The result of this method can change if this bundle wiring requires additional capabilities.

Parameters:
  capabilityNamespace - The name space of the required capabilities to return or null to return the required capabilities from all name spaces.

Returns: A list containing a snapshot of the WiredCapabilitys used by this bundle wiring, or an empty list if this bundle wiring requires no capabilities in the specified name space. If this bundle wiring is not in use, null will be returned. The list contains the required capabilities in the order they are specified in the manifest.
getBundleRevision

**BundleRevision getBundleRevision()**

Returns the bundle revision for the bundle in this bundle wiring. Since a bundle update can change the entries in a bundle, different bundle wirings for the same bundle can have different bundle revisions.

The bundle object referenced by the returned BundleRevision may return different information than the returned BundleRevision since the returned BundleRevision may refer to an older revision of the bundle.

**Returns:**
The bundle revision for this bundle wiring.

getFragmentRevisions

**List<BundleRevision> getFragmentRevisions()**

Returns the bundle revisions for all attached fragments of this bundle wiring. Since a bundle update can change the entries in a fragment, different bundle wirings for the same bundle can have different bundle revisions.

The bundle revisions in the list are ordered in fragment attachment order such that the first revision in the list is the first attached fragment and the last revision in the list is the last attached fragment.

**Returns:**
A list containing a snapshot of the BundleRevisions 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.

getClassLoader

**ClassLoader getClassLoader()**

Returns the class loader for this bundle wiring. Since a bundle refresh creates a new bundle wiring for a bundle, different bundle wirings for the same bundle will have different class loaders.

**Returns:**
The class loader for this bundle wiring. If this bundle wiring is not in use, null will be returned.

**Throws:**
SecurityException - If the caller does not have the appropriate RuntimePermission("getClassLoader"), and the Java Runtime Environment supports permissions.

findEntries

**List<URL> findEntries(String path, String filePattern, int options)**

Returns entries in this bundle wiring's bundle revision and its attached fragment revisions. This bundle wiring's class loader is not used to search for entries. Only the contents of this bundle wiring's bundle revision and its attached fragment revisions are searched for the specified entries.

This method takes into account that the "contents" of this bundle wiring can have attached fragments. This "bundle space" is not a namespace with unique members; the same entry name can be present multiple times. This method therefore returns a list 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.

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 wiring and may begin with "/". A path value of "/" indicates the root of this bundle wiring.
- `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.
- `options` - The options for listing resource names. See `FINDENTRIES_RECURSE`. The method must ignore unrecognized options.

**Returns:**
An unmodifiable list of URL objects for each matching entry, or an empty list if no matching entry could not be found or if the caller does not have the appropriate `AdminPermission[bundle,RESOURCE]` and the Java Runtime Environment supports permissions. The list is ordered such that entries from the bundle revision are returned first followed by the entries from attached fragment revisions in attachment order. If this bundle wiring is not in use, null will be returned.

**See Also:**
org.osgi.framework.Bundle.findEntries(String, String, boolean)

---

**listResources**

```java
List<String> listResources(String path,
  String filePattern,
  int options)
```

Returns the names of resources visible to this bundle wiring's class loader. The returned names can be used to access the resources via this bundle wiring's class loader.

**Parameters:**
- `path` - The path name in which to look. The path is always relative to the root of this bundle wiring's class loader and may begin with "/". A path value of "/" indicates the root of this bundle wiring's class loader.
- `filePattern` - The file name pattern for selecting resource names in the specified path. The pattern is only matched against the last element of the resource path. If the resource 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.
- `options` - The options for listing resource names. See `LISTRESOURCES_LOCAL` and `LISTRESOURCES_RECURSE`. The method must ignore unrecognized options.

**Returns:**
An unmodifiable list of resource names for each matching resource, or an empty list if no matching resource could not be found or if the caller does not have the appropriate `AdminPermission[bundle,RESOURCE]` and the Java Runtime Environment supports permissions. The list is ordered such that resource names from this bundle are returned in the order they are visible in this bundle wiring's class loader. If this bundle wiring is not in use, null will be returned.
Interface BundleWirings

```
public interface BundleWirings
extends org.osgi.framework.BundleReference
```

The in use bundle wirings for a bundle. Each time a bundle is resolved, a new bundle wiring of the bundle is created. A bundle wiring consists of a bundle and it attached fragments and represents the dependencies with other bundle wirings.

The in use bundle wirings for a bundle can be obtained by calling `bundle.adapt(BundleWirings.class).getWirings()`.

Version: $Id: dc8656f9ab4562e4eda1c461c1b3414e5743515e $

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>List&lt;BundleWiring&gt; getWirings()</code></td>
<td>Return the in use wirings for the referenced bundle.</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.BundleReference

getBundle

Method Detail

`getWirings`

```
List<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 in use bundle wirings. The list is ordered in reverse chronological order such that the first bundle wiring is the current bundle wiring and last wiring is the oldest in use bundle wiring.

If the referenced bundle is a fragment bundle, then the result is a list of in use bundle wirings to which the referenced fragment bundle is attached. The ordering of the list is unspecified. If the fragment bundle is not attached to any bundle wiring, then the returned list will be empty.

The list must only contain in use bundle wirings. Generally the list will have at least one bundle wiring for the bundle: the current bundle wiring. However, for an uninstalled bundle with no in use bundle wirings or a newly installed bundle which has not been resolved, the list will be empty.

**Returns:**

A list containing a snapshot of the `BundleWirings` for the referenced bundle.
public interface Capability

A capability that has been declared from a bundle revision.

The framework defines capabilities for packages and bundles.

Version: $Id: 89b33162fc8df94495391fe8b3fc2d9d932a59b5 $
ThreadSafe

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String BUNDLE_CAPABILITY</td>
<td>Capability name space for bundle capabilities.</td>
</tr>
<tr>
<td>String PACKAGE_CAPABILITY</td>
<td>Capability name space for package capabilities.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map&lt;String, Object&gt; getAttributes()</td>
<td>Returns the attributes of this capability.</td>
</tr>
<tr>
<td>Map&lt;String, String&gt; getDirectives()</td>
<td>Returns the directives of this capability.</td>
</tr>
<tr>
<td>String getNamespace()</td>
<td>Returns the name space of this capability.</td>
</tr>
<tr>
<td>BundleRevision getProviderRevision()</td>
<td>Returns the bundle revision declaring this capability.</td>
</tr>
</tbody>
</table>

Field Detail

PACKAGE_CAPABILITY

public static final String PACKAGE_CAPABILITY = "osgi.package"

Capability name space for package capabilities. The name of the package is stored in the capability attribute of the same name as this name space. The other directives and attributes of the package, from the Export-Package manifest header, can be found in the capability's directives and attributes. The version capability attribute must contain the org.osgi.framework.Version of the package if one is specified.

The package capabilities provided by the system bundle, that is the bundle with id zero, must include the package specified by the org.osgi.framework.Constants.FRAMEWORK_SYSTEMPACKAGES and org.osgi.framework.Constants.FRAMEWORK_SYSTEMPACKAGES_EXTRA framework properties as well as any other package exported by the framework implementation.

A bundle revision declares zero or more package capabilities (this is, exported packages).

A bundle wiring provides zero or more resolved package capabilities (that is, exported packages) and requires zero or more resolved package capabilities (that is, imported packages). The number of package
capabilities required by a bundle wiring may change as the bundle wiring may dynamically import additional packages.

**BUNDLE_CAPABILITY**

```java
public static final String BUNDLE_CAPABILITY = "osgi.bundle"
```

Capability name space for bundle capabilities. The bundle symbolic name of the bundle is stored in the capability attribute of the same name as this name space. The other directives and attributes of the bundle, from the Bundle-SymbolicName manifest header, can be found in the capability's directives and attributes. The bundle-version capability attribute must contain the org.osgi.framework.Version of the bundle, from the Bundle-Version manifest header.

A bundle wiring provides exactly one† bundle capability (that is, the bundle can be required by another bundle) and requires zero or more bundle capabilities (that is, requires other bundles).

† A bundle with no bundle symbolic name (that is, a bundle with Bundle-ManifestVersion< 2) must not provide a bundle capability.

**Method Detail**

**getNamespace**

```java
String getNamespace()
```

Returns the name space of this capability.

**Returns:**
The name space of this capability.

**getDirectives**

```java
Map<String,String> getDirectives()
```

Returns the directives of this capability.

**Returns:**
A map of directive names to directive values for this capability, or an empty map if this capability has no directives.

**getAttributes**

```java
Map<String,Object> getAttributes()
```

Returns the attributes of this capability.

**Returns:**
A map of attribute names to attribute values for this capability, or an empty map if this capability has no attributes.

**getProviderRevision**

```java
BundleRevision getProviderRevision()
```
Returns the bundle revision declaring this capability.

**Returns:**

The bundle revision declaring this capability.
Interface FrameworkWiring

org.osgi.framework.wiring

All Superinterfaces:
org.osgi.framework.BundleReference

public interface FrameworkWiring
extends org.osgi.framework.BundleReference

Query and modify wiring information for the framework. The framework wiring object for the framework can be obtained by calling bundle.adapt(FrameworkWiring.class) on the system bundle. Only the system bundle can be adapted to a FrameworkWiring object.

The system bundle associated with this FrameworkWiring object can be obtained by calling org.osgi.framework.BundleReference.getBundle().

Version:
$Id: 820cd38ec470b064999d6eff0c2bb4a214bd8d9b $
ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection&lt;org.osgi.framework.Bundle&gt; bundles getDependencyClosure()</td>
<td>35</td>
</tr>
<tr>
<td>Returns the dependency closure for the specified bundles.</td>
<td></td>
</tr>
<tr>
<td>Collection&lt;org.osgi.framework.Bundle&gt; getRemovalPendingBundles()</td>
<td>35</td>
</tr>
<tr>
<td>Returns the bundles that have non-current, in use bundle wirings.</td>
<td></td>
</tr>
<tr>
<td>void refreshBundles(Collection&lt;org.osgi.framework.Bundle&gt; bundles, org.osgi.framework.FrameworkListener... listeners)</td>
<td>33</td>
</tr>
<tr>
<td>Refreshes the specified bundles.</td>
<td></td>
</tr>
<tr>
<td>boolean resolveBundles(Collection&lt;org.osgi.framework.Bundle&gt; bundles)</td>
<td>34</td>
</tr>
<tr>
<td>Resolves the specified bundles.</td>
<td></td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.BundleReference
getBundle

Method Detail

refreshBundles

void refreshBundles(Collection<org.osgi.framework.Bundle> bundles, org.osgi.framework.FrameworkListener... listeners)

Refreshes the specified bundles. This forces the update (replacement) or removal of packages exported by the specified bundles.

The technique by which the framework refreshes bundles may vary among different framework implementations. A 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 removal pending bundles.
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 bundles satisfying the dependencies have been installed in the Framework.

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.

When this process completes after the bundles are refreshed, the Framework will fire a Framework event of type FrameworkEvent.PACKAGES_REFRESHED to announce it has completed the bundle refresh. The specified FrameworkListeners are notified in the order specified. Each specified FrameworkListener will be called with a Framework event of type FrameworkEvent.PACKAGES_REFRESHED.

Parameters:
- bundles - The bundles to be refreshed, or null to refresh the removal pending bundles.
- listeners - Zero or more listeners to be notified when the bundle refresh 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 Bundles were not created by the same framework instance associated with this FrameworkWiring.
- SecurityException - If the caller does not have AdminPermission[System Bundle, RESOLVE] and the Java runtime environment supports permissions.

---

resolveBundles

boolean resolveBundles(Collection<org.osgi.framework.Bundle> bundles)

Resolves 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 no bundles are 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; false otherwise.

Throws:
- IllegalArgumentException - If the specified Bundles were not created by the same framework instance associated with this FrameworkWiring.
- SecurityException - If the caller does not have AdminPermission[System Bundle, RESOLVE] and the Java runtime environment supports permissions.
getRemovalPendingBundles

```java
Collection<org.osgi.framework.Bundle> getRemovalPendingBundles()
```

Returns the bundles that have non-current, in use bundle wirings. 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 Bundles which have non-current, in use BundleWirings, or an empty collection if there are no such bundles.

getDependencyClosure

```java
Collection<org.osgi.framework.Bundle> getDependencyClosure(Collection<org.osgi.framework.Bundle> bundles)
```

Returns 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 UNINSTALLED bundles that are removal pending.

**Parameters:**
- `bundles`: The initial bundles for which to generate the dependency closure.

**Returns:**
A collection containing a snapshot of the dependency closure of the specified bundles, or an empty collection if there were no specified bundles.

**Throws:**
- `IllegalArgumentException`: If the specified Bundles were not created by the same framework instance associated with this FrameworkWiring.
public interface WiredCapability
extends Capability

A wired capability that has been provided from a bundle wiring. This capability may or may not be required by any bundle wiring.

A wired capability represents a capability from a resolved bundle wiring.

Version:
$Id: f3b78da78b853ca418015f600923a715368c7349 $
ThreadSafe

Fields inherited from interface org.osgi.framework.wiring.Capability
BUNDLE_CAPABILITY, PACKAGE_CAPABILITY

Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleWiring getProviderWiring()</td>
<td>36</td>
</tr>
<tr>
<td>Collection&lt;BundleWiring&gt; getRequirerWirings()</td>
<td>36</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.wiring.Capability
getAttribute, getDirectives, getNamespace, getProviderRevision

Method Detail

getProviderWiring

BundleWiring getProviderWiring()

Returns the bundle wiring providing this capability.

Returns:
The bundle wiring providing this capability. If the bundle wiring providing this capability is not in use, null will be returned.

getRequirerWirings

Collection<BundleWiring> getRequirerWirings()

Returns the bundle wirings that require this capability.

The result of this method can change if this capability becomes required by additional bundle wirings.
**Interface Wired Capability**

**Returns:**
A collection containing a snapshot of the bundle wirings currently requiring this capability, or an empty collection if no bundle wirings require this capability. If the bundle wiring providing this capability is not in use, null will be returned.

---

### 7 Command Line API

Command line API is not considered relevant to this design. Hooks are a very low level concept which should not be accessed by the command line.

---

### 8 JMX API

JMX API is not considered relevant to this design. Hooks are a very low level concept which should not be accessed by JMX.

---

### 9 Initial Spec Chapter

Provide a link to where the Initial Spec Chapter can be found. The Initial Spec Chapter is typically written by someone other than the author(s) of this RFC and represents a rewrite of this document as close as possible to what will ultimately appear in the OSGi Specifications. It will be used by the Specification Editor as the basis for the ultimate specification chapter.

The spec template and writing guidelines can be found here:

10 Considered Alternatives

The list of alternatives is too long and painful to list. Take a look at the document history to see the long and painful journey up to this point.

11 Security Considerations

When Java permissions are in effect, this design is secured by ServicePermissions.

The bundle registering the various hook services must have the necessary ServicePermission.REGISTER. Since there are various hook services, we have fine grained control over what specific hooks a bundle can register.

12 Document Support

12.1 References


12.2 Author’s Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Thomas Watson</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>+1 512 286 9168</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:tjwatson@us.ibm.com">tjwatson@us.ibm.com</a></td>
</tr>
</tbody>
</table>
12.3 Acronyms and Abbreviations

12.4 End of Document
Abstract

Updates to the core specification including generifying the framework and other core API.
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.................................................................................................................................. 4

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

3 Problem Description.................................................................................................................... 5

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

5 Technical Solution....................................................................................................................... 6
  5.1 ee.mimimum........................................................................................................................ 6
  5.2 Framework API.................................................................................................................... 6
  5.3 Tracker API.......................................................................................................................... 7

6 Javadoc......................................................................................................................................... 7

7 Considered Alternatives.............................................................................................................. 182
  7.1 Version 2 API....................................................................................................................... 182
  7.2 Retroweaving...................................................................................................................... 182
  7.3 Moving all the PackageAdmin and StartLevel API into the Framework API...................... 183
  7.4 Removed BundleAdapter interface.................................................................................... 183
  7.5 Reverted changed to org.osgi.service.packageadmin and org.osgi.service.startlevel .... 183
  7.6 Remove specific methods for exported packages and required bundles.......................... 183
  7.7 Bundle specific entry methods could be added later to BundleRevsion......................... 183

8 Security Considerations.............................................................................................................. 183

9 Document Support....................................................................................................................... 184
  9.1 References.................................................................................................................................. 184
  9.2 Author’s Address................................................................................................................... 184
  9.3 Acronyms and Abbreviations............................................................................................... 184
  9.4 End of Document..................................................................................................................... 184

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. 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 redid PackageAdmin and StartLevel to exploit. Tracker API is also updated.</td>
</tr>
<tr>
<td>3rd draft</td>
<td>01/12/10</td>
<td>Updated based upon comments from CPEG Southampton f2f meeting. Removed BundleAdapter interface from signature of Bundle.adapt method. This allows, for example, for a bundle to be adapted to its ProtectionDomain. Then all other uses of BundleAdapter were replaced with BundleReference which is an interface we already have. BundlePackageAdmin is changed to return wiring information for fragments. isRemovalPending renamed to isCurrent and isStale renamed to isInUse in BundleWiring. 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>
<tr>
<td>5th draft</td>
<td>07/06/10</td>
<td>Retained adapt api over proposal to use Parameterized Services. Reverted org.osgi.service.packageadmin and org.osgi.service.startlevel and deprecated them. They are replaced by the new org.osgi.framework.wiring and org.osgi.framework.startlevel packages. Additional changes are still needed to reflect RFC 154 capabilities in the org.osgi.framework.wiring package.</td>
</tr>
<tr>
<td>6th draft</td>
<td>07/22/10</td>
<td>Added capabilities support for RFC 154. The changes now model exported packages and required bundles as capabilities.</td>
</tr>
<tr>
<td>7th draft</td>
<td>07/23/10</td>
<td>After conversation with Tom Watson and Richard Hall, I renamed BundleInfo to BundleRevision and remove the getEntry, getEntryPaths and getHeaders methods from BundleWiring. Those methods only applied to the host bundle and not the attached fragments. A BundleWiring represents a bundle and its attached fragments.</td>
</tr>
<tr>
<td>Final Draft</td>
<td>08/30/10</td>
<td>Updated Javadoc in preparation for RFC voting.</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 at 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 begin 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 types in the new org.osgi.framework.wiring and org.osgi.framework.startlevel packages. These new packages replace the org.osgi.service.packageadmin and org.osgi.service.startlevel packages, respectively, both of which will be deprecated.

RFC 154 is adding support for generic capabilities which will establish resolve-time wires between bundles that provide a capability and those that require the capability. The org.osgi.framework.wiring package now allows inspection of capability wiring between bundles and models exported packages and required bundles as capabilities by defining 2 capabilities.

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
## Package Summary

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Version Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.osgi.framework</td>
<td>Framework Package Version 1.6.</td>
<td>8</td>
</tr>
<tr>
<td>org.osgi.framework.startlevel</td>
<td>Framework Start Level Package Version 1.0.</td>
<td>143</td>
</tr>
<tr>
<td>org.osgi.framework.wiring</td>
<td>Framework Wiring Package Version 1.0.</td>
<td>149</td>
</tr>
<tr>
<td>org.osgi.util.tracker</td>
<td>Tracker Package Version 1.5.</td>
<td>164</td>
</tr>
</tbody>
</table>
# Package org.osgi.framework


See: [Description](#)

## Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AllServiceListe</strong>r</td>
<td>A <code>ServiceEvent</code> listener that does not filter based upon package wiring.</td>
<td>16</td>
</tr>
<tr>
<td><strong>Bundle</strong></td>
<td>An installed bundle in the Framework.</td>
<td>17</td>
</tr>
<tr>
<td><strong>BundleActivato</strong>r</td>
<td>Customizes the starting and stopping of a bundle.</td>
<td>36</td>
</tr>
<tr>
<td><strong>BundleContext</strong></td>
<td>A bundle's execution context within the Framework.</td>
<td>38</td>
</tr>
<tr>
<td><strong>BundleListener</strong></td>
<td>A <code>BundleEvent</code> listener.</td>
<td>62</td>
</tr>
<tr>
<td><strong>BundleReferen</strong>ce</td>
<td>A reference to a Bundle.</td>
<td>67</td>
</tr>
<tr>
<td><strong>Configurable</strong></td>
<td>Deprecated. As of 1.2.</td>
<td>68</td>
</tr>
<tr>
<td><strong>Constants</strong></td>
<td>Defines standard names for the OSGi environment system properties, service properties, and Manifest header attribute keys.</td>
<td>69</td>
</tr>
<tr>
<td><strong>Filter</strong></td>
<td>An RFC 1960-based Filter.</td>
<td>99</td>
</tr>
<tr>
<td><strong>FrameworkListe</strong>ner</td>
<td>A <code>FrameworkEvent</code> listener.</td>
<td>107</td>
</tr>
<tr>
<td><strong>ServiceFactory</strong></td>
<td>Allows services to provide customized service objects in the OSGi environment.</td>
<td>124</td>
</tr>
<tr>
<td><strong>ServiceListener</strong></td>
<td>A <code>ServiceEvent</code> listener.</td>
<td>126</td>
</tr>
<tr>
<td><strong>ServiceReferen</strong>ce</td>
<td>A reference to a service.</td>
<td>131</td>
</tr>
<tr>
<td><strong>ServiceRegistr</strong>ation</td>
<td>A registered service.</td>
<td>135</td>
</tr>
<tr>
<td><strong>SynchronousBu</strong>ndleListener</td>
<td>A synchronous <code>BundleEvent</code> listener.</td>
<td>137</td>
</tr>
</tbody>
</table>

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AdminPermissio</strong>n</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><strong>BundleEvent</strong></td>
<td>An event from the Framework describing a bundle lifecycle change.</td>
<td>52</td>
</tr>
<tr>
<td><strong>BundlePermis</strong>sion</td>
<td>A bundle's authority to require or provide a bundle or to receive or attach fragments.</td>
<td>63</td>
</tr>
<tr>
<td><strong>FrameworkEve</strong>nt</td>
<td>A general event from the Framework.</td>
<td>102</td>
</tr>
<tr>
<td><strong>FrameworkUtil</strong></td>
<td>Framework Utility class.</td>
<td>108</td>
</tr>
<tr>
<td><strong>PackagePermis</strong>sion</td>
<td>A bundle's authority to import or export a package.</td>
<td>114</td>
</tr>
<tr>
<td><strong>ServiceEvent</strong></td>
<td>An event from the Framework describing a service lifecycle change.</td>
<td>118</td>
</tr>
<tr>
<td><strong>ServicePermis</strong>sion</td>
<td>A bundle's authority to register or get a service.</td>
<td>127</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>Version identifier for bundles and packages.</td>
<td>138</td>
</tr>
</tbody>
</table>

## Exception Summary

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BundleExceptio</strong>n</td>
<td>A Framework exception used to indicate that a bundle lifecycle problem occurred.</td>
<td>56</td>
</tr>
<tr>
<td><strong>InvalidSyntaxEx</strong>ception</td>
<td>A Framework exception used to indicate that a filter string has an invalid syntax.</td>
<td>111</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.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.framework; version="[1.6,2.0)"
```
Framework Update

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>StartLevel.setStartLevel</td>
</tr>
<tr>
<td>context</td>
<td>StartLevel.setInitialBundleStartLevel</td>
</tr>
<tr>
<td></td>
<td>Bundle.getBundleContext</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:

$Id: 06eb00bd1e8f657d4d58f5e529c7803897501827 $

ThreadSafe
Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS</td>
<td>The action string class. The class action implies the resolve action.</td>
</tr>
<tr>
<td>CONTEXT</td>
<td>The action string context.</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>The action string execute.</td>
</tr>
<tr>
<td>EXTENSIONLIFECYCLE</td>
<td>The action string extensionLifecycle.</td>
</tr>
<tr>
<td>LIFECYCLE</td>
<td>The action string lifecycle.</td>
</tr>
<tr>
<td>LISTENER</td>
<td>The action string listener.</td>
</tr>
<tr>
<td>METADATA</td>
<td>The action string metadata.</td>
</tr>
<tr>
<td>RESOLVE</td>
<td>The action string resolve.</td>
</tr>
<tr>
<td>RESOURCE</td>
<td>The action string resource.</td>
</tr>
<tr>
<td>STARTLEVEL</td>
<td>The action string startlevel.</td>
</tr>
</tbody>
</table>

Constructor Summary

AdminPermission()

Creates a new AdminPermission object that matches all bundles and has all actions.

AdminPermission(String filter, String actions)

Create a new AdminPermission.

AdminPermission(Bundle bundle, String actions)

Creates a new requested AdminPermission object to be used by the code that must perform checkPermission.

Method Summary

boolean equals(Object obj)

Determines the equality of two AdminPermission objects.

String getActions()

Returns the canonical string representation of the AdminPermission actions.

int hashCode()

Returns the hash code value for this object.

boolean implies(Permission p)

Determines if the specified permission is implied by this object.

PermissionCollection newPermissionCollection()

Returns a new PermissionCollection object suitable for storing AdminPermissionS.

Field Detail

CLASS

public static final String CLASS = "class"

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

public static final String EXECUTE = "execute"

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

Since: 1.3

EXTENSIONLIFECYCLE

public static final String EXTENSIONLIFECYCLE = "extensionLifecycle"

The action string extensionLifecycle.

Since: 1.3

LIFECYCLE

public static final String LIFECYCLE = "lifecycle"

The action string lifecycle.

Since: 1.3

LISTENER

public static final String LISTENER = "listener"

The action string listener.

Since: 1.3

METADATA

public static final String METADATA = "metadata"

The action string metadata.

Since: 1.3

RESOLVE

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.
Interface `AllServiceListener`

```java
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: $Id: 35cee8a49e89b7b222aa3f85e1af0b4a4b550ce6 $

See Also: `ServiceEvent`, `ServicePermission`

ThreadSafe

<table>
<thead>
<tr>
<th>Methods inherited from interface org.osgi.framework.ServiceListener</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>serviceChanged</code></td>
</tr>
</tbody>
</table>
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.

Version: $Id: 2e3940b8b9ac15016d589cadcb7349fe3c4274dd $

ThreadSafe

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int ACTIVE</td>
<td>20</td>
</tr>
<tr>
<td>The bundle is now running.</td>
<td></td>
</tr>
<tr>
<td>int INSTALLED</td>
<td>19</td>
</tr>
<tr>
<td>The bundle is installed but not yet resolved.</td>
<td></td>
</tr>
<tr>
<td>int RESOLVED</td>
<td>19</td>
</tr>
<tr>
<td>The bundle is resolved and is able to be started.</td>
<td></td>
</tr>
<tr>
<td>int SIGNERS_ALL</td>
<td>21</td>
</tr>
<tr>
<td>Request that all certificates used to sign the bundle be returned.</td>
<td></td>
</tr>
<tr>
<td>int SIGNERS_TRUSTED</td>
<td>21</td>
</tr>
<tr>
<td>Request that only certificates used to sign the bundle that are trusted by the framework be returned.</td>
<td></td>
</tr>
<tr>
<td>int START_ACTIVATION_POLICY</td>
<td>21</td>
</tr>
<tr>
<td>The bundle start operation must activate the bundle according to the bundle's declared activation policy.</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>START_TRANSIENT</td>
</tr>
<tr>
<td>int</td>
<td>STARTING</td>
</tr>
<tr>
<td>int</td>
<td>STOP_TRANSIENT</td>
</tr>
<tr>
<td>int</td>
<td>STOPPING</td>
</tr>
<tr>
<td>int</td>
<td>UNINSTALLED</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A adapt(Class&lt;A&gt; type)</td>
<td>Adapt a bundle to the specified type.</td>
</tr>
<tr>
<td>Enumeration&lt;URL&gt; findEntries(String path, String filePattern, boolean recurse)</td>
<td>Returns entries in this bundle and its attached fragments.</td>
</tr>
<tr>
<td>BundleContext getBundleContext()</td>
<td>Returns this bundle's BundleContext.</td>
</tr>
<tr>
<td>long getBundleId()</td>
<td>Returns this bundle's unique identifier.</td>
</tr>
<tr>
<td>File getDataFile(String filename)</td>
<td>Creates a File object for a file in the persistent storage area provided for this bundle by the Framework.</td>
</tr>
<tr>
<td>URL 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; 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; getHeaders()</td>
<td>Returns this bundle's Manifest headers and values.</td>
</tr>
<tr>
<td>Dictionary&lt;String,String&gt; getHeaders(String locale)</td>
<td>Returns this bundle's Manifest headers and values localized to the specified locale.</td>
</tr>
<tr>
<td>long getLastModified()</td>
<td>Returns the time when this bundle was last modified.</td>
</tr>
<tr>
<td>String getLocation()</td>
<td>Returns this bundle's location identifier.</td>
</tr>
<tr>
<td>ServiceReference&lt;[]&gt; 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 getResource(String name)</td>
<td>Find the specified resource from this bundle's class loader.</td>
</tr>
<tr>
<td>Enumeration&lt;URL&gt; getResources(String name)</td>
<td>Find the specified resources from this bundle's class loader.</td>
</tr>
<tr>
<td>ServiceReference&lt;[]&gt; 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>
<tr>
<td>Map&lt;X509Certificate, List&lt;X509Certificate&gt;&gt; getSignerCertificates(int signersType)</td>
<td>Return the certificates for the signers of this bundle and the certificate chains for those signers.</td>
</tr>
<tr>
<td>int getState()</td>
<td>Returns this bundle's current state.</td>
</tr>
</tbody>
</table>
Framework Update

Final

30 August 2010

String 

**getSymbolicName**() 

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

Version 

**getVersion**() 

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

Boolean 

**hasPermission**(Object permission) 

Determines if this bundle has the specified permissions.

Class<?> 

**loadClass**(String name) 

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

Void 

**start**() 

Starts this bundle with no options.

Void 

**start**(int options) 

Starts this bundle.

Void 

**stop**() 

Stops this bundle with no options.

Void 

**stop**(int options) 

Stops this bundle.

Void 

**uninstall**() 

Uninstalls this bundle.

Void 

**update**() 

Updates this bundle.

Void 

**update**(InputStream input) 

Updates this bundle from an InputStream.

---

**Field Detail**

**UNINSTALLED**

public static final int UNINSTALLED = 1

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.

**INSTALLED**

public static final int INSTALLED = 2

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**

```java
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**

```java
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**

```java
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**

```java
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

```java
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

```java
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

```java
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

```java
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)

### Method Detail

#### getState

```java
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

```java
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

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)
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 UNINSTALLING 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 then set this bundle’s persistent autostart setting to 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 UNINSTALLING, 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


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
def 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 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.
- **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:**

stop(), start()

---

**update**

```java
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.
- **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], and the Java Runtime Environment supports permissions.

**See Also:**
- `stop()`

---

### 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:**
An unmodifiable 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.security.ProtectionDomain; 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.

getResources

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()
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 + "" + 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: An unmodifiable 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

`Enumeration<URL> getResources(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

`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 no matching 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 attachment 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

dataFile

File getDataFile(String filename)
Creates a File object for a file in the persistent storage area provided for this bundle by the Framework. This method will return null if the platform does not have file system support or this bundle is a fragment bundle.

A File object for the base directory of the persistent storage area provided for this 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 this bundle has the java.io.FilePermission with actions read, write, delete for all files (recursively) in the persistent storage area provided for this 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 or this bundle is a fragment bundle.

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

**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: <i>class-name</i>

where <i>class-name</i> 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().

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void start(BundleContext context)</td>
<td>36</td>
</tr>
<tr>
<td>void stop(BundleContext context)</td>
<td>37</td>
</tr>
</tbody>
</table>

Method Detail

start

void start(BundleContext context) throws Exception

Called when this bundle is started so the Framework can perform the bundle-specific activities necessary to start this bundle. This method can be used to register services or to allocate any resources that this bundle needs.

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.
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.
Interface BundleContext

```java
go.osgi.framework
All Superinterfaces:
    BundleReference
```

```java
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:**

`$Id: 89eb063ec09f44477a17c89c31926f7f3b46ab38 $`

**ThreadSafe**

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void addBundleListener(BundleListener listener)</code></td>
<td>43</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><code>void addFrameworkListener(FrameworkListener listener)</code></td>
<td>43</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><code>void addServiceListener(ServiceListener listener)</code></td>
<td>42</td>
</tr>
<tr>
<td>Adds the specified ServiceListener object to the context bundle's list of listeners.</td>
<td></td>
</tr>
<tr>
<td><code>void addServiceListener(ServiceListener listener, String filter)</code></td>
<td>42</td>
</tr>
<tr>
<td>Adds the specified ServiceListener object with the specified filter to the context bundle's list of listeners.</td>
<td></td>
</tr>
<tr>
<td><code>Filter createFilter(String filter)</code></td>
<td>50</td>
</tr>
<tr>
<td>Creates a Filter object.</td>
<td></td>
</tr>
</tbody>
</table>
### Method Detail

#### getBundle

**Bundle getBundle()**

Returns the Bundle object associated with this BundleContext.

#### getBundle

**Bundle getBundle(long id)**

Returns the bundle with the specified identifier.

#### getBundles

**Bundle[] getBundles()**

Returns a list of all installed bundles.

#### getDataFile

**File getDataFile(String filename)**

Creates a File object for a file in the persistent storage area provided for the bundle by the Framework.

#### getServiceReference

**ServiceReference<S> getServiceReference(Class<S> clazz)**

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

**ServiceReference<? extends S> getServiceReference(String clazz)**

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

#### getServiceReferences

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

Returns a collection of ServiceReference objects.

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

Returns an array of ServiceReference objects.

#### installBundle

**Bundle installBundle(String location)**

Installs a bundle from the specified location identifier.

**Bundle installBundle(String location, InputStream input)**

Installs a bundle from the specified InputStream object.

#### registerService

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

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

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

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

#### unregisterService

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

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

#### removeBundleListener

**void removeBundleListener(BundleListener listener)**

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

#### removeFrameworkListener

**void removeFrameworkListener(FrameworkListener listener)**

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

#### removeServiceListener

**void removeServiceListener(ServiceListener listener)**

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

#### ungetService

**boolean ungetService(ServiceReference<?> reference)**

Releases the service object referenced by the specified ServiceReference object.

---

### Method getBundle

**String getProperty(String key)**

Returns the value of the specified property.
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**

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**

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:

- If a bundle containing the same location identifier is already installed, the Bundle object for that bundle is returned.
- The bundle's content is read from the input stream. If this fails, a BundleException is thrown.
- 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.
- The bundle's state is set to INSTALLED.
- A bundle event of type BundleEvent.INSTALLED is fired.
- The Bundle Object for the newly or previously installed bundle is returned.

**Postconditions, no exceptions thrown**

1. getState() in {INSTALLED, RESOLVED}.
2. Bundle has a unique ID.

**Postconditions, when an exception is thrown**

- Bundle is not installed. If there was an existing bundle for the specified location, then that bundle must still be in the state it was prior to calling this method.
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

Bundle installBundle(String location)
throws BundleException

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

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.

getBundles

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)
throws InvalidSyntaxException
```

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 \( l==\text{listener} \), 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:
- InvalidSyntaxException - 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.

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 \( l == \text{listener} \), 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 \( \text{listener} \) 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.

---

### 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
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:
IllegalArgumentException - If this BundleContext is no longer valid.

registerService

ServiceRegistration<?> registerService(String[] clazzes, 
Object service, 
Dictionary<String,?> 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.
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.getProperties() 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:

1. service is null.
2. service is not a ServiceFactory object and is not an instance of all the named classes in clazzes.
3. 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.
registerService

ServiceRegistration<?< registervService(String clazz, Object service, Dictionary<String,?> 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<
 registerService(Class<S> clazz, S service, Dictionary<String,?> 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**

```java
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.

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

---

**getAllServiceReferences**

```java
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

---

### getServiceReference

`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)`

---

### getServiceReference

`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)
throws InvalidSyntaxException
```

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 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

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 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 service'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 instance of 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

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:

- If the context bundle's use count for the service is zero or the service has been unregistered, false is returned.
- The context bundle's use count for this service is decremented by one.
- 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.
- 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.

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

```java
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 SynchronousBundleListeners and BundleListeners 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:
```text
$Id: ac96d1ab8c5b491539af238ad41071b5bb9ae95 $```

See Also:
```
BundleListener, SynchronousBundleListener
```

Immutable

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int INSTALLED</td>
<td>53</td>
</tr>
<tr>
<td>The bundle has been installed.</td>
<td></td>
</tr>
<tr>
<td>static int LAZY_ACTIVATION</td>
<td>55</td>
</tr>
<tr>
<td>The bundle will be lazily activated.</td>
<td></td>
</tr>
<tr>
<td>static int RESOLVED</td>
<td>54</td>
</tr>
<tr>
<td>The bundle has been resolved.</td>
<td></td>
</tr>
<tr>
<td>static int STARTED</td>
<td>53</td>
</tr>
<tr>
<td>The bundle has been started.</td>
<td></td>
</tr>
<tr>
<td>static int STARTING</td>
<td>54</td>
</tr>
<tr>
<td>The bundle is about to be activated.</td>
<td></td>
</tr>
<tr>
<td>static int STOPPED</td>
<td>53</td>
</tr>
<tr>
<td>The bundle has been stopped.</td>
<td></td>
</tr>
<tr>
<td>static int STOPPING</td>
<td>54</td>
</tr>
<tr>
<td>The bundle is about to deactivated.</td>
<td></td>
</tr>
<tr>
<td>static int UNINSTALLED</td>
<td>53</td>
</tr>
<tr>
<td>The bundle has been uninstalled.</td>
<td></td>
</tr>
<tr>
<td>static int UNRESOLVED</td>
<td>54</td>
</tr>
<tr>
<td>The bundle has been unresolved.</td>
<td></td>
</tr>
<tr>
<td>static int UPDATED</td>
<td>53</td>
</tr>
<tr>
<td>The bundle has been updated.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleEvent(int type, Bundle bundle)</td>
<td>55</td>
</tr>
<tr>
<td>Creates a bundle event of the specified type.</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 which had a lifecycle change.</td>
</tr>
<tr>
<td>int</td>
<td>getType()</td>
<td>Returns the type of lifecycle event.</td>
</tr>
</tbody>
</table>

Field Detail

INSTALLED

public static final int INSTALLED = 1

The bundle has been installed.

See Also:

BundleContext.installBundle(String)

STARTED

public static final int STARTED = 2

The bundle has been started.

The bundle's BundleActivator.start method has been executed if the bundle has a bundle activator class.

See Also:

Bundle.start()

STOPPED

public static final int STOPPED = 4

The bundle has been stopped.

The bundle's BundleActivator.stop method has been executed if the bundle has a bundle activator class.

See Also:

Bundle.stop()

UPDATED

public static final int UPDATED = 8

The bundle has been updated.

See Also:

Bundle.update()

UNINSTALLED

public static final int UNINSTALLED = 16
The bundle has been uninstalled.

See Also:
Bundle.uninstall()

---

**RESOLVED**

public static final int RESOLVED = 32

The bundle has been resolved.

Since: 1.3
See Also: Bundle.RESOLVED

---

**UNRESOLVED**

public static final int UNRESOLVED = 64

The bundle has been unresolved.

Since: 1.3
See Also: Bundle.INSTALLED

---

**STARTING**

public static final int STARTING = 128

The bundle is about to be activated.

The bundle's BundleActivator.start method is about to be called if the bundle has a bundle activator class. This event is only delivered to SynchronousBundleListeners. It is not delivered to BundleListeners.

Since: 1.3
See Also: Bundle.start()

---

**STOPPING**

public static final int STOPPING = 256

The bundle is about to be deactivated.

The bundle's BundleActivator.stop method is about to be called if the bundle has a bundle activator class. This event is only delivered to SynchronousBundleListeners. It is not delivered to BundleListeners.

Since: 1.3
See Also: Bundle.stop()
LAZY_ACTIVATION

```java
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 SynchronousBundleListeners. It is not delivered to BundleListeners.

Since: 1.4

Constructor Detail

BundleEvent

```java
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

```java
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

```java
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:
  $Id: ad1058a00e2bcb6e6c39b1acc2c657eacc972f84 $

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int ACTIVATOR_ERROR</td>
<td>58</td>
</tr>
<tr>
<td>The bundle activator was in error.</td>
<td></td>
</tr>
<tr>
<td>static int DUPLICATE_BUNDLE_ERROR</td>
<td>59</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>57</td>
</tr>
<tr>
<td>The operation was invalid.</td>
<td></td>
</tr>
<tr>
<td>static int MANIFEST_ERROR</td>
<td>58</td>
</tr>
<tr>
<td>The bundle manifest was in error.</td>
<td></td>
</tr>
<tr>
<td>static int NATIVECODE_ERROR</td>
<td>58</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>58</td>
</tr>
<tr>
<td>The bundle was not resolved.</td>
<td></td>
</tr>
<tr>
<td>static int SECURITY_ERROR</td>
<td>58</td>
</tr>
<tr>
<td>The operation failed due to insufficient permissions.</td>
<td></td>
</tr>
<tr>
<td>static int START_TRANSIENT_ERROR</td>
<td>59</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>58</td>
</tr>
<tr>
<td>The operation failed to complete the requested lifecycle state change.</td>
<td></td>
</tr>
<tr>
<td>static int UNSPECIFIED</td>
<td>57</td>
</tr>
<tr>
<td>No exception type is specified.</td>
<td></td>
</tr>
<tr>
<td>static int UNSUPPORTED_OPERATION</td>
<td>57</td>
</tr>
<tr>
<td>The operation was unsupported.</td>
<td></td>
</tr>
</tbody>
</table>
### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BundleException(String msg)</strong></td>
<td>Creates a BundleException with the specified message.</td>
<td>59</td>
</tr>
<tr>
<td><strong>BundleException(String msg, int type)</strong></td>
<td>Creates a BundleException with the specified message and type.</td>
<td>60</td>
</tr>
<tr>
<td><strong>BundleException(String msg, int type, Throwable cause)</strong></td>
<td>Creates a BundleException with the specified message, type and exception cause.</td>
<td>59</td>
</tr>
<tr>
<td><strong>BundleException(String msg, Throwable cause)</strong></td>
<td>Creates a BundleException with the specified message and exception cause.</td>
<td>59</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throwable <strong>getCause</strong>()</td>
<td>Returns the cause of this exception or null if no cause was set.</td>
<td>60</td>
</tr>
<tr>
<td>Throwable <strong>getNestedException</strong>()</td>
<td>Returns the cause of this exception or null if no cause was specified when this exception was created.</td>
<td>60</td>
</tr>
<tr>
<td>int <strong>getType</strong>()</td>
<td>Returns the type for this exception or UNSPECIFIED if the type was unspecified or unknown.</td>
<td>61</td>
</tr>
<tr>
<td>Throwable <strong>initCause</strong>(Throwable cause)</td>
<td>Initializes the cause of this exception to the specified value.</td>
<td>60</td>
</tr>
</tbody>
</table>

### Field Detail

**UNSPECIFIED**

```java
public static final int UNSPECIFIED = 0
```

No exception type is specified.

**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
public static final int MANIFEST_ERROR = 3

    The bundle manifest was in error.

    Since: 1.5

---

public static final int RESOLVE_ERROR = 4

    The bundle was not resolved.

    Since: 1.5

---

public static final int ACTIVATOR_ERROR = 5

    The bundle activator was in error.

    Since: 1.5

---

public static final int SECURITY_ERROR = 6

    The operation failed due to insufficient permissions.

    Since: 1.5

---

public static final int STATECHANGE_ERROR = 7

    The operation failed to complete the requested lifecycle state change.

    Since: 1.5

---

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
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.
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().

cause

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.
**getType**

```java
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.

Version: $Id: 77cdaebd3ac97c6798fc3043957abd1bd6d01ccbb $
See Also: BundleEvent
NotThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void bundleChanged(BundleEvent event)</td>
<td>62</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.
Class BundlePermission

org.osgi.framework

defined by

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

All Implemented Interfaces:
Guard, Serializable

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: $Id: d30c9c987cc13007ed19d3a9fdd11b00739591c0 $
ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>Static String</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAGMENT</td>
<td>64</td>
</tr>
<tr>
<td>HOST</td>
<td>64</td>
</tr>
<tr>
<td>PROVIDE</td>
<td>64</td>
</tr>
<tr>
<td>REQUIRE</td>
<td>64</td>
</tr>
</tbody>
</table>

The action strings:
- FRAGMENT: The action string fragment.
- HOST: The action string host.
- PROVIDE: The action string provide.
- REQUIRE: The action string require.

Constructor Summary

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.

64

Method Summary

<table>
<thead>
<tr>
<th>boolean equals(Object obj)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>65</td>
</tr>
</tbody>
</table>

Determines the equality of two BundlePermission objects.

<table>
<thead>
<tr>
<th>String getActions()</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>65</td>
</tr>
</tbody>
</table>

Returns the canonical string representation of the BundlePermission actions.
int hashCode()

Returns the hash code value for this object.

boolean implies(Permission p)

Determines if the specified permission is implied by this object.

Permission Collection newPermissionCollection()

Returns a new PermissionCollection object suitable for storing BundlePermission objects.

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.

Parameters:
    symbolicName - The bundle symbolic name.
    actions - provide,require,host,fragment (canonical order).
Method Detail

implies

```java
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" -> x.y.z, "provide" is true
*, "require" -> x.y, "require" is true
*, "provide" -> x.y, "require" is true
x.y, "provide" -> 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

```java
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

```java
public PermissionCollection newPermissionCollection()
```

Returns a new PermissionCollection object suitable for storing BundlePermission objects.

Overrides: newPermissionCollection in class BasicPermission

Returns:
- A new PermissionCollection object.

equals

```java
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: $Id: adbba3dfc5ac9af0b534c3008ffab1b29984c4bb $
ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bundle getBundle ()</td>
<td>67</td>
</tr>
</tbody>
</table>

Returns the Bundle object associated with this BundleReference.

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:
$Id: 29705c0c238aa456cda1b1a13458079bf1542771 $
Interface Constants

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: $Id: d673da95c2e445217360c5c3b17016a4f40098b5 $

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String ACTIVATION_LAZY</td>
<td>86</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>86</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's activation policy.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_ACTIVATOR</td>
<td>76</td>
</tr>
<tr>
<td>Manifest header attribute identifying the bundle's activator class.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_CATEGORY</td>
<td>74</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's category.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_CLASSPATH</td>
<td>74</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>76</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>74</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's copyright information.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_DESCRIPTION</td>
<td>74</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>76</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>80</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>80</td>
</tr>
<tr>
<td>Default value for the Bundle-Localization manifest header.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_MANIFESTVERSION</td>
<td>81</td>
</tr>
<tr>
<td>Manifest header identifying the bundle manifest version.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_NAME</td>
<td>74</td>
</tr>
<tr>
<td>Manifest header identifying the bundle's name.</td>
<td></td>
</tr>
<tr>
<td>String BUNDLE_NATIVECODE</td>
<td>75</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>78</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>
<tr>
<td>String</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BUNDLE_NATIVECODE_OSNAME</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_OSVERSION</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_NATIVECODEPROCESSOR</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>EXTENSIONDirective</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 environment property identifying packages for which the Framework must delegate class loading to the parent class loader of the bundle.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FRAMEWORK_BOOTDELEGATION</td>
<td></td>
</tr>
<tr>
<td>FRAMEWORK_BUNDLE_PARENT</td>
<td>Specifies the parent class loader type for all bundle class loaders.</td>
</tr>
<tr>
<td>FRAMEWORK_BUNDLE_PARENT_APP</td>
<td>Specifies to use the application class loader as the parent class loader for all bundle class loaders.</td>
</tr>
<tr>
<td>FRAMEWORK_BUNDLE_PARENT_BOOT</td>
<td>Specifies to use of the boot class loader as the parent class loader for all bundle class loaders.</td>
</tr>
<tr>
<td>FRAMEWORK_BUNDLE_PARENT_EXT</td>
<td>Specifies to use the extension class loader as the parent class loader for all bundle class loaders.</td>
</tr>
<tr>
<td>FRAMEWORK_BUNDLE_PARENT_FRAMEWORK</td>
<td>Specifies to use the framework class loader as the parent class loader for all bundle class loaders.</td>
</tr>
<tr>
<td>FRAMEWORK_COMMANDABSPATH</td>
<td>Specified the substitution string for the absolute path of a file.</td>
</tr>
<tr>
<td>FRAMEWORK_EXECPERMISION</td>
<td>Specifies an optional OS specific command to set file permissions on extracted native code.</td>
</tr>
<tr>
<td>FRAMEWORK_EXECUTIONENVIRONMENT</td>
<td>Framework environment property identifying execution environments provided by the Framework.</td>
</tr>
<tr>
<td>FRAMEWORK_JARURLS</td>
<td>Specifies that an returned URLs from bundle class loaders must be a jar: or file: URL if set to any value.</td>
</tr>
<tr>
<td>FRAMEWORK_LANGUAGE</td>
<td>Framework environment property identifying the Framework implementation language (see ISO 639 for possible values).</td>
</tr>
<tr>
<td>FRAMEWORK_LIBRARY_EXTENSIONS</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>
<tr>
<td>FRAMEWORK_OS_NAME</td>
<td>Framework environment property identifying the Framework host-computer's operating system.</td>
</tr>
<tr>
<td>FRAMEWORK_OS_VERSION</td>
<td>Framework environment property identifying the Framework host-computer's operating system version number.</td>
</tr>
<tr>
<td>FRAMEWORK_PROCESSOR</td>
<td>Framework environment property identifying the Framework host-computer's processor name.</td>
</tr>
<tr>
<td>FRAMEWORK_SECURITY</td>
<td>Specifies the type of security manager the framework must use.</td>
</tr>
<tr>
<td>FRAMEWORK_SECURITY.osgi</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>
<tr>
<td>FRAMEWORK_STORAGE</td>
<td>Specified the persistent storage area used by the framework.</td>
</tr>
<tr>
<td>FRAMEWORK_STORAGE_CLEAN</td>
<td>Specifies if and when the persistent storage area for the framework should be cleaned.</td>
</tr>
<tr>
<td>FRAMEWORK_STORAGE_CLEAN_ONFIRSTINIT</td>
<td>Specifies that the framework storage area must be cleaned before the framework is initialized for the first time.</td>
</tr>
<tr>
<td>FRAMEWORK_SYSTEMPACKAGES</td>
<td>Framework environment property identifying packages which the system bundle must export.</td>
</tr>
<tr>
<td><strong>String</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FRAMEWORK_SYSTEM_PACKAGES_EXTRA</td>
<td>Framework environment property identifying extra packages which the system bundle must export from the current execution environment.</td>
</tr>
<tr>
<td>FRAMEWORK_TRUST_REPOSITORIES</td>
<td>Specifies the trust repositories used by the framework.</td>
</tr>
<tr>
<td>FRAMEWORK_UUID</td>
<td>Framework environment property identifying the Framework's universally unique identifier (UUID).</td>
</tr>
<tr>
<td>FRAMEWORK_VENDOR</td>
<td>Framework environment property identifying the Framework implementation vendor.</td>
</tr>
<tr>
<td>FRAMEWORK_VERSION</td>
<td>Framework environment property identifying the Framework version.</td>
</tr>
<tr>
<td>FRAMEWORK_WINDOWSYSTEM</td>
<td>Specifies the current windowing system.</td>
</tr>
<tr>
<td>IMPORT_PACKAGE</td>
<td>Manifest header identifying the packages on which the bundle depends.</td>
</tr>
<tr>
<td>IMPORT_SERVICE</td>
<td>Deprecated. As of 1.2.</td>
</tr>
<tr>
<td>INCLUDE_DIRECTIVE</td>
<td>Manifest header directive identifying a list of classes to include in the exported package.</td>
</tr>
<tr>
<td>MANDATORY_DIRECTIVE</td>
<td>Manifest header directive identifying names of matching attributes which must be specified by matching Import-Package statements in the Export-Package manifest header.</td>
</tr>
<tr>
<td>OBJECTCLASS</td>
<td>Service property identifying all of the class names under which a service was registered in the Framework.</td>
</tr>
<tr>
<td>PACKAGE_SPECIFICATION_VERSION</td>
<td>Deprecated. As of 1.3.</td>
</tr>
<tr>
<td>REMOTE_CONFIGS_SUPPORTED</td>
<td>Service property identifying the configuration types supported by a distribution provider.</td>
</tr>
<tr>
<td>REMOTE_INTENTS_SUPPORTED</td>
<td>Service property identifying the intents supported by a distribution provider.</td>
</tr>
<tr>
<td>REQUIRE_BUNDLE</td>
<td>Manifest header identifying the symbolic names of other bundles required by the bundle.</td>
</tr>
<tr>
<td>RESOLUTION_DIRECTIVE</td>
<td>Manifest header directive identifying the resolution type in the Import-Package or Require-Bundle manifest header.</td>
</tr>
<tr>
<td>RESOLUTION_MANDATORY</td>
<td>Manifest header directive value identifying a mandatory resolution type.</td>
</tr>
<tr>
<td>RESOLUTION_OPTIONAL</td>
<td>Manifest header directive value identifying an optional resolution type.</td>
</tr>
<tr>
<td>SELECTION_FILTER_ATTRIBUTE</td>
<td>Manifest header attribute is used for selection by filtering based upon system properties.</td>
</tr>
<tr>
<td>SERVICE_DESCRIPTION</td>
<td>Service property identifying a service's description.</td>
</tr>
<tr>
<td>SERVICE_EXPORTED_CONFIGS</td>
<td>Service property identifying the configuration types that should be used to export the service.</td>
</tr>
<tr>
<td>SERVICE_EXPORTED_INTENTS</td>
<td>Service property identifying the intents that the distribution provider must implement to distribute the service.</td>
</tr>
<tr>
<td>SERVICE_EXPORTED_INTENTS_EXTRA</td>
<td>Service property identifying the extra intents that the distribution provider must implement to distribute the service.</td>
</tr>
<tr>
<td>SERVICE_EXPORTED_INTERFACES</td>
<td>Service property marking the service for export.</td>
</tr>
<tr>
<td>String</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>service_id</td>
<td>Service property identifying a service's registration number.</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 Require-Bundle manifest header.</td>
</tr>
<tr>
<td>system_bundle_location</td>
<td>Location identifier of the OSGi system bundle, 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 system bundle.</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".
public static final String SYSTEM_BUNDLE_SYMBOLICNAME = "system.bundle"

Alias for the symbolic name of the OSGi system bundle. It is defined to be "system.bundle".

Since: 1.3

---

public static final String BUNDLE_CATEGORY = "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.

---

public static final String BUNDLE_CLASSPATH = "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.

---

public static final String BUNDLE_COPYRIGHT = "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.

---

public static final String BUNDLE_DESCRIPTION = "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.

---

public static final String BUNDLE_NAME = "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"

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**

```java
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**

```java
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**

```java
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_OSNAME**

```java
public static final String BUNDLE_NATIVECODE_OSNAME = "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
public static final String BUNDLE_NATIVECODE_OSVERSION = "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:

    Bundle-NativeCode: http.so ; osversion="2.34" ...

See Also:  
BUNDLE_NATIVECODE

BUNDLE_NATIVECODE_LANGUAGE

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:

    Bundle-NativeCode: http.so ; language=nl_be ...

See Also:  
BUNDLE_NATIVECODE

BUNDLE_REQUIREDEXECUTIONENVIRONMENT

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

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_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

```java
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:

```plaintext
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

```java
public static final String FRAGMENT_HOST = "Fragment-Host"
```

Manifest header identifying the symbolic name of another bundle for which 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

```java
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:

```plaintext
Bundle-NativeCode: libgtk.so; selection-filter="(ws=gtk)"; ...
```

Since: 1.3

See Also: [BUNDLE_NATIVECODE](#)

### BUNDLE_MANIFESTVERSION

```java
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:

Export-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
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:="mandatory"
Require-Bundle: com.acme.module.test; resolution:="mandatory"

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

```
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

```
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: REQUIRE_BUNDLE, 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"
```
EXTENSION_FRAMEWORK

public 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, FRAGMENT_HOST, EXTENSION_BOOTCLASSPATH

EXTENSION_BOOTCLASSPATH

public 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

public 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

public 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.
FRAMEWORK_PROCESSOR

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

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

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

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

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
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

Copyright © IBM Corporation 2010 All Rights Reserved
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

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`.

Since:
1.5
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 FRAMEWORK_BUNDLE_PARENT_FRAMEWORK.

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 FRAMEWORK_BUNDLE_PARENT_APP.

Since: 1.5
See Also: FRAMEWORK_BUNDLE_PARENT

OBJECTCLASS

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 `propertiesDictionary` 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 `propertiesDictionary` 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.

---
**SERVICE_DESCRIPTION**

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**

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. A new UUID is generated by the org.osgi.framework.launch.Framework.init() method each time a framework is initialized. 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

---

**FRAMEWORK_JARURLS**

public static final String FRAMEWORK_JARURLS = "org.osgi.framework.jarurls"

Specifies that returned URLs from bundle class loaders must be a jar: or file: URL if set to any value. This property must be set in the launching parameters of the framework. If a Framework cannot support this property it must throw an Illegal Argument Exception during its initialization. URLs obtained through the OSGi API do not have this guarantee, these URLs must follow the existing rules for resource URLs.

Since: 1.6

---

**REMOTE_CONFIGS_SUPPORTED**

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**

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[String].

Since: 1.6
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 propertiesDictionary object passed to the BundleContext.registerService method. The value of this property must be of type String[String[], or Collection[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 propertiesDictionary object passed to the BundleContext.registerService method. The value of this property must be of type String[String[], or Collection[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 propertiesDictionary object passed to the BundleContext.registerService method. The value of this property must be of type String[String[], or Collection[String].

Since: 1.6
SERVICE_EXPORTED_INTERFACES

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", "\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 propertiesDictionary 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

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

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

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))"
"(&(sn=Jensen)(cn=Babs J*))"
"(o=univ*of*mich*)"

Since: 1.1
Version: $Id: f1924083294ca6eb8098dd50c563393f041d3345 $
See Also: “Core Specification, section 5.5, for a description of the filter string syntax.”

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals (Object obj)</td>
<td>100</td>
</tr>
<tr>
<td>int hashCode()</td>
<td>100</td>
</tr>
<tr>
<td>boolean match (Dictionary&lt;String,?&gt; dictionary)</td>
<td>100</td>
</tr>
<tr>
<td>boolean matchCase (Dictionary&lt;String,?&gt; dictionary)</td>
<td>101</td>
</tr>
<tr>
<td>boolean matches (Map&lt;String,?&gt; map)</td>
<td>101</td>
</tr>
<tr>
<td>String toString()</td>
<td>100</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 looked up in a case insensitive manner.
Parameters:
reference - The reference to the service whose properties are used in the match.

Returns:
true if the service's properties match this Filter; false otherwise.

match

boolean match(Dictionary<String,?> dictionary)

Filter using a Dictionary with case insensitive key lookup. This Filter is executed using the specified Dictionary's keys and values. The keys are looked up in a case insensitive manner.

Parameters:
dictionary - The Dictionary whose key/value pairs are used in the match.

Returns:
true if the Dictionary's 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,?> dictionary)

Filter using a Dictionary. This Filter is executed using the specified Dictionary's keys and values. The keys are looked up in a normal manner respecting case.

Parameters:
dictionary - The Dictionary whose key/value pairs are used in the match.

Returns:
true if the Dictionary's values match this filter; false otherwise.

Since: 1.3

matches

boolean matches(Map<String,?> map)

Filter using a Map. This Filter is executed using the specified Map's keys and values. The keys are looked up in a normal manner respecting case.

Parameters:
map - The Map whose key/value pairs are used in the match. Maps with null key or values are not supported. A null value is considered not present to the filter.

Returns:
true if the Map's values match this filter; false otherwise.

Since: 1.6
Class FrameworkEvent

org.osgi.framework

descends java.lang.Object

descends java.util.EventObject

descends org.osgi.framework.FrameworkEvent

All Implemented Interfaces:
  Serializable

public class FrameworkEvent
extends EventObject

A general event from the Framework.

FrameworkEvent objects are delivered to FrameworkListener 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:
$Id: 897075a0bc075c0bb89e77d113f05ca84406a073 $  
See Also:
  FrameworkListener

Immutable

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>static int</strong> ERROR</td>
</tr>
<tr>
<td>An error has occurred.</td>
</tr>
<tr>
<td><strong>static int</strong> INFO</td>
</tr>
<tr>
<td>An informational event has occurred.</td>
</tr>
<tr>
<td><strong>static int</strong> PACKAGES_REFRESHED</td>
</tr>
<tr>
<td>A PackageAdmin.refreshPackage operation has completed.</td>
</tr>
<tr>
<td><strong>static int</strong> STARTED</td>
</tr>
<tr>
<td>The Framework has started.</td>
</tr>
<tr>
<td><strong>static int</strong> STARTLEVEL_CHANGED</td>
</tr>
<tr>
<td>A StartLevel.setStartLevel operation has completed.</td>
</tr>
<tr>
<td><strong>static int</strong> STOPPED</td>
</tr>
<tr>
<td>The Framework has stopped.</td>
</tr>
<tr>
<td><strong>static int</strong> STOPPED_BOOTCLASSPATH_MODIFIED</td>
</tr>
<tr>
<td>The Framework has stopped and the boot class path has changed.</td>
</tr>
<tr>
<td><strong>static int</strong> STOPPED_UPDATE</td>
</tr>
<tr>
<td>The Framework has stopped during update.</td>
</tr>
<tr>
<td><strong>static int</strong> WAIT_TIMEDOUT</td>
</tr>
<tr>
<td>The Framework did not stop before the wait timeout expired.</td>
</tr>
<tr>
<td><strong>static int</strong> WARNING</td>
</tr>
<tr>
<td>A warning has occurred.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>FrameworkEvent(int type, Object source)</td>
</tr>
<tr>
<td>Deprecated. As of 1.2.</td>
</tr>
<tr>
<td>FrameworkEvent(int type, Bundle bundle, Throwable throwable)</td>
</tr>
<tr>
<td>Creates a Framework event regarding the specified bundle.</td>
</tr>
</tbody>
</table>
Method Summary

<table>
<thead>
<tr>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bundle getBundle()</td>
<td>Returns the bundle associated with the event.</td>
</tr>
<tr>
<td>Throwable getThrowable()</td>
<td>Returns the exception related to this event.</td>
</tr>
<tr>
<td>int getType()</td>
<td>Returns the type of framework event.</td>
</tr>
</tbody>
</table>

Field Detail

**STARTED**

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**

public static final int ERROR = 2

An error has occurred.

There was an error associated with a bundle.

**PACKAGES_REFRESHED**

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**

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

```java
public static final int WARNING = 16
```

A warning has occurred.

There was a warning associated with a bundle.

Since: 1.3

INFO

```java
public static final int INFO = 32
```

An informational event has occurred.

There was an informational event associated with a bundle.

Since: 1.3

STOPPED

```java
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

```java
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

```java
public static final int STOPPED_BOOTCLASSPATH_MODIFIED = 256
```
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.

### Method Detail

**getThrowable**

public Throwable getThrowable()

Returns the exception related to this event.
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 `FrameworkEvent` objects when the `Framework` starts and when asynchronous errors occur.

Version:
$Id: a32e7599ea09d3510759d77e824cb8d9eff67f9d $

See Also:
- `FrameworkEvent`
- `NotThreadSafe`

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>frameworkEvent(FrameworkEvent event)</code></td>
<td>107</td>
</tr>
</tbody>
</table>

Receives notification of a general `FrameworkEvent` object.

## Method Detail

### frameworkEvent

```java
void frameworkEvent(FrameworkEvent event)
```

Receives notification of a general `FrameworkEvent` object.

**Parameters:**
- `event` - The `FrameworkEvent` object.
Class FrameworkUtil

`org.osgi.framework`

`java.lang.Object`

```java
org.osgi.framework.FrameworkUtil
```

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: $Id: 06d0c5a63859e96eda5f7a7bdf8831ba3403356f $

ThreadSafe

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>createFilter(String filter)</code></td>
<td>108</td>
</tr>
<tr>
<td><code>getBundle(Class&lt;?&gt; classFromBundle)</code></td>
<td>110</td>
</tr>
<tr>
<td><code>matchDistinguishedNameChain(String matchPattern, List&lt;String&gt; dnChain)</code></td>
<td>109</td>
</tr>
</tbody>
</table>

### Method Detail

#### createFilter

```java
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`
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 ("*" \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 ) * | '-'
rdn-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 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. DrV\ezWory, 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. DrV\ezWory, 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(;' \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 ('-' \u002D) 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

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:
  $Id: adb84e3bc0b82b842e4da84542057fdf53e2ca6a $

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidSyntaxException(String msg, String filter)</td>
<td>111</td>
</tr>
<tr>
<td>InvalidSyntaxException(String msg, String filter, Throwable cause)</td>
<td>112</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throwable getCause()</td>
<td>112</td>
</tr>
<tr>
<td>String getFilter()</td>
<td>112</td>
</tr>
<tr>
<td>Throwable initCause(Throwable cause)</td>
<td>112</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.
## Class PackagePermission

### org.osgi.framework

```java
java.lang.Object
  java.security.Permission
    java.security.BasicPermission
      org.osgi.framework.PackagePermission
```

All Implemented Interfaces:
- Guard, Serializable

---

**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:

```
org.osgi.service.http
```

**PackagePermission** has three actions: **exportonly**, **import** and **export**. The export action, which is deprecated, implies the import action.

**Version:** $Id: bc511e79216fc704b5a18bd6814c7e28740a0cdd $

**ThreadSafe**

---

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String EXPORT</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>Deprecated. Since 1.5.</td>
</tr>
<tr>
<td>static String EXPORTONLY</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>The action string exportonly.</td>
</tr>
<tr>
<td>static String IMPORT</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>The action string import.</td>
</tr>
</tbody>
</table>

### Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackagePermission(String name, String actions)</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td>Creates a new PackagePermission object.</td>
</tr>
<tr>
<td>PackagePermission(String name, Bundle exportingBundle, String actions)</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>Creates a new requested PackagePermission object to be used by code that must perform checkPermission for the import action.</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(Object obj)</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Determines the equality of two PackagePermission objects.</td>
</tr>
<tr>
<td>String getActions()</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Returns the canonical string representation of the PackagePermission actions.</td>
</tr>
<tr>
<td>int hashCode()</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>Returns the hash code value for this object.</td>
</tr>
<tr>
<td>boolean implies(Permission p)</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td>Determines if the specified permission is implied by this object.</td>
</tr>
</tbody>
</table>
Field Detail

EXPORT

public static final String EXPORT = "export"

Deprecated.

The action string export. The export action implies the import action.

EXPORTONLY

public static final String EXPORTONLY = "exportonly"

The action string exportonly. The exportonly action does not imply the import action.

Since: 1.5

IMPORT

public static final String IMPORT = "import"

The action string import.

Constructor Detail

PackagePermission

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
javafx.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.
- location - The location of the exporting bundle.
- id - The bundle ID of the exporting bundle.
- name - The symbolic name of the exporting bundle.
- package.name - The name of the requested package.
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" -> x.y.z, "export"` is true
- `*, "import" -> x.y, "import"` is true
- `*, "export" -> x.y, "import"` is true
- `x.y, "export" -> x.y.z, "export"` is false

**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 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 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 class ServiceEvent
extends EventObject

An event from the Framework describing a service lifecycle change.

ServiceEvent objects are delivered to ServiceListener\$ and AllServiceListener\$ 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:
$Id: 2b9458d90004411b6ca0cb4b361bc282b04c85eb $

See Also:
ServiceListener, AllServiceListener

Immutable
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)
```

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

### getServiceReference

```java
public 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:

1. REGISTERED
2. MODIFIED
3. MODIFIED_ENDMATCH
4. UNREGISTERING

**Returns:**
Type of service lifecycle change.
Class ServiceException

org.osgi.framework

java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.lang.RuntimeException
        org.osgi.framework.ServiceException

All Implemented Interfaces:
  Serializable

public class ServiceException
    extends RuntimeException

A service exception used to indicate that a service problem occurred.

A ServletException 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: $Id: 7eb3f12f99fa32b5a28ea318cea9faece24df0b5 $

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int FACTORY_ERROR</td>
<td>122</td>
</tr>
<tr>
<td>The service factory produced an invalid service object.</td>
<td></td>
</tr>
<tr>
<td>static int FACTORY_EXCEPTION</td>
<td>122</td>
</tr>
<tr>
<td>The service factory threw an exception.</td>
<td></td>
</tr>
<tr>
<td>static int REMOTE</td>
<td>122</td>
</tr>
<tr>
<td>An error occurred invoking a remote service.</td>
<td></td>
</tr>
<tr>
<td>static int SUBCLASSED</td>
<td>122</td>
</tr>
<tr>
<td>The exception is a subclass of ServiceException.</td>
<td></td>
</tr>
<tr>
<td>static int UNREGISTERED</td>
<td>122</td>
</tr>
<tr>
<td>The service has been unregistered.</td>
<td></td>
</tr>
<tr>
<td>static int UNSPECIFIED</td>
<td>122</td>
</tr>
<tr>
<td>No exception type is unspecified.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceException(String msg)</td>
<td>123</td>
</tr>
<tr>
<td>Creates a ServiceException with the specified message.</td>
<td></td>
</tr>
<tr>
<td>ServiceException(String msg, int type)</td>
<td>123</td>
</tr>
<tr>
<td>Creates a ServletException with the specified message and type.</td>
<td></td>
</tr>
<tr>
<td>ServiceException(String msg, int type, Throwable cause)</td>
<td>123</td>
</tr>
<tr>
<td>Creates a ServletException with the specified message, type and exception cause.</td>
<td></td>
</tr>
<tr>
<td>ServiceException(String msg, Throwable cause)</td>
<td>122</td>
</tr>
<tr>
<td>Creates a ServletException with the specified message and exception cause.</td>
<td></td>
</tr>
</tbody>
</table>
Method Summary

int getType()

Returns the type for this exception or UNSPECIFIED if the type was unspecified or unknown.

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.

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 `get_type()`**

Returns the type for this exception or `UNSPECIFIED` if the type was unspecified or unknown.

**Returns:**

The type of this exception.
Interface ServiceFactory

org.osgi.framework

Type Parameters:

S - Type of Service

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.

Version:

Std: 6dad978a4354eedf8a4317b4aac3712f2315d093 $

See Also:

BundleContext.getService()

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getService</td>
<td>124</td>
</tr>
<tr>
<td>ungetService</td>
<td>125</td>
</tr>
</tbody>
</table>

Method Detail

g.getService

S getService(Bundle bundle, ServiceRegistration<?> 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.

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()
Interface ServiceListener

org.osgi.framework

All Superinterfaces:
  EventListener

All Known Subinterfaces:
  AllServiceListener

public interface ServiceListener extends EventListener

A ServiceEvent listener. 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:
$Id: d73f8e9b4babc8b53b5d1cbe7b17b732f54bb2a3 $

See Also:
ServiceEvent, ServicePermission

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void serviceChanged(ServiceEvent event)</td>
<td>126</td>
</tr>
</tbody>
</table>

Receives notification that a service has had a lifecycle change.

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.

1. The register action allows a bundle to register a service on the specified names.
2. 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:
$Id: 1f94e06913cdfe0eaa24f90925290df1215f3d5ff $
ThreadSafe

Field Summary

| static String GET | The action string get. | Page 128 |
| static String REGISTER | The action string register. | 128 |

Constructor Summary

ServicePermission(String name, String actions)
Create a new ServicePermission.

ServicePermission(ServiceReference<?> reference, String actions)
Creates a new requested ServicePermission object to be used by code that must perform checkPermission for the get action.

Method Summary

boolean equals(Object obj)
Determines the equality of two ServicePermission objects.

String getActions()
Returns the canonical string representation of the actions.

int hashCode()
Returns the hash code value for this object.

boolean implies(Permission p)
Determines if a ServicePermission object "implies" the specified permission.

PermissionCollection newPermissionCollection()
Returns a new PermissionCollection object for storing ServicePermission objects.
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:

1. 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.
2. location - The location of the bundle publishing the service.
3. id - The bundle ID of the bundle publishing the service.
4. 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.
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**

```java
public int hashCode()
```

Returns the hash code value for this object.

**Overrides:**
`hashCode` in class `BasicPermission`

**Returns:**
Hash code value for this object.
**Interface ServiceReference**

**org.osgi.framework**

**Type Parameters:**

\( S \) - Type of Service.

**All Superinterfaces:**

Comparable\( <\)Object\( >\)

---

```java
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:**

$Id: 10e6b437df20dea9d8327f8135634a48a9dfd88 $

**See Also:**

`BundleContext.getServiceReference()`, `BundleContext.getServiceReferences()`, `BundleContext.getService()`

**ThreadSafe**

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int compareTo(Object reference)</code></td>
<td>133</td>
</tr>
<tr>
<td><strong>Compares this ServiceReference with the specified ServiceReference for order.</strong></td>
<td></td>
</tr>
<tr>
<td><code>Bundle getBundle()</code></td>
<td>132</td>
</tr>
<tr>
<td><strong>Returns the bundle that registered the service referenced by this ServiceReference object.</strong></td>
<td></td>
</tr>
<tr>
<td><code>Object getProperty(String key)</code></td>
<td>132</td>
</tr>
<tr>
<td><strong>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.</strong></td>
<td></td>
</tr>
<tr>
<td><code>String[] getPropertyKeys()</code></td>
<td>132</td>
</tr>
<tr>
<td><strong>Returns an array of the keys in the properties Dictionary object of the service referenced by this ServiceReference object.</strong></td>
<td></td>
</tr>
<tr>
<td><code>Bundle[] getUsingBundles()</code></td>
<td>132</td>
</tr>
<tr>
<td><strong>Returns the bundles that are using the service referenced by this ServiceReference object.</strong></td>
<td></td>
</tr>
<tr>
<td><code>boolean isAssignableTo(Bundle bundle, String className)</code></td>
<td>133</td>
</tr>
<tr>
<td><strong>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.</strong></td>
<td></td>
</tr>
</tbody>
</table>
Method Detail

getProperty

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

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

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

Returns:
An array of using bundles.
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:

- Get the package name from the specified class name.
- 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.
- 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.
Interface ServiceRegistration

org.osgi.framework

Type Parameters:

S - Type of Service.

class 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:

\$Id: 6487c568cab6629deb05feaf86d83caade9ac94e \$

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>135</td>
</tr>
<tr>
<td>void setProperties(Dictionary&lt;String,?&gt; properties)</td>
<td>135</td>
</tr>
<tr>
<td>void unregister()</td>
<td>136</td>
</tr>
</tbody>
</table>

Method Detail

getReference

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,?> 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:
The service’s properties are replaced with the provided properties.

- 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:

- The service is removed from the Framework service registry so that it can no longer be obtained.
- 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.
- 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

```java
public interface SynchronousBundleListener
    extends BundleListener
```

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.

For BundleEvent types **STARTED** and **LAZY_ACTIVATION**, the Framework must not hold the referenced bundle's "state change" lock when the BundleEvent is delivered to a SynchronousBundleListener. For the other BundleEvent types, the Framework must hold the referenced bundle's "state change" lock when the BundleEvent is delivered to a SynchronousBundleListener. A SynchronousBundleListener cannot directly call life cycle methods on the referenced bundle when the Framework is holding the referenced bundle's "state change" lock.

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, SynchronousBundleListeners are synchronously called during bundle lifecycle processing. The bundle lifecycle processing will not proceed until all SynchronousBundleListeners 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:** $Id: b22484f48ebdcb2141da9bba9eb65f5c40e0f520 $  
**See Also:** BundleEvent  
**ThreadSafe**

#### Methods inherited from interface org.osgi.framework.BundleListener

- `bundleChanged`

---

*Copyright © IBM Corporation 2010 All Rights Reserved*
Class Version

org.osgi.framework

java.lang.Object

org.osgi.framework.Version

All Implemented Interfaces:

Comparable<Version>

public class Version
extends Object
implements Comparable<Version>

Version identifier for bundles and packages.

Version identifiers have four components.

- Qualifier. A text string. See Version(String) for the format of the qualifier string.

Version objects are immutable.

Since:

1.3

Version:

$Id: 6b36c5c1ac6ff508fca81eeec6a25c20d1f00ce $

Immutable

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Version emptyVersion</td>
<td>139</td>
</tr>
<tr>
<td>The empty version &quot;0.0.0&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version(int major, int minor, int micro)</td>
<td>139</td>
</tr>
<tr>
<td>Creates a version identifier from the specified numerical components.</td>
<td></td>
</tr>
<tr>
<td>Version(int major, int minor, int micro, String qualifier)</td>
<td>139</td>
</tr>
<tr>
<td>Creates a version identifier from the specified components.</td>
<td></td>
</tr>
<tr>
<td>Version(String version)</td>
<td>140</td>
</tr>
<tr>
<td>Created a version identifier from the specified string.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int compareTo(Version other)</td>
<td>142</td>
</tr>
<tr>
<td>Compares this Version object to another object.</td>
<td></td>
</tr>
<tr>
<td>boolean equals(Object object)</td>
<td>141</td>
</tr>
<tr>
<td>Compares this Version object to another object.</td>
<td></td>
</tr>
<tr>
<td>int getMajor()</td>
<td>140</td>
</tr>
<tr>
<td>Returns the major component of this version identifier.</td>
<td></td>
</tr>
<tr>
<td>int getMicro()</td>
<td>141</td>
</tr>
<tr>
<td>Returns the micro component of this version identifier.</td>
<td></td>
</tr>
<tr>
<td>int getMinor()</td>
<td>140</td>
</tr>
<tr>
<td>Returns the minor component of this version identifier.</td>
<td></td>
</tr>
</tbody>
</table>
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.
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

public int getMicro()

Returns the micro component of this version identifier.

Returns:
The micro component.

getQualifier

public String getQualifier()

Returns the qualifier component of this version identifier.

Returns:
The qualifier component.

toString

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

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

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
Parameters:
object - The Version object to be compared.

Returns:
true if object is a Version and is equal to this object; false otherwise.

compareTo

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.
Package org.osgi.framework.startlevel

Framework Start Level Package Version 1.0.

See:  
Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleStartLevel</td>
<td>144</td>
</tr>
<tr>
<td>Query and modify the start level information for a bundle.</td>
<td></td>
</tr>
<tr>
<td>FrameworkStartLevel</td>
<td>146</td>
</tr>
<tr>
<td>Query and modify the start level information for the framework.</td>
<td></td>
</tr>
</tbody>
</table>

Package org.osgi.framework.startlevel Description

Framework Start Level Package Version 1.0.

The Framework Start Level package allows management agents to manage a start level assigned to each bundle and the active start level of the Framework. This package is a replacement for the now deprecated org.osgi.service.startlevel package.

A start level is defined to be a state of execution in which the Framework exists. Start level 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. For example, 2 is a higher start level than 1.

AdminPermission is required to modify start level information.

Start Level support in the Framework includes the ability to modify the active start level of the Framework and to assign a specific start level to a bundle. The beginning start level of a Framework is specified via the Constants.FRAMEWORK_BEGINNING_STARTLEVEL framework property when configuring a framework.

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(int) 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 Framework Start Level package can be used by management bundles to alter the active start level of the framework.

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.startlevel; version="[1.0,2.0)"

Copyright © IBM Corporation 2010  All Rights Reserved
Interface BundleStartLevel

org.osgi.framework.startlevel
All Superinterfaces:
BundleReference

public interface BundleStartLevel
extends BundleReference

Query and modify the start level information for a bundle. The start level object for a bundle can be obtained by calling BundleReference.adapt(BundleStartLevel.class) on the bundle.

The bundle associated with this BundleStartLevel object can be obtained by calling BundleReference.getBundle().

Version:
$Id: cb32d3a867c1844e1c06913bfb4cac67b71cd070 $

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getStartLevel()</td>
<td>144</td>
</tr>
<tr>
<td>Return the assigned start level value for the bundle.</td>
<td></td>
</tr>
<tr>
<td>boolean isActivationPolicyUsed()</td>
<td>145</td>
</tr>
<tr>
<td>Returns whether the bundle's autostart setting indicates that the activation policy declared in the bundle manifest must be used.</td>
<td></td>
</tr>
<tr>
<td>boolean isPersistentlyStarted()</td>
<td>145</td>
</tr>
<tr>
<td>Returns whether the bundle's autostart setting indicates it must be started.</td>
<td></td>
</tr>
<tr>
<td>void setStartLevel(int startlevel)</td>
<td>144</td>
</tr>
<tr>
<td>Assign a start level value to the bundle.</td>
<td></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: 
IllegalArgumentException - 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

```java
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

```java
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 FrameworkStartLevel
extends BundleReference

Query and modify the start level information for the framework. The start level object for the framework can be obtained by calling bundle.adapt(FrameworkStartLevel.class) on the system bundle. Only the system bundle can be adapted to a FrameworkStartLevel object.

The system bundle associated with this FrameworkStartLevel object can be obtained by calling BundleReference.getBundle().

Version: $Id: 648f7e4ea924a3a34cd7e8d2f092f88cfd552ae2 $
ThreadSafe

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getInitialBundleStartLevel()</td>
<td>Return the initial start level value that is assigned to a Bundle when it is first installed.</td>
<td>147</td>
</tr>
<tr>
<td>int getStartLevel()</td>
<td>Return the active start level value of the Framework.</td>
<td>146</td>
</tr>
<tr>
<td>void setInitialBundleStartLevel(int startlevel)</td>
<td>Set the initial start level value that is assigned to a Bundle when it is first installed.</td>
<td>147</td>
</tr>
<tr>
<td>void setStartLevel(int startlevel, FrameworkListener... listeners)</td>
<td>Modify the active start level of the Framework and notify when complete.</td>
<td>146</td>
</tr>
</tbody>
</table>

### Methods inherited from interface org.osgi.framework.BundleReference
getBundle

### Method Detail

#### getStartLevel

**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

**void setStartLevel(int startlevel, FrameworkListener... listeners)**

Modify the active start level of the Framework and notify when complete.

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. The specified FrameworkListener s are notified, 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.

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:

- Change the active start level to the intermediate start level value.
- 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 BundleStartLevel.isActivationPolicyUsed() 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 returns.

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.

---

getInitialBundleStartLevel

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

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.
Package org.osgi.framework.wiring

Framework Wiring Package Version 1.0.

See: Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleRevision</td>
<td>150</td>
</tr>
<tr>
<td>BundleWiring</td>
<td>152</td>
</tr>
<tr>
<td>BundleWirings</td>
<td>157</td>
</tr>
<tr>
<td>Capability</td>
<td>158</td>
</tr>
<tr>
<td>FrameworkWiring</td>
<td>161</td>
</tr>
</tbody>
</table>

BundleRevision

Bundle Revision.

BundleWiring

A wiring for a bundle.

BundleWirings

The in use bundle wirings for a bundle.

Capability

A capability that has been provided from a bundle Wiring.

FrameworkWiring

Query and modify wiring information for the framework.

Package org.osgi.framework.wiring Description

Framework Wiring 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.wiring; version="[1.0,2.0)"

Copyright © IBM Corporation 2010 All Rights Reserved
public interface BundleRevision
extends BundleReference

Bundle Revision. Since a bundle update can change the entries in a bundle, different bundle wirings for the same bundle can be associated with different bundle revisions.

The current bundle revision for a bundle can be obtained by calling bundle.adapt(BundleRevision.class).

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_FRAGMENT</td>
<td>150</td>
</tr>
</tbody>
</table>

Bundle revision type indicating the bundle revision is a fragment.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getSymbolicName()</td>
<td>150</td>
</tr>
<tr>
<td>int getTypes()</td>
<td>151</td>
</tr>
<tr>
<td>Version getVersion()</td>
<td>151</td>
</tr>
</tbody>
</table>

Returns the symbolic name for this bundle revision.

Returns the special types of this bundle revision.

Returns the version for this bundle revision.

### Methods inherited from interface org.osgi.framework.BundleReference

getBundle

### Field Detail

**TYPE_FRAGMENT**

public static final int TYPE_FRAGMENT = 1

Bundle revision type indicating the bundle revision is a fragment.

See Also:
getTypes()
Interface ExportsedPackage

See Also:
Bundle.getSymboticName()

getVersion

Version getVersion()

Returns the version for this bundle revision.

Returns: The version for this bundle revision, or Version.emptyVersion if this bundle revision has no version information.

See Also:
Bundle.getVersion()

getypes

int getTypes()

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

1. TYPE_FRAGMENT

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

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

Returns: The special types of this bundle revision. The type values are ORed together.
public interface BundleWiring extends BundleReference

A wiring for a bundle. Each time a bundle is resolved, a new bundle wiring for the bundle is created. A bundle wiring 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 bundles with non-current, in use bundle wirings are considered removal pending. A bundle wiring is in use if it is the current wiring or if some other in use 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 has a 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 bundle wirings for a bundle can be obtained by calling bundle.adapt(BundleWirings.class).getWirings(). For non-fragment bundles, the first item in the returned list is the current bundle wiring.

The current bundle wiring for a non-fragment bundle can be obtained by calling bundle.adapt(BundleWiring.class). A fragment bundle does not itself have bundle wirings. So calling bundle.adapt(BundleWiring.class) on a fragment must return null.

Version: $Id: 6c7087fe10720d62f87aefba32a38bebec6f50fe $
ThreadSafe

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int FINDENTRIES_RECURSE</td>
<td>153</td>
</tr>
<tr>
<td>The find entries operation must recurse into subdirectories.</td>
<td></td>
</tr>
<tr>
<td>int LISTRESOURCES_LOCAL</td>
<td>153</td>
</tr>
<tr>
<td>The list resource names operation must limit the result to the names of matching resources contained in this bundle wiring's bundle revision and its attached fragment revisions.</td>
<td></td>
</tr>
<tr>
<td>int LISTRESOURCES_RECURSE</td>
<td>153</td>
</tr>
<tr>
<td>The list resource names operation must recurse into subdirectories.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List&lt;URL&gt; findEntries(String path, String filePattern, int options)</td>
<td>155</td>
</tr>
<tr>
<td>Returns entries in this bundle wiring's bundle revision and its attached fragment revisions.</td>
<td></td>
</tr>
<tr>
<td>BundleRevision getBundleRevision()</td>
<td>155</td>
</tr>
<tr>
<td>Returns the bundle revision for the bundle in this bundle wiring.</td>
<td></td>
</tr>
<tr>
<td>ClassLoader getClassLoader()</td>
<td>155</td>
</tr>
<tr>
<td>Returns the class loader for this bundle wiring.</td>
<td></td>
</tr>
<tr>
<td>List&lt;BundleRevision&gt; getFragmentRevisions()</td>
<td>155</td>
</tr>
<tr>
<td>Returns the bundle revisions for all attached fragments of this bundle wiring.</td>
<td></td>
</tr>
<tr>
<td>List&lt;Capability&gt; getProvidedCapabilities(String capabilityNamespace)</td>
<td>154</td>
</tr>
<tr>
<td>Returns the capabilities provided by this bundle wiring.</td>
<td></td>
</tr>
<tr>
<td>List&lt;Capability&gt; getRequiredCapabilities(String capabilityNamespace)</td>
<td>154</td>
</tr>
<tr>
<td>Returns the required capabilities used by this bundle wiring.</td>
<td></td>
</tr>
<tr>
<td>boolean isCurrent()</td>
<td>154</td>
</tr>
<tr>
<td>Returns true if this bundle wiring is the current bundle wiring.</td>
<td></td>
</tr>
</tbody>
</table>
boolean **isInUse**()  
Returns true if this bundle wiring is in use.

List<String> **listResources**(String path, String filePattern, int options)  
Returns the names of resources visible to this bundle wiring's class loader.

Methods inherited from interface org.osgi.framework.BundleReference

**getPage**

Field Detail

**FINDENTIRES_RECURSE**

public static final int FINDENTIRES_RECURSE = 1

The find entries operation must recurse into subdirectories.

This bit may be set when calling findEntries(String, String, int) to specify the result must include the matching entries from the specified path and its subdirectories. If this bit is not set, then the result must only include matching entries from the specified path.

See Also:  
findEntries(String, String, int)

**LISTRESOURCES_RECURSE**

public static final int LISTRESOURCES_RECURSE = 1

The list resource names operation must recurse into subdirectories.

This bit may be set when calling listResources(String, String, int) to specify the result must include the names of matching resources from the specified path and its subdirectories. If this bit is not set, then the result must only include names of matching resources from the specified path.

See Also:  
listResources(String, String, int)

**LISTRESOURCES_LOCAL**

public static final int LISTRESOURCES_LOCAL = 2

The list resource names operation must limit the result to the names of matching resources contained in this bundle wiring's bundle revision and its attached fragment revisions.

This bit may be set when calling listResources(String, String, int) to specify the result must only include the names of matching resources contained in this bundle wiring's bundle revision and its attached fragment revisions. If this bit is not set, then the result must include the names of matching resources reachable from this bundle wiring's class loader which may include the names of matching resources contained in imported packages and required bundles.

See Also:  
listResources(String, String, int)
Method Detail

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 bundles with non-current, in use bundle wirings 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 in use 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.

getProvidedCapabilities

List<Capability> getProvidedCapabilities(String capabilityNamespace)

Returns the capabilities provided by this bundle wiring.

Parameters:

capabilityNamespace - The name space of the provided capabilities to return or null to return the provided capabilities from all name spaces.

Returns: A list containing a snapshot of the Capabilitys, or an empty list if this bundle wiring provides no capabilities in the specified name space. If this bundle wiring is not in use, null will be returned. The list contains the provided capabilities in the order they are specified in the manifest.

getRequiredCapabilities

List<Capability> getRequiredCapabilities(String capabilityNamespace)

Returns the required capabilities used by this bundle wiring.

The result of this method can change if this bundle wiring requires additional capabilities.

Parameters:

capabilityNamespace - The name space of the required capabilities to return or null to return the required capabilities from all name spaces.

Returns: A list containing a snapshot of the Capabilitys used by this bundle wiring, or an empty list if this bundle wiring requires no capabilities in the specified name space. If this bundle wiring is not in use, null will be returned. The list contains the required capabilities in the order they are specified in the manifest.
getBundleRevision

BundleRevision getBundleRevision()

Returns the bundle revision for the bundle in this bundle wiring. Since a bundle update can change the entries in a bundle, different bundle wirings for the same bundle can have different bundle revisions.

The bundle object referenced by the returned BundleRevision may return different information than the returned BundleRevision since the returned BundleRevision may refer to an older revision of the bundle.

Returns:
The bundle revision for this bundle wiring.

getFragmentRevisions

List<BundleRevision> getFragmentRevisions()

Returns the bundle revisions for all attached fragments of this bundle wiring. Since a bundle update can change the entries in a fragment, different bundle wirings for the same bundle can have different bundle revisions.

The bundle revisions in the list are ordered in fragment attachment order such that the first revision in the list is the first attached fragment and the last revision in the list is the last attached fragment.

Returns:
A list containing a snapshot of the BundleRevisions 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.

classLoader

ClassLoader getClassLoader()

Returns the class loader for this bundle wiring. Since a bundle refresh creates a new bundle wiring for a bundle, different bundle wirings for the same bundle will have different class loaders.

Returns:
The class loader for this bundle wiring. If this bundle wiring is not in use, null will be returned.

Throws:
SecurityException - If the caller does not have the appropriate RuntimePermission("getClassLoader"), and the Java Runtime Environment supports permissions.

findEntries

List<URL> findEntries(String path, String filePattern, int options)

Returns entries in this bundle wiring's bundle revision and its attached fragment revisions. This bundle wiring's class loader is not used to search for entries. Only the contents of this bundle wiring's bundle revision and its attached fragment revisions are searched for the specified entries.

This method takes into account that the "contents" of this bundle wiring can have attached fragments. This "bundle space" is not a namespace with unique members; the same entry name can be present multiple times. This method therefore returns a list 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.
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 wiring and may begin with "/". A path value of "/" indicates the root of this bundle wiring.
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.
options - The options for listing resource names. See FINDENTRIES_RECURSE. The method must ignore unrecognized options.

Returns:
An unmodifiable list of URL objects for each matching entry, or an empty list if no matching entry could not be found or if the caller does not have the appropriate AdminPermission[bundle,RESOURCE] and the Java Runtime Environment supports permissions. The list is ordered such that entries from the bundle_revision are returned first followed by the entries from attached_fragment_revisions in attachment order. If this bundle wiring is not in use, null will be returned.

See Also:
Bundle.findEntries(String, String, boolean)

### listResources

List<String> listResources(String path,
String filePattern,
int options)

Returns the names of resources visible to this bundle wiring's class loader. The returned names can be used to access the resources via this bundle wiring's class loader.

Parameters:
path - The path name in which to look. The path is always relative to the root of this bundle wiring's class loader and may begin with "/". A path value of "/" indicates the root of this bundle wiring's class loader.
filePattern - The file name pattern for selecting resource names in the specified path. The pattern is only matched against the last element of the resource path. If the resource 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.
options - The options for listing resource names. See LISTRESOURCES_LOCAL and LISTRESOURCES_RECURSE. The method must ignore unrecognized options.

Returns:
An unmodifiable list of resource names for each matching resource, or an empty list if no matching resource could not be found or if the caller does not have the appropriate AdminPermission[bundle,RESOURCE] and the Java Runtime Environment supports permissions. The list is ordered such that resource names from this bundle are returned in the order they are visible in this bundle wiring's class loader. If this bundle wiring is not in use, null will be returned.
public interface BundleWirings
    extends BundleReference

The in use bundle wirings for a bundle. Each time a bundle is resolved, a new bundle wiring of the bundle is created. A bundle wiring consists of a bundle and it attached fragments and represents the dependencies with other bundle wirings.

The in use bundle wirings for a bundle can be obtained by calling bundle.adapt(BundleWirings.class).getWirings().

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getWirings()</td>
<td>Return the in use wirings for the referenced bundle.</td>
</tr>
</tbody>
</table>

Methods inherited from interface org.osgi.framework.BundleReference

getBundle

Method Detail

getWirings

List<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 in use bundle wirings. The list is ordered in reverse chronological order such that the first bundle wiring is the current bundle wiring and last wiring is the oldest in use bundle wiring.

If the referenced bundle is a fragment bundle, then the result is a list of in use bundle wirings to which the referenced fragment bundle is attached. The ordering of the list is unspecified. If the fragment bundle is not attached to any bundle wiring, then the returned list will be empty.

The list must only contain in use bundle wirings. Generally the list will have at least one bundle wiring for the bundle: the current bundle wiring. However, for an uninstalled bundle with no in use bundle wirings or a newly installed bundle which has not been resolved, the list will be empty.

Returns:
A list containing a snapshot of the BundleWirings for the referenced bundle.
public interface Capability

A capability that has been provided from a bundle wiring. This capability may or may not be required by any bundle wiring.

The framework defines capabilities for packages and bundles.

Version:
$Id: 5173f1e89bf654b3887f1ee32cfc6257b424dd4e $

ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String BUNDLE_CAPABILITY</td>
<td>159</td>
</tr>
<tr>
<td>Capability name space for bundle capabilities.</td>
<td></td>
</tr>
<tr>
<td>String PACKAGE_CAPABILITY</td>
<td>158</td>
</tr>
<tr>
<td>Capability name space for package capabilities.</td>
<td></td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map&lt;String ,Object&gt; getAttributes()</td>
<td>159</td>
</tr>
<tr>
<td>Returns the attributes of this capability.</td>
<td></td>
</tr>
<tr>
<td>Map&lt;String ,String&gt; getDirectives()</td>
<td>159</td>
</tr>
<tr>
<td>Returns the directives of this capability.</td>
<td></td>
</tr>
<tr>
<td>String getNamespace()</td>
<td>159</td>
</tr>
<tr>
<td>Returns the name space of this capability.</td>
<td></td>
</tr>
<tr>
<td>BundleWiring getProviderWiring()</td>
<td>159</td>
</tr>
<tr>
<td>Returns the bundle wiring providing this capability.</td>
<td></td>
</tr>
<tr>
<td>Collection &lt;BundleWiring&gt; getRequirerWirings()</td>
<td>160</td>
</tr>
<tr>
<td>Returns the bundle wirings that require this capability.</td>
<td></td>
</tr>
</tbody>
</table>

Field Detail

PACKAGE_CAPABILITY

public static final String PACKAGE_CAPABILITY = "osgi.package"

Capability name space for package capabilities. The name of the package is stored in the capability attribute of the same name as this name space. The other directives and attributes of the package, from the Export-Package manifest header, can be found in the capability's directives and attributes. The version capability attribute must contain the Version of the package if one is specified.

The package capabilities provided by the system bundle, that is the bundle with id zero, must include the package specified by the Constants.FRAMEWORK_SYSTEMPACKAGES and Constants.FRAMEWORK_SYSTEMPACKAGES_EXTRA framework properties as well as any other package exported by the framework implementation.

A bundle wiring provides zero or more package capabilities (that is, exported packages) and requires zero or more package capabilities (that is, imported packages). The number of package capabilities required by a bundle wiring may change as the bundle wiring may dynamically import additional packages.
public static final String BUNDLE_CAPABILITY = "osgi.bundle"

Capability name space for bundle capabilities. The bundle symbolic name of the bundle is stored in the capability attribute of the same name as this name space. The other directives and attributes of the bundle, from the Bundle-SymbolicName manifest header, can be found in the capability's directives and attributes. The bundle-version capability attribute must contain the Version of the bundle, from the Bundle-Version manifest header.

A bundle wiring provides exactly one† bundle capability (that is, the bundle can be required by another bundle) and requires zero or more bundle capabilities (that is, requires other bundles).

† A bundle with no bundle symbolic name (that is, a bundle with Bundle-ManifestVersion< 2) must not provide a bundle capability.

Method Detail

getNamespace

String getNamespace()

Returns the name space of this capability.

Returns:
  The name space of this capability.

getDirectives

Map<String,String> getDirectives()

Returns the directives of this capability.

Returns:
  A map of directive names to directive values for this capability, or an empty map if this capability has no directives.

getAttributes

Map<String,Object> getAttributes()

Returns the attributes of this capability.

Returns:
  A map of attribute names to attribute values for this capability, or an empty map if this capability has no attributes.

getProviderWiring

BundleWiring getProviderWiring()

Returns the bundle wiring providing this capability.

Returns:
  The bundle wiring providing this capability. If the bundle wiring providing this capability is not in use, null will be returned.
getRequirerWirings

Collection<BundleWiring> getRequirerWirings()

Returns the bundle wirings that require this capability.

The result of this method can change if this capability becomes required by additional bundle wirings.

**Returns:**

A collection containing a snapshot of the bundle wirings currently requiring this capability, or an empty collection if no bundle wirings require this capability. If the bundle wiring providing this capability is not in use, null will be returned.
### Interface FrameworkWiring

**Package** org.osgi.service.startlevel  

**org.osgi.framework.wiring**  
All Superinterfaces:  
BundleReference

```java
public interface FrameworkWiring extends BundleReference
```

Query and modify wiring information for the framework. The framework wiring object for the framework can be obtained by calling `bundle.adapt(FrameworkWiring.class)` on the system bundle. Only the system bundle can be adapted to a FrameworkWiring object.

The system bundle associated with this FrameworkWiring object can be obtained by calling `BundleReference.getBundle()`.

**Version:** $Id: 820cd38ec470b064999d6eff0c2bb4a214bd8d9b $

**ThreadSafe**

#### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDependencyClosure(Collection&lt;Bundle&gt; bundles)</code></td>
<td>163</td>
</tr>
<tr>
<td><code>getRemovalPendingBundles()</code></td>
<td>162</td>
</tr>
<tr>
<td><code>refreshBundles(Collection&lt;Bundle&gt; bundles, FrameworkListener... listeners)</code></td>
<td>161</td>
</tr>
<tr>
<td><code>resolveBundles(Collection&lt;Bundle&gt; bundles)</code></td>
<td>162</td>
</tr>
</tbody>
</table>

#### Method Detail

**refreshBundles**

```java
void refreshBundles(Collection<Bundle> bundles,
                    FrameworkListener... listeners)
```

Refreshes the specified bundles. This forces the update (replacement) or removal of packages exported by the specified bundles.

The technique by which the framework refreshes bundles may vary among different framework implementations. A 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 `removal_pending` bundles.
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 bundles satisfying the dependencies have been installed in the Framework.

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.

When this process completes after the bundles are refreshed, the Framework will fire a framework event of type FrameworkEvent.PACKAGES_REFRESHED to announce it has completed the bundle refresh. The specified FrameworkListeners are notified in the order specified. Each specified FrameworkListener will be called with a Framework event of type FrameworkEvent.PACKAGES_REFRESHED.

### Parameters:
- **bundles** - The bundles to be refreshed, or null to refresh the removal pending bundles.
- **listeners** - Zero or more listeners to be notified when the bundle refresh 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 Bundle(s) were not created by the same framework instance associated with this FrameworkWiring.
- SecurityException - If the caller does not have AdminPermission[System Bundle, RESOLVE] and the Java runtime environment supports permissions.

---

**resolveBundles**

```java
boolean resolveBundles(Collection<Bundle> bundles)
```

Resolves 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 no bundles are 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; false otherwise.

### Throws:
- IllegalArgumentException - If the specified Bundle(s) were not created by the same framework instance associated with this FrameworkWiring.
- SecurityException - If the caller does not have AdminPermission[System Bundle, RESOLVE] and the Java runtime environment supports permissions.

---

**getRemovalPendingBundles**

```java
Collection<Bundle> getRemovalPendingBundles()
```

Returns the bundles that have non-current, in use bundle wirings. 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 which have non-current, in use BundleWiring, or an empty collection if there are no such bundles.
getDependencyClosure

Collection<Bundle> getDependencyClosure(Collection<Bundle> bundles)

Returns 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 UNINSTALLED bundles that are removal pending.

Parameters:
bundles - The initial bundles for which to generate the dependency closure.

Returns:
A collection containing a snapshot of the dependency closure of the specified bundles, or an empty
collection if there were no specified bundles.

Throws:
IllegalArgumentException - If the specified Bundles were not created by the same framework
instance associated with this FrameworkWiring.
Package org.osgi.util.tracker

Tracker Package Version 1.5.

See: Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleTrackerCustomizer</td>
<td>170</td>
</tr>
<tr>
<td>ServiceTrackerCustomizer</td>
<td>180</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleTracker</td>
<td>165</td>
</tr>
<tr>
<td>ServiceTracker</td>
<td>172</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.

Example import for consumers using the API in this package:

Import-Package: org.osgi.util.tracker; version="[1.5,2.0)"
Class BundleTracker

type.org.osgi.util.tracker
java.lang.Object

protected BundleContext context

The Bundle Context used by this BundleTracker.

public class BundleTracker
extends Object
implements BundleTrackerCustomizer<T>

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: $Id: 4f1410cfefaf6b9fb6060090d605324e5d9254ac $
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()
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

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

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

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

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

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.

ggetObject

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.

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**

```java
public int size()
```

Return the number of bundles being tracked by this `BundleTracker`.

**Returns:**

The number of bundles being tracked.

---

**getTrackingCount**

```java
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**

```java
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
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: $Id: 0e80f2555530b217faef57726a5938f0087a45c5 $

ThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>addingBundle(Bundle bundle, BundleEvent event)</td>
<td>A bundle is being added to the BundleTracker.</td>
</tr>
<tr>
<td>void modifiedBundle(Bundle bundle, BundleEvent event, T object)</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>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.

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

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 ServiceEvents 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: $Id: b375e1b70754866966b42fc55546375f0546b0d58 $
ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>context</td>
<td>BundleContext</td>
<td>The Bundle Context used by this ServiceTracker.</td>
</tr>
<tr>
<td>filter</td>
<td>Filter</td>
<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>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceTracker(BundleContext context, Class&lt;S&gt; clazz, ServiceTrackerCustomizer&lt;S,T&gt; customizer)</td>
<td>Create a ServiceTracker on the specified class.</td>
</tr>
<tr>
<td>ServiceTracker(BundleContext context, String clazz, ServiceTrackerCustomizer&lt;S,T&gt; customizer)</td>
<td>Create a ServiceTracker on the specified class name.</td>
</tr>
<tr>
<td>ServiceTracker(BundleContext context, Filter filter, ServiceTrackerCustomizer&lt;S,T&gt; customizer)</td>
<td>Create a ServiceTracker on the specified Filter object.</td>
</tr>
<tr>
<td>ServiceTracker(BundleContext context, ServiceReference&lt;S&gt; reference, ServiceTrackerCustomizer&lt;S,T&gt; customizer)</td>
<td>Create a ServiceTracker on the specified ServiceReference.</td>
</tr>
</tbody>
</table>
## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addingService(ServiceReference&lt;S&gt; reference)</td>
<td>Default implementation of the ServiceTrackerCustomizer.addingService method.</td>
</tr>
<tr>
<td>close()</td>
<td>Close this ServiceTracker.</td>
</tr>
<tr>
<td>getService()</td>
<td>Returns a service object for one of the services being tracked by this ServiceTracker.</td>
</tr>
<tr>
<td>getService(ServiceReference&lt;S&gt; reference)</td>
<td>Returns the service object for the specified ServiceReference if the specified referenced service is being tracked by this ServiceTracker.</td>
</tr>
<tr>
<td>getServiceReference()</td>
<td>Returns a ServiceReference for one of the services being tracked by this ServiceTracker.</td>
</tr>
<tr>
<td>getServiceReferences()</td>
<td>Return an array of ServiceReferences for all services being tracked by this ServiceTracker.</td>
</tr>
<tr>
<td>getServices()</td>
<td>Return an array of service objects for all services being tracked by this ServiceTracker.</td>
</tr>
<tr>
<td>getTracked()</td>
<td>Return a SortedMap of the ServiceReferences and service objects for all services being tracked by this ServiceTracker.</td>
</tr>
<tr>
<td>getTrackingCount()</td>
<td>Returns the tracking count for this ServiceTracker.</td>
</tr>
<tr>
<td>modifiedService(ServiceReference&lt;S&gt; reference, T service)</td>
<td>Default implementation of the ServiceTrackerCustomizer.modifiedService method.</td>
</tr>
<tr>
<td>open()</td>
<td>Open this ServiceTracker and begin tracking services.</td>
</tr>
<tr>
<td>open(boolean trackAllServices)</td>
<td>Open this ServiceTracker and begin tracking services.</td>
</tr>
<tr>
<td>remove(ServiceReference&lt;S&gt; reference)</td>
<td>Remove a service from this ServiceTracker.</td>
</tr>
<tr>
<td>removedService(ServiceReference&lt;S&gt; reference, T service)</td>
<td>Default implementation of the ServiceTrackerCustomizer.removedService method.</td>
</tr>
<tr>
<td>size()</td>
<td>Return the number of services being tracked by this ServiceTracker.</td>
</tr>
<tr>
<td>waitForService(long timeout)</td>
<td>Wait for at least one service to be tracked by this ServiceTracker.</td>
</tr>
</tbody>
</table>

## Field Detail

### context

**protected final BundleContext context**

The Bundle Context used by this ServiceTracker.

### filter

**protected final Filter filter**

The Filter used by this ServiceTracker which specifies the search criteria for the services to track.
Class BundleTracker

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.
class - 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.

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
Class BundleTracker

ServiceTracker

public ServiceTracker(BundleContext context, Class<? extends 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: IllegalAccessException - 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: IllegalAccessException - 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**

```java
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**

```java
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**

```java
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**

```java
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**

```java
public ServiceReference<S>[] getServiceReferences()
```

Return an array of `ServiceReference`s for all services being tracked by this `ServiceTracker`.

**Returns:**
- Array of `ServiceReference`s or `null` if no services are being tracked.

**getServiceReference**

```java
public ServiceReference<S> 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.
Class BundleTracker

Since:

1.1

getService

public T getService(ServiceReference<S> 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

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

public T 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

`org.osgi.util.tracker`

Type Parameters:

- `S` - The type of the service being tracked.
- `T` - The type of the tracked object.

All Known Implementing Classes:

`ServiceTracker`

---

```java
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.getProperties`) 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:

`$Id: c654a96336cee74762b8f54c8cef8d5774f8b4d $`

ThreadSafe

---

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>T addingService(ServiceReference&lt;S&gt; reference)</code></td>
<td>180</td>
</tr>
<tr>
<td><code>void modifiedService(ServiceReference&lt;S&gt; reference, T service)</code></td>
<td>181</td>
</tr>
<tr>
<td><code>void removedService(ServiceReference&lt;S&gt; reference, T service)</code></td>
<td>181</td>
</tr>
</tbody>
</table>

---

### Method Detail

#### addingService

```java
T 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`.

**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).

### 7.5 Reverted changed to org.osgi.service.packageadmin and org.osgi.service.startlevel

The org.osgi.service.packageadmin and org.osgi.service.startlevel packages were reverted to their OSGi Core 4.2 level. They are being deprecated and the org.osgi.framework.wiring and org.osgi.framework.startlevel packages proposed by this RFC are replacing them.

### 7.6 Remove specific methods for exported packages and required bundles

The change to add support for RFC 154 generic capabilities led to API changes to model exported packages and required bundles as capabilities. Thus the methods which directly return package and bundle wire information were removed.

### 7.7 Bundle specific entry methods could be added later to BundleRevision

Draft 7 removed getEntry, getEntryPaths and getHeaders from BundleWiring. BundleWiring applies to the resolve state of a bundle and its attached fragments. Having methods which only operate on a specific bundle or fragment on BundleWiring is inappropriate. If necessary we can consider adding the getEntryPaths and getEntry method to BundleRevision in the future.

### 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
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 ................................................................................................................. 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
  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
7 Security Considerations............................................................................................................................................. 10

8 Document Support......................................................................................................................................................... 10
  8.1 References......................................................................................................................................................... 10
  8.2 Author’s Address.................................................................................................................................................. 11
  8.3 Acronyms and Abbreviations............................................................................................................................... 11
  8.4 End of Document.................................................................................................................................................. 11

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></td>
</tr>
<tr>
<td>Southampton F2F feedback</td>
<td>Mar. 1st, 2010</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>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>
<tr>
<td>Mountview F2F feedback</td>
<td>May 13th, 2010</td>
<td>Rolled back the prior changes, added language to indicate the resolve-time scope of the solution, and added initial requirements.</td>
</tr>
<tr>
<td>London F2F feedback</td>
<td>June 1st, 2010</td>
<td>Add “effective time” directive and adjusted scope discussion accordingly</td>
</tr>
<tr>
<td>RFC 151</td>
<td>July 23rd, 2010</td>
<td>Added a simple comment to tie this RFC to RFC 151 for purposes of an API for reflecting on wiring.</td>
</tr>
<tr>
<td>Ottawa F2F feedback</td>
<td>August 25th, 2010</td>
<td>Added “uses” directive, added List attribute type, and more namespace clarifications.</td>
</tr>
<tr>
<td>CPEG call feedback</td>
<td>August 30th, 2010</td>
<td>Added note about case sensitivity, defined List syntax, clarified some host/fragment issues, added permission.</td>
</tr>
</tbody>
</table>
1 Introduction

The OSGi specification defines a handful of different dependency types that can 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 specific capabilities from the underlying platform, such as particular hardware devices or windowing toolkit. Since the target of the dependency is not manifested as a bundle, there is no proper way to express such dependencies. It would be worthwhile if the framework and 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 RFC intends to provide a generic dependency model for bundles that will be understood and enforced by the resolver. Since existing resolve-time dependencies are related to code visibility management, the resolution of such dependencies and the RESOLVED bundle life cycle state have traditionally indicated that the bundles in question are "ready" to share and/or use their code. With a generic dependency model, this meaning is stretched to some degree.

Further, some dependencies are explicitly or implicitly tied to the ACTIVE bundle life cycle state and are not actually "ready" at resolve-time. For example, service dependencies require that both the provider and consumer components are ACTIVE for them to be "ready", while extender/extendee dependencies typically require (at a minimum) the extender bundle to be ACTIVE for them to be "ready". This RFC is intended to address "static dependencies" and not these sorts of "active dependencies". However, since this is a generic dependency module, it is difficult, if not impossible, to limit its application to static dependencies. As such, this RFC will attempt to differentiate between static and active dependencies, but will only be able to enforce static (i.e., resolve-time dependencies).

The main impacts of modeling active dependencies with this mechanism are pushing the dependency resolution earlier in the bundle life cycle than what is strictly necessary and effectively concluding that the bundles are "ready" before they can technically be ready. The former impact cannot be avoided and typically results in having to deploy more bundles up front to resolve all dependencies at resolve time. The second impact can effectively be addressed by layering additional mechanisms on top to transition bundles to their appropriate states based on more detailed knowledge about their active dependencies.
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 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.

3.2 Limitations of Bundle-RequiredExecutionEnvironment

3.2.1 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 following situations: a bundle that 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.

3.2.2 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.
3.3 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.

3.4 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. Technically, this is an active dependency and can only be partially addressed by this RFC when modeled as a static dependency.

4 Requirements

REQ01 The solution MUST provide a way to capture dependencies among bundles not related to code visibility.

REQ02 The solution MUST provide a way to capture dependencies between bundles and the underlying platform/execution environment.

REQ03 The solution MUST be verifiable by the resolver without significant additional overhead.

REQ04 The solution MUST be limited to resolve-time enforcement/verification of dependencies.

REQ05 The solution MUST differentiate between static dependencies (i.e., resolve-time) versus active dependencies (i.e., active-time).
5 Technical Solution

This RFC proposes additional manifest headers for bundles to declare arbitrary provided capabilities and requirements on them. These headers impact resolution of the bundles involved and the dependencies among them, but have no other impact. 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 only "injects" dependencies among bundles, but does not grant access to any capability as a result of this injection. 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 namespace syntax is limited to the same construction rules as bundle symbolic names (i.e., dot-delimited strings). The specification reserves the "osgi." namespace for its own purposes. Comparisons for namespaces and attribute names are case sensitive.

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
                  ( ';' effective-directive ) ( ';' uses-directive )
                  ( ';' typed-attr) * ( ';' untyped-attr ) *
effective-directive ::= 'effective' ':=' ( 'resolve' | 'active' )
uses-directive ::= 'uses' ':=' packages
packages ::= '"' package-name ( ',' package-name ) '"'
typed-attr ::= token ':' attr-type '=' argument
untyped-attr ::= token '=' argument
attr-type ::= [ 'String' | 'Version' | 'Long' | 'Double' | 'List' ]
```

The effective directive is used to indicate at which time the capability is in effect. The solution proposed by this RFC will only consider capabilities with effective time resolve; all other capabilities are ignored. If the effective directive is not specified, then resolve is assumed.

The uses directive for generic capabilities has the same meaning as defined for the `Export-Package` header. Specifically, it is a comma-delimited list of package names on which the associated generic capability depends. The specified package names are ones either exported or imported by the bundle providing the generic capability; if not, then it is ignored. It is not possible to specify uses constraints on arbitrary generic capabilities, only on packages.
The presence of the uses directive on a generic capability causes the resolver to place an additional constraint on any bundles requiring it. The additional constraint requires that if they also import a package used by the generic capability, then they must get it from the same source as the bundle providing the generic capability.

It is not possible to make generic uses constraints transitive. For example, with packages you can specify a uses constraints on your exported package to an imported package; doing so makes the uses constraints from the provider of the imported package apply to importers of the original exported package (i.e., they are transitive). The analog is not possible with generic capabilities. Uses constraints cannot be transitive across generic requirements on generic capabilities, since it is not possible to refer to generic requirements in the uses constraint of a generic capability.

An example capability is:

```
Provide-Capability: foo-extender; version:version="1.1.0"; secure=false
```

An untyped attribute defaults to String type.

An attribute of type List may also include a component type declaration as well as a comma-delimited list of values, such as:

```
versions:List<Version>="1.0.0,1.0.1"
```

If the value of a list element must include quotes or commas, then they must be escaped.

### What about whitespace?

To require a generic capability, the Require-Capability header is used, which has the following syntax:

```
Require-Capability ::= requirement ( ',' requirement ) *
requirement ::= namespace
( ';' effective-directive )
( ';' resolution-directive )
( ';' filter-attribute )

effective-directive ::= 'effective' ':=' ( 'resolve' | 'active' )
resolution-directive ::= 'resolution' ':=' ( 'mandatory' | 'optional' )
filter-attribute ::= 'filter' '=' ldap-filter
```

The effective directive is used to indicate at which time the requirement is in effect. The solution proposed by this RFC will only enforce/verify requirements with effective time resolve; all other requirements are ignored. If the effective directive is not specified, then resolve is assumed.

The resolution directive is used to indicate whether a requirement is mandatory or optional, following the same semantics as optional imports.

An example requirement for some imaginary extender bundle is:

```
Require-Capability: foo-extender;
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.

There is no concept of substitutable capabilities like we have for exported packages. Therefore, there is no special meaning placed on a bundle providing and requiring a given capability namespace, even if it matches
itself. Such occurrences are treated as independently and are specifically allowed to match each other, which may be necessary if a fragment supplies something needed by a host.

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 default effective time of generic capabilities and requirements 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.

The resolver will only match a generic requirement to a capability if both have an effective time of resolve, since the framework ignores requirements and capabilities with any other effective time. On the other hand, upper layers are free to match active requirements to resolve capabilities.

5.4 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.

RFC 151 is refactoring the Package Admin API and is providing an API for reflecting over bundle dependencies in a generic way, which will encompass reflecting over the resulting RFC 154 dependencies. Note that RFC 151 maps existing capabilities and requirements (e.g., packages, bundles, etc.) into this generic model for reflective purposes. As a result, it creates specific namespaces under ".osgi." that may be illegal to directly specify in the manifest. For example, it is not possible to use the "osgi.package" namespace to export or import a package. An installation exception will be thrown for any bundle specifying a non-allowed OSGi namespace in its manifest.

5.5 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,

5.6 Cardinality of requirements

All generic requirements are assumed to match only a single provider.

5.7 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.

5.8 Host/fragment issues

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.

Any fragment capabilities that are attached to a host bundle will appear as if they come from the host bundle after attachment. This is necessary since fragments can attach to multiple hosts, which effectively makes them different providers with respect to “uses” constraints.

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.

6.5 Bundle state related capabilities.

Framework level handling of bundle state issues was ruled out on complexity grounds. Whether it is possible and or 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.

### 6.6 Management agent control of namespaces

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:

```
org.osgi.framework.processcapabilities = ' -*'|*|(-)?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.

### 7 Security Considerations

The following permission is defined to control providing and requiring generic capabilities.
**Class CapabilityPermission**

`org.osgi.framework`

```
java.lang.Object
  | java.security.Permission
  | java.security.BasicPermission
  | org.osgi.framework.CapabilityPermission
```

All Implemented Interfaces:
- Guard, Serializable

final public class `CapabilityPermission`
extends `BasicPermission`

A bundle's authority to provide or require a capability.

- The `provide` action allows a bundle to provide a capability matching the specified filter.
- The `require` action allows a bundle to require a capability matching the specified filter.

Since: 1.6
Version: $Id: 11e47883a0c36de7cd69d0436287d41f8bd30f17 $
ThreadSafe

## Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String PROVIDE</td>
<td>13</td>
</tr>
<tr>
<td>static String REQUIRE</td>
<td>13</td>
</tr>
</tbody>
</table>

- The action string `provide`.
- The action string `require`.

## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CapabilityPermission(String name, String actions)</code></td>
<td>13</td>
</tr>
<tr>
<td>Create a new CapabilityPermission.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CapabilityPermission(String namespace, Map&lt;String,?&gt; attributes, Bundle providingBundle, String actions)</code></td>
<td>14</td>
</tr>
<tr>
<td>Creates a new requested CapabilityPermission object to be used by code that must perform checkPermission for the require action.</td>
<td></td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean <code>equals(Object obj)</code></td>
<td>15</td>
</tr>
<tr>
<td>Determines the equality of two CapabilityPermission objects.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String <code>getActions()</code></td>
<td>14</td>
</tr>
<tr>
<td>Returns the canonical string representation of the actions.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int <code>hashCode()</code></td>
<td>15</td>
</tr>
<tr>
<td>Returns the hash code value for this object.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean <code>implies(Permission p)</code></td>
<td>14</td>
</tr>
<tr>
<td>Determines if a CapabilityPermission object &quot;implies&quot; the specified permission.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission Collection <code>newPermissionCollection()</code></td>
<td>14</td>
</tr>
<tr>
<td>Returns a new PermissionCollection object for storing CapabilityPermission objects.</td>
<td></td>
</tr>
</tbody>
</table>
Field Detail

REQUIRE

public static final String REQUIRE = "require"

   The action string require.

PROVIDE

public static final String PROVIDE = "provide"

   The action string provide.

Constructor Detail

CapabilityPermission

public CapabilityPermission(String name,
                          String actions)

Create a new CapabilityPermission.

The name is specified as a dot-separated string. Wildcards may be used.

name ::= <namespace> | <namespace ending in ".*"> | *

Examples:

   com.acme.capability.*
   org.foo.capability
   *

For the require action, the name can also be a filter expression. The filter gives access to the capability attributes as well as the following attributes:

   • signer - A Distinguished Name chain used to sign the bundle providing the capability. 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 providing the capability.
   • id - The bundle ID of the bundle providing the capability.
   • name - The symbolic name of the bundle providing the capability.
   • capability.namespace - The namespace of the required capability.

Since the above attribute names may conflict with attribute names of a capability, you can prefix an attribute name with '@' in the filter expression to match against the capability attributes and not one of the above attributes. Filter attribute names are processed in a case sensitive manner.

There are two possible actions: require and provide. The require permission allows the owner of this permission to require a capability matching the attributes. The provide permission allows the bundle to provide a capability in the specified capability namespace.

Parameters:

   name - The capability namespace or a filter over the attributes.
   actions - require,provide (canonical order)

Throws:

   IllegalArgumentException - If the specified name is a filter expression and either the specified action is not require or the filter has an invalid syntax.
Class CapabilityPermission

CapabilityPermission

```java
public CapabilityPermission(String namespace,
        Map<String,?> attributes,
        Bundle providingBundle,
        String actions)
```

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

Parameters:
- namespace - The requested capability namespace.
- attributes - The requested capability attributes.
- providingBundle - The bundle providing the requested capability.
- actions - The action require.

Throws:
- IllegalArgumentException - If the specified action is not require or any parameters are null.

Method Detail

implies

```java
public boolean implies(Permission p)
```

Determines if a CapabilityPermission 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

```java
public String getActions()
```

Returns the canonical string representation of the actions. Always returns present actions in the following order: require, provide.

Overrides:
getActions in class BasicPermission

Returns:
The canonical string representation of the actions.

newPermissionCollection

```java
public PermissionCollection newPermissionCollection()
```

Returns a new PermissionCollection object for storing CapabilityPermission objects.

Overrides:
newPermissionCollection in class BasicPermission

Returns:
A new PermissionCollection object suitable for storing CapabilityPermission objects.
equals

public boolean equals(Object obj)

Determines the equality of two CapabilityPermission objects. Checks that specified object has the same name and action as this CapabilityPermission.

Overrides:
equals in class BasicPermission
Parameters:
obj - The object to test for equality.
Returns:
true if obj is a CapabilityPermission, and has the same name and actions as this CapabilityPermission 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.

8 Document Support

8.1 References

8.2 Author’s Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Richard S. Hall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td></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:richard.s.hall@oracle.com">richard.s.hall@oracle.com</a></td>
</tr>
</tbody>
</table>

8.3 Acronyms and Abbreviations
8.4 End of Document
Abstract

This RFC describes a proposed specification for a Command processing interface for the OSGi Framework.
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 .................................................................................................................................. 5

2 Problem Description .................................................................................................................... 6
  2.1 Command Interface ............................................................................................................. 6

3 Requirements ............................................................................................................................... 6
  3.1 Non Functional.................................................................................................................... 6
  3.2 Command Names............................................................................................................... 7
  3.3 Shell .................................................................................................................................... 7
  3.4 Shell Commands................................................................................................................. 7
  3.5 Source Providers................................................................................................................. 8

4 Technical Solution ....................................................................................................................... 8
  4.1 Shell Design ........................................................................................................................ 8
  4.2 Command Service ............................................................................................................... 9
  4.3 Thread IO Service............................................................................................................... 10
  4.4 Standard IO Handling.......................................................................................................... 10
  4.5 Command Provider Discovery............................................................................................. 11
  4.6 Other Commands................................................................................................................ 12
  4.8 Converters........................................................................................................................... 13
  4.9 Services and their Commands.............................................................................................. 15
  4.10 Help ................................................................................................................................... 15
  4.11 Feedback from Bug Report................................................................................................. 15
  4.12 Javadoc............................................................................................................................. 16
      4.12.1 org.osgi.service.command
            Interface CommandProcessor ................................................................. 16
      4.12.2 org.osgi.service.command
            Interface CommandSession ................................................................. 18
      4.12.3 org.osgi.service.command
            Interface Converter .................................................................................. 22
      4.12.4 org.osgi.service.command
            Interface Function ................................................................................... 25
5 Shell Syntax.................................................................................................................................. 28
   5.1 Shell syntax: TSL................................................................................................................ 28
       5.1.1 Introduction to TSL (Tiny Shell Language)............................................................... 28
       5.1.2 Program Syntax.......................................................................................................... 29
       5.1.3 Examples of Syntax usage........................................................................................ 31
   5.2 Piping.................................................................................................................................. 31
   5.3 Command Calling................................................................................................................ 31
       5.3.1 Remove Variable........................................................................................................ 31
       5.3.2 Assignment................................................................................................................. 31
       5.3.3 Single Value................................................................................................................ 31
       5.3.4 Call Cmd..................................................................................................................... 32
       5.3.5 Message Send............................................................................................................. 32
   5.4 Argument Coercion.............................................................................................................. 32
   5.5 Printing or Not..................................................................................................................... 33
   5.6 TSL In OSGi........................................................................................................................ 33

6 Alternatives................................................................................................................................... 34

7 Security Considerations.............................................................................................................. 35

8 Document Support....................................................................................................................... 35
   8.1 References.......................................................................................................................... 35
   8.1 Author's Address................................................................................................................. 35
   8.2 Acronyms and Abbreviations............................................................................................... 35
   8.3 End of Document................................................................................................................. 36

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>05 MAR 2008</td>
<td><a href="mailto:Peter.Kriens@aQute.biz">Peter.Kriens@aQute.biz</a></td>
</tr>
</tbody>
</table>
1 Introduction

This RFC is an solution for RFP 99 Command Provider. This RFC outlines the interfaces necessary to implement command line shells in OSG frameworks.
2 Problem Description

This RFC addresses the problem of standardized external access. The purpose of this RFC is to enable a set of basic commands that are supported by all implementations, but it must enable that bundles can provide additional commands.

2.1 Command Interface

There is a need for a service that allows human users as well as programs to interact with an OSGi based system with a line based interface: a shell. This shell should allow interactive and string based programmatic access to the core features of the framework as well as provide access to functionality that resides in bundles.

Shells can be used from many different sources it is therefore necessary to have a flexible scheme that allows bundles to provide shells based on telnet, the Java 6 Console class, plain Java console, serial ports, files, etc.

Supporting commands from bundles should be made very lightweight and simple as to promote supporting the shell in any bundle. Commands need access to the input and output streams. Commands sometimes need to store state between invocations in the same session.

There is a need for a very basic shell functionality in small embedded devices, however, the design should permit complex shells that support background processing, piping, full command line editing, and scripting. It is possible that a single framework holds multiple shells.

The shell must provide a means to authenticate the user and the commands must be able to investigate the current user and its authorizations, preferably through standardized security mechanisms. To minimize footprint, it must also be possible to implement a shell without security.

3 Requirements

3.1 Non Functional

- Lightweight to allow shells for low footprint devices
- Allow shells with piping, background, scripting, etc.
- Make commands trivial to implement
- Make it easy to connect the shell to different sources.
- Provide an optional security framework based on existing security facilities
- Minimize the cost of a command (e.g. do not require eager loads of objects implementing commands)
• Support use of existing code by making a design that closely follows practices for command line applications in Java.

3.2 Command Names

• Provide a list of basic command signatures to manage the framework so they are consistent among implementations.

3.3 Shell

• Provide interface to execute a string as command
• Allow other bundles to implement commands
• Allow other bundles to provide a connection to: telnet, console, serial port, etc.
• Provide help for each command
• Provide a means to disambiguate commands with the same name
• Provide a means to disambiguate when there are multiple shells
• Authenticate the user
• Provide programmatic access to the shell, that is, a program generates the commands.

3.4 Shell Commands

● Read input from user or previous command
● Write output to user or next command
● Allow sessions, i.e. group commands over a period of time, allowing them to share state
● Provide usage information of the command
● Allow commands to be protected with permissions
● Provide access to the authenticated user via User Admin (though User Admin may not be present)
● Optional: Allow computer readable meta information about the commands to support forms
● Optional: Provide formatting rules + library to standardize look and feel for output. This could consist of routines to show tables in a consistent form.
● Commands should not have to do low level parsing of command line arguments.
● Commands should be able to have access to the command line arguments
● Commands must be able to get access to the console input
● Commands must be able to use the keyboard input stream
3.5 Source Providers

- Provide an easy way to allow bundles to connect the shell to sources like telnet, serial ports, etc.

4 Technical Solution

The designs in this section are to be part of the Compendium Specifications.

The drivers of this design have been:

- Core Engine Implementable in < 30k
- Very easy to add new commands
- Leverage existing mechanisms

The basic idea of the design is that there are three parts. The bundle that interacts directly with the user. This bundle handles the IO streams and parses out one or more lines of text, called the “program”. This program creates a Command Session from the selected Command service. This IO processor then gets a command from the input and gives it to a command session execute method. The command session parses the program, and executes it. The session then returns an Object result.

The command is executed synchronously. The shell will execute all commands in the program. These commands are implemented by services. A service can be registered with list of COMMAND_FUNCTION properties. These properties list the commands (potentially wildcarded) that a service can provide to the shell. These functions do not require a specific prototype, the shell matches the parameters to the function using parameter coercion. The type information available in the reflection API is used to convert the strings in the input to specific types.

Each command can print to System.out and it can retrieve information from the user (or previous command in the pipe) with System.in. Each command can also return an object.
Therefore, the Command Shell service consists of three distinct parts:

- **Command Processor service** - This service is used by bundles that can connect the shell to an outside interface like: Telnet, Console, Web, SSH, etc. These bundles provide streams or a Terminal service and get a Command Session in return. The session is then used to execute their commands.

- **Command Provider service** - Command implementations can register this service to provide commands. Commands are methods on the service. The names (and help) of the methods can be listed through properties. There is no actual Command Provider interface because a service property allows any service to provide commands.

- **Converter Service** – Provide facilities to convert from strings to specific types and from specific types to strings.

- **Formatter Service** – Provides a facility to format objects as strings with three levels of detail.

- **ThreadIO Service** - Commands can use System.in, System.out, and System.err to interact with the user. However, this requires that different commands are separated in their output. This RFC therefore defines a service (likely a Framework service) that can multiplex the System IO streams.
4.1 Command Service

The command service consists of the following interfaces:

- **CommandProcessor** – The engine is running the scripts. It has no UI of its own. A UI (telnet, web, console, etc.) is expected to create a session from this engine. The Command Processor service is registered by the implementer of the script processor.

- **CommandSession** – The command session represents the link between a UI processor (telnet) and the command processor. It maintains a set of variables that can be set by the UI processor as well as from the script. Commands should maintain any state in the session. The session is also associated with a keyboard stream as well as a console stream. This allows commands to directly talk to the user, regardless if they are piped or not.

- **Function** – A function is an executable piece of code. Commands providers can add Function objects to their arguments and execute them. This allows commands that implement iteration blocks, if statements, etc.

- **Scope** – Provides access to meta-information about to the sub-commands of a scope. The Command Session can provide a list of scopes.

- **Terminal** – An interface to a terminal allowing clients to control the cursor and attributes of the screen.

- **ArgumentList** – A list of arguments that is expanded when processing a command line.

4.2 Starting a Session

A user of the Command Processor must create a Command Session to use the script. The following code executes a small program, assuming it is injected with a CommandProcessor called cp:

```java
ByteArrayOutputStream bout = new ByteArrayOutputStream();
CommandSession session = cp.createSession(System.in, bout, System.out);
Object result = session.execute("bundles|grep aQute");
String s = new String(bout.toByteArray());
```

In this case the session is created on a set of streams. It is also possible to create a session with a Terminal. A Terminal has the standard streams but provide additional capabilities to control the screen is color, attributes and position of the cursor. If there is no terminal provided, the Command Processor service must provide a default terminal. The current terminal can be obtained with getTerminal().

4.2.1 Syntax

The tsl language requires a very simple parser to keep its footprint small. Basically a command line is broken into tokens, where the rules to parse a token are quit simple. These tokens are then each evaluated. The resulting list is then executed. Depending on patterns in this list it is either a command execution, a method call, an assignment, or a variable removal. Command execution and method calls use the Java type information to coerce the parameters to the target types.

Some examples:

```
$ echo Hello World
Hello World
```
The first token is the command name because it is a simple string, and is a command.

Objects implementing a command (that is, having a method with a command name) are registered as a variable with a structure. The same object can be registered under many different names. That is, if an object implements ls and cd, then it is registered as file:ls and file:cd.

As can be seen in this example, the actual name of the command is a structured name consisting of a <scope> and a function name, separated by a ':'. I.e. in the earlier example the command '*:echo' is searched because the scope is not defined.

The commands are coming from the variable space of the session. It must be possible to register commands by creating variables of the proper name.

A command is represented by a Function object. In the hello world examples, the execute method is called on this object with two CharSequence parameters: ["Hello","World"]. Because the number of arguments of echo is variable, it is declared with an array of Object.

```java
public CharSequence echo( Object ... args ) {
    StringBuilder sb = new StringBuilder();
    for ( Object arg : args )
        sb.append(arg);
    return sb;
}
```

Methods can print to System.out, but are normally expected to return an object. Returning an object allows the result to be used in other commands. Tsl will print out the object to standard out if it is not used as a value in another command, for example with piping. If a program contains multiple statements, only the last value is printed out. Converter services are used to print out the objects in a proper format.

### 4.2.2 Program Syntax and Semantics

The syntax of the shell should be simple to implement because the framework must provide a parser for this syntax. On the other hand, a more powerful syntax simplifies the implementation of the commands. For example, when Microsoft introduced a command line shell, it did not support piping. As a consequence, each command had to implement functionally to page the output. There are other examples like handling of variables, executing subcommands, etc.

The shell syntax must also be easy to use by a user. That is, a minimum number of parentheses, semicolons, etc. Some compatibility with the Unix shells like bash is desired to for users to not have to learn completely new concepts. Then again, the current popular shells have a convoluted syntax because they added more and more features over time.

An OSGi shell syntax can rely 100% on the fact that there is a Java VM. This makes it easy to control the framework and implement a shell with Java. However, a shell implies that the users directly types the commands as they go. The requirements for a shell are therefore different than the requirements for a programming language. However, in contrast with a shell like bash that must be totally text based, it seems a waste not to tie the shell language closely to the Java object model. Though there are many script languages, there seems to be no shell language for Java that provide such a syntax. Jacl comes close but has the disadvantage that it brings its own function library derived from tcl. Beanshell comes close from the other side but has a syntax that is virtually the same as Java, which contains a lot of cruft characters. A new syntax can reuse the concepts of tcl but tie the language close to the Java language. I.e., no need for separate function libraries.

The syntax and dependencies are defined in an ANTLR file for now. The specification will write out those rules in more detail. By keeping this in the ANTLR it is easier to verify changes.

```
grammar tsh;
```
@header {
package output;
}
@lexer::header {
package output;
}

// The pipe's have higher precedence so a program is a collection of
// pipes separated by newline or semicolons. It is allowed to
// have no pipe after the last semicolon/nl
//
tsh         : program ;
program     : pipe  (separator (pipe| ))*  ;

// A pipe consists of statements separated by a vertical bar. Each statement
// must be executed in a separate thread. The System.out of an earlier
// statement must be connected to the System.in of a later statement. The first
// statement has the current Terminal as input and the last statement must have
// the Terminal as output. The value of a pipe is the result of the last
// statement. If a statement that is not the last statement returns
// an object than this object must be formatted and send to System.out.
//
pipe        : statement ( '|' statement )* ;
statement   : TOKEN '=' pipe                        // Assignment
            | TOKEN '='                             // Remove var
            | TOKEN value*                          // Command (scope:function)
            | value+                                // Method invocation
            ;

// A core aspect of tsh is the concept of a value. A value is thought of
// as a string that can be evaluated to provide an object depending on
// its type. The type is defined by its first character. The following
// types are defined:
//
value      : TOKEN                                // Simple token
            | STRING                               // Single or Double Quoted
            | '{' program '}'                      // Closure
            | '{}' program ')'                     // Evaluate
            | '[' value* ']'                       // list
            | '<' ( value '=' value ) * '>'         // map
            | '$' TOKEN                            // basic macro
            | '${' TOKEN ( ':' value )? '} '        // braced macro
            | '${' program ')'                     // program macro
            ;

// Specifies the separator when there are multiple pipes.
//
separator  :    ';' | '
' | COMMENT;

Copyright © OSGi Alliance 2010 All Rights Reserved
// A single or double quoted string. Double quoted strings are
// expanded with macros, single quoted not. A string can contain
// virtually any character except its start character including
// newlines. Escaping for the quotes and other escapes are supported.

STRING : '\'' (ESC_SEQ | ~(\'|'\''))\s+ '``'
        | """ (ESC_SEQ | ~(\'|"""))\s+ '""';

// The TOKEN is the basic element of a command line. A TOKEN is
// very liberally specified to be a sequence of one or more characters that
// is not SPECIAL to this parser.

TOKEN : ~SPECIAL+;

fragment SPECIAL : '\u0000'..'\u001F'
                  | '#'
                  | '\'
                  | '='
                  | '{'
                  | '}'
                  | '('
                  | ')'
                  | '\$
                  | '<'
                  | '>'
                  | '\['
                  | '\]'
                  | '\n'
                  | '\r';

// Comments start with # and are ignored.

COMMENT : '#' ~('\n'|'\r')\s+ '\r'? '\n' ;

// Whitespace is defined as spaces and tabs.

WS : ( ' ' | '\t' | '\r' )\s+ { $channel=HIDDEN; };

// Escaping
// \b = backspace 08h
The previous syntax is rather extensive to be able to describe the semantics. However, TSH can rather easily be parsed in an unstructured way. The input can be very easily tokenized, where the closure, execution, list, and map must ensure that they match their respective end tokens correctly and ensure that their start character is ignored inside strings. For example, {{{} "{{{ "}}} would be a single closure. A statement is a collection of tokens without '|' or ';' (or newline) symbols. A pipe is a collection of statements separated by the '|' symbol and a program is a set of pipes.

### 4.2.3 Built-In Commands

Built-in commands are in the built-in scope.

- **Object new(String fqn, Object … args)** – Create a new object. The fqn is the fully qualified name and the args are given to the constructor.

- **void cat(File ..file)** – Copy a file to the stdin. This command replaces the Unix shell IO redirection.

- **void cat(URI ..file)** – Copy a file to the stdin. This command replaces the Unix shell IO redirection.

- **String tac(File)** – Copy a file to stdin, the return is the contents of the output. This command replaces the Unix shell IO redirection.

- **Object eval(String)** – Evaluate a string as a tsh script and return the result of the last statement.

- **Object eval(Object[])** - Evaluate a list of tokens.

- **ArgumentList args( Collection )** - Return an argument list from a collection. This list will be expanded when used in a command line.

- **void addcommand(String scope, Object object, String … functions )** - Add a new command to the session only.
• Object global(String, Object) – Set a global variable in the shell. This variable must be persistently stored.

• URI cd(String path) - Switch the directory indicated by path, relative to the cwd. Path can be an absolute URI, in that case the cwd becomes that URI. If path starts with a slash, it is an absolute file path. The path '..' indicates parent directory, the '.' indicates current directory. The return is the new working directory. The current working directory is available via the ${cwd} session variable.

• MetaScope help(String scope name, String ... command )

• Collection<MetaScope> help

4.2.4 osgi scope

The osgi scope is optional, however, if present must provide (in the given order):

• All methods on BundleContext. This must be the Bundle Context of the bundle that created the Command Session object.

• All commands on the bundle of the previous BundleContext

• StartLevel service

• Package Admin service

Additionally, it must provide the following commands:

• shutdown – exit the framework → (bundle 0) stop

4.2.5 Variables

There are the following variables (in order of priority)

1. Session – Variables that are set with the assignment operators always set the session variables. Session variables are local to the session and not shared with anybody else.

2. Global – The shell maintains global variables that are shared between all sessions. These variables are consulted if the session has no value for a given name. Global variables can be set with the global command.

3. Framework – If a variable cannot be found in the session or in the global variables then the framework is asked for the variable.

4. System Properties – If the Framework also cannot define a value the System properties are used.

The Command Processor maintains a map with all global variables, this Map can be obtained with CommandSession.getGlobalVariables. The session variables are available through a getSessionVariables() method. Framework variables or system properties cannot be directly set, they are filled by the Command Processor to provide share variables between sessions.

Session variables provide locally added or modified commands, variables set in the command line, or variables managed by commands.
Session variables are transient, they are not persistently stored, global variables that are set with the global command must be maintained persistently.

The following session variables provide general features:

- CWD – (URI) Current working directory
- SCOPE – (LIST modifiable) A list of scopes.

### 4.2.6 Redirection

Piping seems to introduce a significant complexity in the command processing. However, it turns out that it can be implemented with very little code that easily outweighs the advantages if the increased simplicity of the commands. The key example is of course the “less” or “more” command. Many of the other commands can generate output that is too much to fit the screen. Using piping, the output can be paged through a centralized command. Functions like grep, uniq, etc. are all impossible without piping.

Normal unix shells have io redirection. It was chosen to not implement this, but instead use commands. There are two commands that can redirect io:

- `cat <file>`+ concatenate files and pipe to output
- `tac <file>` receive input and store it in file or return object

That is, to get the output of bundles as a string in a variable:

```plaintext
output = (bundles | tac)
```

### 4.2.7 Invocation

In the end, a statement consists of a collection of parsed raw values, this collection is called the arguments. The arguments consists of all parts that were presented as a pipe in the grammar. During invocation, at first the arguments are the raw input. An argument can have a different type depending on its lexical value:

1. TOKEN
2. Closure
3. Execution
4. STRING
5. Argument List
6. List
7. Map
8. Macro Reference

Before a command is executed or a method invoked, the shell must look at the types of the arguments before evaluation and detect the pattern that is used for a command. There are a number of distinct cases in priority order:

1. TOKEN '=' pipe – The assignment pattern consists.
2. TOKEN '!=' - The variable removal pattern
3. TOKEN value+ - Command

4. value value value* - General method invocation

These patterns must be detected before the raw arguments are evaluated because the evaluation will lose an important distinction: it is possible that a value is evaluated to a general string making it impossible to distinguish between. For example, the statement "abc" length will map to the general method invocation while abc length maps to a command execution. After evaluation, they're equal.

The next step is therefore to evaluate all arguments (including the first). Evaluating has the following meaning for the different types:

1. TOKEN – Evaluates to String with its contents
2. Closure – Evaluates to an object implementing Function
3. Execution – The contents inside the parentheses are executed as a program. The resulting value is the value of the last pipe.
4. STRING – The value is the value of the string with any escaping applied. If it is a double quoted string, macros must be replaced as well.
5. Argument List – Take each member of the list as an evaluated value and add it as a new argument. This is used to implement the special behavior of $. It allows a closure to expand its arguments for a new command. Though $ does not implement ArgumentList, it is easy to expand with the args() method: (args $)
6. List – The list consist of a set of values. Each value must be evaluated and added to an ArrayList.
7. Map – The lists consists of a set of entries, where an entry is a value=value. Each entry must be evaluated and added to a LinkedHashMap.
8. Macro Reference – Macro references must be replaced with their value. See macros.

Arguments can refer to changing values, it is therefore important that the evaluation for each pipe takes place just before it is invoked. After the evaluation, the shell has a collection of evaluated arguments. For example:

$ foreach (bundles) {echo($it location)}

This command first creates the following collection of raw arguments:

1. TOKEN 'foreach'
2. Execution 'bundles'
3. Closure with value 'echo($it location)'

After evaluation, the arguments looks like:

1. String: 'foreach'
2. Bundle[]
3. Function
1. Assignment – first argument is the name of the variable. The evaluated objects after the equal sign are a pipe. This pipe must be evaluated to get the value, this value is assigned to the given variable. For example: \( bs = (\text{bundles}) \text{length} \).

2. Removal – Remove the variable with the name of the first argument, for example: \( bs = \)

3. Command – The first argument in the collection is the name of the command, how this name is interpreted is explained in the command provider discovery. For example: \( \text{obr list -f x com.example} \)

4. Invocation – The first argument is the object, the second value. For example: \( \$abc \text{length} \).

In the case of the Command and the Invocation case it is necessary to map the remaining arguments to the parameters of the method invocation. In the case of the command call, the arguments after the command name are all method parameters because the command name implies an object and method, in the case of an invocation the values after the object and the method name are values. When a Java method is called, it is necessary to match the arguments to the method that has the best parameters to match those arguments. This requires that parameters are converted to their correct type.

Assume we have a target object \( i \). If the execution is about a command, \( t \) is found from the command name. If the case is the invocation then \( t \) is the evaluated first value. Next the method \( m \) must be selected. In the command case, \( m \) is implied by the command name as described in provider discovery. In the invocation case \( m \) is the second evaluated value. So yes, \( \$abc \$def \) will invoke the method in \( \$def \) on the object in \( \$abc \).

For \( t, m \) must be compared to all public declared methods (no inheritance). So for each declared method:

1. If \( m \) is a reserved word, accept method names that start with '_-' and then \( m \). That is, if \( m \) is “new”, then a method with the name “_new” must be accepted.

2. Compare the name case insensitive.

3. Apply the bean design pattern to the method name and compare the name of the property in a case insensitive way. That is, a command like “bundles” must find the method getBundles and setBundles and isBundles. See the beans design patterns for proper converting a name to a getter.

Going through all declared methods will return a list of methods that match the name. The next step is to sort this list with the methods that best match the given set of parameters.

1. If the first parameter of a method is a CommandSession, then the Command Shell must insert the current session in this parameter. This is a way for a command to receive the shell session itself. This can be used to recursively execute commands, to get access to the Terminal, or to get and put variables. The Terminal streams are for example necessary to do a “more” command. CommandSession parameters are ignored for the coercion, for the coercion, this case looks like the method did not have a first parameter.

2. The parameters at the beginning of the method can be annotated with a Parameter annotation. Parameter annotations can be used for flags (e.g. -t) or options (e.g. -f out.txt). For each annotated parameter, the argument list must be searched for a matching parameter alias. If the argument list contains ‘-’ (two consecutive dashes) then no parameters should be looked for after this marker to allow options and flag names to be used as proper parameters.

If found, the Parameter must be asked for its ifPresent() value. If this contains the magic value NOT_SET then this is an option and the argument after the matched alias must be used to inject it in the annotated parameter. If the ifPresent() value has a different value, then this value must be used to inject the
corresponding parameter. The alias as well as the option value (if it is an option) must then be removed from the argument list. If the flag or option is not used, the ifAbsent() value must be used.

3. The use of an option or flag (before the '- -') is an error if that option or flag is not defined as an annotation.

For example:

$ foo -f out.bak -x /home/keystore bar biz

Is mapped to:

```java
void foo(CommandSession session,
    @Parameter(alias="-f", ifAbsent="a.out") File file,
    @Parameter(alias="-x", ifAbsent="false", ifPresent="true") boolean extra,
    @Parameter(alias="-v", ifAbsent="false", ifPresent="true") boolean verbose,
    File keystore,
    @Description("private keys") String[] keys);
```

The pruned argument list now contains only arguments that must be matched with the remaining method's parameters after the last parameter that has a Parameter annotation. In the previous example, only the File keystore and the String[]keys has to be matched.

If a method ends with an array then this can be treated as a a varargs method. The remaining arguments can be collected in an array. An empty array is acceptable.

Matching of the methods is done with the following priority:

1. Assignment has priority over conversion. That is a method with fewer necessary conversions wins.

2. A method that does not have to use varargs wins over a method that needs varargs

If no suitable method can be found and a “main(Object[])” method is available then this method must be called. The whole command line must be given, where the first parameter must be the name of the function including its scope (even if it was not specified on the command line).

If none is found, a NoSuchMethodException must be thrown.

### I removed padding ...

#### 4.2.8 Argument List

There is a special interface that changes the behavior of the command line processing: Argument List. A Argument List is a a List<Object>. The special behavior is that when a Argument List is used as a raw value, its content must be treated as a list of evaluated arguments. These evaluated arguments must be expanded in the list that is being processed. A normal List can be turned into an argument with the expand method. For example:

$ g = { grep -i (args $*) } //
$ g pattern file // → grep -i pattern file

Any list that implements this interface will expand its contents into the array of arguments.

The $* is a normal list and will this not automatically be expanded. It is also a mutable list which allows for alternative manipulations:

```sh
$ g = { $* add 0 “-i”; grep (args $*) } //
```
4.2.9 Objects, no Strings
Arguments are not strings, they are proper objects after evaluating. Variables can refer to objects, arrays are objects, and also the result of a direct command (using parentheses ()) can result in a proper object. Matching these objects to a method is non-trivial.

4.2.10 Strings
TSH supports two types of Strings: single quoted and double quoted. Double quoted strings can contain macros. A TSH implementation must replace the macros with their value when the statement's parameters are evaluated. This must happen for every evaluation so that the macro always reflects the latest value.

There are two types of macros:

- \${ ... } - Braced macros can contain a TOKEN as variable name and will be replaced with the actual variable value.
- \${ name : 'default '} - The special construct allows a default value to be specified if no variable value is found.
- $( ... ) - Parentheses indicate that the macro is actually a parenthesized value and must be evaluated.

To use a string like ${ or $( in the double quoted string, escape the $ like “This is a macro: \$(...)”.

4.2.11 Printing or Not
In principle, tsh must only print the object when it would otherwise gets lost. It will therefore only print the object when a command is piped because in that case there is nobody to use the resulting object. In all other cases, the object is kept.

If an object is printed, it must use the Aggregate Formatter service with level INSPECT.

4.2.12 Command Provider Discovery
Command Provider discovery is based on the OSGi service model. Any service can be used as a command provider.

Dedicated command providers must register their service with two properties:

- osgi.command.scope - This property defines the name of the command provider. This name is not normally used because the function names are unique. However, if the function names are no longer unique, then this scope can be used to disambiguate.
- osgi.command.function - The name of the function. This is an simple or an array property, so many names can be listed. This function name should match to a public method in the service object. If no function list is provided then the shell must get the functions by analyzing the class of the service object.
- osgi.command.description – A description of the scope

For example, the following code is a DS that provides a few utility commands:
public class Tools {
    public void grep(String match) throws IOException {
        Pattern p = Pattern.compile(match);
        BufferedReader rdr = new BufferedReader(
            new InputStreamReader(System.in));
        String s = rdr.readLine();
        while (s != null) {
            if (p.matcher(s).find()) {
                System.out.println(s);
            }
            s = rdr.readLine();
        }
    }
    public void echo(Object[] args) {
        StringBuffer sb = new StringBuffer();
        for (Object arg : args)
            sb.append(arg);
        System.out.println(sb);
    }
}

The properties provide sufficient information for the Command Shell to find the providers. Note that it is not necessary to register the service as a Command Provider, the properties suffice. This makes it possible to register these properties on an existing service. For example, the Configuration Admin could just register the following properties:

```
osgi.command.scope = 'cm'
osgi.command.function = {'createFactoryConfiguration', 'getConfiguration', 'listConfigurations'}
```

This will enable shell scripts like:

```
cfg = configuration com.acme.pid
$cfg update [port=23 host=www.acme.com]
```

Or, for the Log Service

```
command.scope = 'log'
command.function = 'log'
```

```
log 2 "hello world"
```

The Command Session's owner must be used for getting the service objects. That is, the bundle that uses the service must be the same as the bundle that got the command session.

### 4.2.13 Other Commands

Any other commands can be added to the shell by storing them in the session, for this purpose there is a built-in `addcommand` function.

Command names are scoped like `<scope>:<function>`. The value of this variable can be a plain object, or it can be an instance of `Function`. If it is an instance of `Function`, it can be directly executed. Else, the method with the function name is called upon it.
4.2.14 Services and their Commands

Implementations of services are recommended to provide commands for their service, this is quite straightforward and described in a later section. For example, assume that the implementation of the Configuration Admin has registered its method as commands. I.e. all its public methods are available.

$ my.pid = configuration my.pid; $my.pid update <port=5012 host=www.aQute.biz >

The configuration command is executed against the Configuration Admin service. This returns a Configuration object. In this case, it is stored in the my.pid variable. In the next statement, we call the update method on the Configuration object and set a dictionary.

Ok, one more example, based on the fact that the Configuration Admin is available as commands:

$ listConfigurations "(service.pid=com.acme.*)"| grep port

4.2.15 Closures

The proposed specification of TSL also has closures. Closures represent a program that can be evaluated later. They start with a '{' and end with a recursively matching '}'. Closures can be injected as a Function in the called method.

Closures implement the Function interface. The follow code will add a command written in tsl:

$ my:echo = { echo xx (args $*) xx }
Closure ...
$ my:echo Hello World
xx Hello World xx

A closure provides a facility for delayed execution. A closure starts with a '{', it can contain a complete program, and ends with a '}'. If evaluated it will be turned into an object implementing the Function nterface. A Function can be executed with parameters. A Function can be invoked many times if needed.

A closure has access to its parameters via the following macros that are only valid inside the closure:

$0 optional name of the script
$1..$n parameters in the given sequence, 1 is first parameter.
$it equals to $1
$* Mutable List<Object> that contains parameters $1 .. $n.

4.2.16 Help

The Command Processor has extensive information about the possible commands. This information is made available through the getMetaScopes() method on the Command Session. This method returns a snapshot of the collection of MetaScope objects. These objects are derived from the meta information of the scopes. It can come from:

1. The osgi.command.description property on a command service
2. The name of the methods, the return type, and the parameters method types
3. The Descriptor and Parameter annotations.

If both a property and a annotation is used, then the annotation must wins. If multiple scopes use the same name, then the descriptive texts must be concatenated with two newlines separating them.
4.2.17 Terminal

The Terminal class represents the output terminal or console. The terminal is either created by the Command Processor or, when the IO Handler passes streams end their encoding, or is created by the IO Handler. Whatever route is taken, the primary interface for IO in the shell is the Terminal.

The IO Handler can interface to many different types of terminals. From simple ASCII terminals to high level graphic windows. The capabilities can therefore differ significantly, this is reflected in the getCapabilities() method. This method returns a bitmap of capabilities.

Any terminal must support the ASCII code set and report and allow simple cursor movements. Even the most primitive ANSI terminal supports this and these movements can be emulated on even the simplest terminal.

A Terminal is associated with two Print Writer objects: System.out, System.err. Input is obtained from two methods: int getIn() and String readLine().

Normally, commands will use the System.{out,err,in} objects to do their output. It is only in special cases that the extended capabilities of the Terminal are needed. Using the Terminal will bypass any piping that was set up. The Command Processor must ensure that the System.* streams, which use the default encoding, are properly converted to the Terminal writers.

The Terminal interface provides a number of methods that can be used to control where and how the output stream ends up on the terminal. There are a number of methods that can query the terminal for its characteristics and there are a number of methods that can change the state of the terminal. Not all methods work on all terminals, the getCapabilities() method can inform the caller about any capabilities.

The terminal can report its width and height. The screen of the terminal should be seen as a table with rows and columns from [0,width) x [0,height). The left-top of the screen is position 0,0 and the right bottom is width-1, height-1. Any text displayed on the screen will not wrap but will be ignored.

The cursor (the place where any new text on the output Print Writer will be displayed) can be positioned with the setPosition method, which takes an x and y coordinate. The current output Print Writer must be flushed before the position is altered. The current position can be queried with getPosition().

Terminals can support a number of attributes that alter the way the screen is displaying the text. The setAttributes(Attribute[]) method sets these attributes. This is in association with the output Print Writer, not the screen position. Any subsequent writes will be displayed with the new attributes. The method returns an array of attributes which represent the previous set of attributes. Passing this array to setAttributes() will restore the previous state.

Terminals can support bi-directional text. Unicode characters have an associated direction. Terminals are not required to support bi-directional text. However, if they support bi-directional text then they must ensure that horizontal cursor movements and the movement caused by forward() and backward() are properly synchronized. That is, when the CURSOR_FORWARD is reported then forward() called, the forward method must adjust the movement depending on the direction of the underlying text. The same for CURSOR_BACKWARD and backward(). It is assumed that the underlying logic to calculate the direction is handled by the Bidi class.

In all cases, the left-top of the screen remains 0,0 and the default progression is left-to-right.

4.3 Thread IO Service

The Thread IO service is a framework service that guards the singletons of System.out, System.in, and System.err. It maintains a stack based model of streams per thread. Any party that wants to receive output or provide input on the current thread can push its streams. The ThreadIO service consists of two methods:
- **pushStreams(.InputStream, PrintStream, PrintStream)** – Push the given streams with the current thread. Any output on the current thread using any of the System Print Streams will in effect be redirected to the appropriate system stream. Input will come from the given input stream. This method can be repeated multiple times for a thread. That is, an implementation must stack the streams per thread. Streams may be null, in that case they refer to the last set stream or the default if no streams are set.

- **pop()** - Cancel (or pop) the streams from the thread local stack. If no more streams are available, use the value of the original System streams.

Usage of the Thread IO service is very straightforward but care must be taken that exceptions do not leave streams on the stack. For example, the following code grabs the output:

```java
String grab(ThreadIO threadio) {
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    PrintStream pout = new PrintStream(out);
    threadio.pushStreams(null, pout, pout);
    try {
        System.out.println("Starting ...");
        doWhatever();
        System.out.println("... Done");
    } catch(Throwable t) {
        t.printStackTrace();
    }
    finally {
        threadio.pop();
    }
    pout.flush();
    return new String(out.toByteArray());
}
```

### 4.3.1 Child threads

If a thread that is associated with a threadio stream creates a child thread (or the child creates a thread, ad nauseum) than that child thread must inherit the state of the parent thread that pushed the streams. This inheritance must be dynamic, that is, if the parent thread is popped, the child should see the new streams. A child thread cannot pop the stream, pushing and popping must occur on the same thread.

In Java, it is impossible to find the parent of a thread, nor is it possible to find out that a new thread is created. So far, there seems to be only one way to manage the parent child relation that, due to multiplexing, has to become public API.

An implementation should use a InheritableThreadLocal. The first time a thread pushes streams, the implementation must set a WeakReference to the current thread. To perform an IO operation, the thread local is consulted and the thread in that thread local is used, if the reference or the referred value is null, no redirection must take place.

An InheritableThreadLocal will copy the value of the parent (who points to the parent!) into the child. This indirection allows an implementation to find the parent and use its settings.

### 4.3.2 Multiplexing

Multiplexing will not be in this release. Once we've decided what to do with general multiplexing then we should add this.
4.4 Converter Service

The Converter service is built around the Aggregate Converter service and the Converter services. The aggregate converter tracks any Converter services. The Aggregate Converter dispatches conversion requests to the appropriate Converter service.

The API for both services consists of (Aggregate Converter extends Converter):

- `<T> boolean canConvert(Object sourceObject, ReifiedType<T> targetType)`
- `<T> T convert(Object sourceObject, ReifiedType<T> targetType);`

The ReifiedType is defined in Blueprint, the description of the Blueprint Type conversion must be copied to the specification text for the conversion.

The Aggregate Converter must support the basic conversions as are also defined in Blueprint.

A Converter service must register with the the service property osgi.converter.classes. Its value is of type String+, reflecting the classes this converter can convert. For conversion, inheritance is not taken into account. When TSH needs to convert an object to a class or print an object of a specific class, it will call the registered Converter objects in the following order:

- filtered by matching class
- sorted by service.ranking, highest first
- sorted by service.id, lowest first

4.5 Formatter Service

The Formatter service is built around the Aggregate Formatter service and the Formatter services. An Aggregate Formatter service can take an object, a level and a set of locales and return a string formatted version of this object. The Aggregate Formatter tracks any Formatter services. The Aggregate Formatter dispatches format requests to the appropriate Formatter service.

The API for both services consists of (Aggregate Formatter extends Formatter):

- `CharSequence format(Object target, int level, Locale... locales) throws Exception - Format an object to a Char Sequence using the int parameter as a hint.`

The hint can be INSPECT, LINE, or PART.

- INSPECT - For an INSPECT, the output can be a multiline columnar output of any reasonable level.
- LINE - A LINE format must make the object look good in a table when different objects of the same type are printed below each other.
- PART - It is allowed to use multiple lines as long as the format works well in a table. A PART format is used to identify the object. E.g. a name or identifier. The PART format should be usable in the convert method when a CharSequence is the object to be converted.

INSPECT, LINE, and PART are ordered. That is, when printing an INSPECT, the next level should be to format an object with LINE, etc.
A Formatter service must register with the service property osgi.formatter.classes. Its value is of type String+, reflecting the classes this formatter can format. For conversion, inheritance is not taken into account. When one needs to convert an object to a class or print an object of a specific class, it will call the registered Converter objects in the following order:

- filtered by matching class
- sorted by service.ranking, highest first
- sorted by service.id, lowest first

The following code shows a simple formatter for Bundle objects:

```java
import org.osgi.framework.*;
import org.osgi.service.command.*;

public class BundleConverter implements Formatter {
    BundleConverter(BundleContext context) {
        this.context = context;
    }

    public CharSequence format(Object o, int level, Locale locales) {
        if (!(o instanceof Bundle))
            return null;
        Bundle b = (Bundle) o;
        StringBuffer sb = new StringBuffer();
        switch (level) {
            case INSPECT:
                cols(sb, "Symbolic Name", b.getSymbolicName());
                cols(sb, "Version", b.getHeaders().get("Bundle-Version"));
                cols(sb, "State", b.getState());
                cols(sb, "Registered Services", escape.format(b.getRegisteredServices(), level + 1, escape));
                // ...
                break;
            case PART:
                sb.append(b.getSymbolicName()).append(";").append(b.getHeaders().get("Bundle-Version"));
                break;
            case LINE:
                sb.append(" ").append(b.getState()).append(" ").append(b.getLocation());
                break;
        }
        return sb;
    }

    void cols(StringBuffer sb, String label, Object value) {
    }
```
sb.append(label);
for (int i = label.length(); i < 24; i++)
    sb.append(' ');
sb.append(value).append('
');
}

5 Javadoc
## Package Summary

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.osgi.service.command</td>
<td>Command Package Version 1.0.</td>
<td>28</td>
</tr>
<tr>
<td>org.osgi.service.command.annotations</td>
<td>Command Annotations Package Version 1.0.</td>
<td>59</td>
</tr>
</tbody>
</table>

Copyright © OSGi Alliance 2010 All Rights Reserved
Package org.osgi.service.command

Command Package Version 1.0.

See:
Description

<table>
<thead>
<tr>
<th>Interface Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentList</td>
<td>A Parameter List must be expanded to its constituents when used in a command line.</td>
</tr>
<tr>
<td>CommandProcessor</td>
<td>A Command Processor is a service that is registered by a TSH script engine that can execute commands.</td>
</tr>
<tr>
<td>CommandSession</td>
<td>A Command Session holds the executable state of a script engine as well as a Terminal that is connected to that session.</td>
</tr>
<tr>
<td>Function</td>
<td>A Function is a a block of code that can be executed with a set of arguments, it returns the result object of executing the script.</td>
</tr>
<tr>
<td>MetaScope</td>
<td>A scope defines the meta information of many commands.</td>
</tr>
<tr>
<td>MetaScope.MetaFunction</td>
<td>A Meta Function describes a scoped function.</td>
</tr>
<tr>
<td>Terminal</td>
<td>The Terminal interface describes a minimal terminal that can easily be mapped to command line editing tools.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal.Attribute</td>
<td>An inner class to provide an enum for the attributes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exception Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParsingException</td>
<td>An exception that can point at the location where the error occurred.</td>
</tr>
</tbody>
</table>

Package org.osgi.service.command Description

Command Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.command; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.command; version="[1.0,1.1)"
```
public interface ArgumentList
extends List<Object>

A Parameter List must be expanded to its constituents when used in a command line. It allows the following pattern:

$ g = { grep -i (args $*) }
$ g pattern file -> grep -i pattern file

A ParameterList extends List, meaning that it can be manipulated as a list as well.
public interface CommandProcessor

A Command Processor is a service that is registered by a TSH script engine that can execute commands. A Command Processor is a factory for Command Session objects. The Command Session maintains execution state and holds the console and keyboard streams. A Command Processor must track any services that are registered with the `COMMAND_SCOPE` and `COMMAND_FUNCTION` properties. The functions listed in the `COMMAND_FUNCTION` property must be made available as functions in the TSH script language.

Field Summary

<table>
<thead>
<tr>
<th>String</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND_DESCRIPTION</td>
<td>30</td>
</tr>
<tr>
<td>COMMAND_FUNCTION</td>
<td>30</td>
</tr>
<tr>
<td>COMMAND_SCOPE</td>
<td>30</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>CommandSession createSession(InputStream in, PrintStream out, PrintStream err, String encoding)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new command session associated with IO streams.</td>
<td>31</td>
</tr>
<tr>
<td>CommandSession createSession(Terminal terminal)</td>
<td>31</td>
</tr>
<tr>
<td>Create a new Command Session that is associated with a <code>Terminal</code>.</td>
<td></td>
</tr>
</tbody>
</table>

Field Detail

**COMMAND_SCOPE**

public static final String `COMMAND_SCOPE` = "osgi.command.scope"

The scope of commands provided by this service. This name can be used to distinguish between different command providers with the same function names. Commands can be executed as `<scope>:<function>`.

**COMMAND_FUNCTION**

public static final String `COMMAND_FUNCTION` = "osgi.command.function"

A `String` of function names that may be called for this command provider. A name may end with a `*`, this will then be calculated from all declared public methods in this service. TODO verify the `*` is true?

**COMMAND_DESCRIPTION**

public static final String `COMMAND_DESCRIPTION` = "osgi.command.description"

A description of the command scope.
Interface CommandSession

Method Detail

createSession

CommandSession createSession(InputStream in, PrintStream out, PrintStream err, String encoding)

Create a new command session associated with IO streams. The session is bound to the life cycle of the bundle getting this service. The session will be automatically closed when this bundle is stopped or the service is returned. The shell will provide any available commands to this session and can set additional variables that will be local to this session.

Parameters:
  in - The value used for System.in. If null is passed, the implementation must create a valid InputStream that always returns end of file.
  out - The stream used for System.out, must not be null
  err - The stream used for System.err, must not be null
  encoding - The character encoding to use

Returns:
  A new session.

createSession

CommandSession createSession(Terminal terminal)

Create a new Command Session that is associated with a Terminal. A Terminal provides the common streams but adds extra capabilities for commands to control the screen. A session maintains this Terminal.

Parameters:
  terminal - The terminal to use in this session

Returns:
  A new sessions
Interface CommandSession

public interface CommandSession

A Command Session holds the executable state of a script engine as well as a Terminal that is connected to that session. A Command Session is not thread safe and should not be used from different threads at the same time.

Version:
$Id: 53298871e5dc9b0de675870722c838cf23cb6a26 $
NotThreadSafe

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void close()</td>
<td>32</td>
</tr>
<tr>
<td>Object execute(CharSequence commandline)</td>
<td>32</td>
</tr>
<tr>
<td>Map&lt;String, Object&gt; getGlobalVariables()</td>
<td>33</td>
</tr>
<tr>
<td>Collection&lt;MetaScope&gt; getMetaScopes()</td>
<td>33</td>
</tr>
<tr>
<td>Map&lt;String, Object&gt; getSessionVariables()</td>
<td>33</td>
</tr>
<tr>
<td>Terminal getTerminal()</td>
<td>33</td>
</tr>
<tr>
<td>URI resolve(String path)</td>
<td>33</td>
</tr>
</tbody>
</table>

Method Detail

execute

Object execute(CharSequence commandline)
  throws Exception, ParsingException

Execute a program in this session.

Parameters:
commandline - A Command line according to the TSH syntax.

Returns:
the result of the execution

Throws:
Exception - if something fails
ParsingException - If the text contained a syntax error

close

void close()

Close this command session. After the session is closed, it will throw IllegalStateException when it is used.
**getTerminal**

```
Terminal getTerminal()
```

Return the input stream that is the first of the pipeline. This stream is sometimes necessary to communicate directly to the end user. For example, a "less" or "more" command needs direct input from the keyboard to control the paging.

**Returns:**

InputStream used closest to the user or (`null`) if input is from a file.

---

**getSessionVariables**

```
Map<String, Object> getSessionVariables()
```

Return the map used to store the session variables. The returned map can be modified to add/remove new variables.

**Returns:**

the map with all the variables

---

**getGlobalVariables**

```
Map<String, Object> getGlobalVariables()
```

Return the map used to store the global variables. The returned map can be modified to add/remove new variables.

**Returns:**

the map with all the variables

---

**getMetaScopes**

```
Collection<MetaScope> getMetaScopes()
```

Return the current list of scopes. A scope represents a command and its sub-commands. The purpose of this information is to simplify command completion and providing extensive help about commands. If an implementation supports annotations it can use the annotations to provide extra information.

**Returns:**

A unmodifiable collection of `MetaScope` objects.

---

**resolve**

```
URI resolve(String path)
```

Resolve a local name to a URI to the current working directory. If the name is absolute, an absolute URI is returned.

**Parameters:**

* path - A relative or absolute URI

**Returns:**

a URI that is the original when it was absolute and resolved against the `$cwd` variable if relative.
A Function is a block of code that can be executed with a set of arguments, it returns the result object of executing the script. The purpose of the Function is to be injected in commands. Many commands require the possibility to execute closures or other commands. For example, a `foreach` command requires a block for execution:

```java
void foreach( Iterable<?> collection, Function block ) {
    for ( Object o : collection ) block.execute( Arrays.asList(o));
}
```

Though the majority of application is when functions are closures, functions can also be used as a target type for conversion.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td><code>execute(CommandSession session, List&lt;?&gt; arguments)</code> Execute this function and return the result.</td>
</tr>
</tbody>
</table>

### Method Detail

**execute**

```java
Object execute(CommandSession session, List<?> arguments)
throws Exception
```

Execute this function and return the result.

**Parameters:**
- `session` - The session in which to execute this function
- `arguments` - The list of arguments. This list will not be modified.

**Returns:** The result from the execution.

**Throws:** `Exception` - if anything goes wrong
public interface MetaScope

A scope defines the meta information of many commands. The information about the scopes can be obtained from
CommandSession.getMetaScopes(). If annotations are present, the information is augmented with special
parameter and descriptive information. Commands are stored in the session variables. Commands are added from
the service registry or commands are added programmatically, however, all commands are part of the session
variables and changing the session variables will change the available commands in a scope. Each command is
stored in the session variables under the name: @code :.

Version:
   $Id: ea32b8a44ffe81a1f903930b54d25b27aafda6f9$
ThreadSafe

<table>
<thead>
<tr>
<th>Nested Class Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static interface MetaScope.MetaFunction</td>
<td>37</td>
</tr>
<tr>
<td>A Meta Function describes a scoped function.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getDescription()</td>
<td>35</td>
</tr>
<tr>
<td>Return a description of this scope, if present.</td>
<td></td>
</tr>
<tr>
<td>Collection&lt;MetaScope.MetaFunction&gt; getMetaFunctions()</td>
<td>35</td>
</tr>
<tr>
<td>Return an unmodifiable list of Meta Function objects.</td>
<td></td>
</tr>
<tr>
<td>String getName()</td>
<td>35</td>
</tr>
<tr>
<td>Return the name of this scope.</td>
<td></td>
</tr>
</tbody>
</table>

**Method Detail**

**getName**

String getName()

Return the name of this scope.

Returns: Name of the scope, always non-null.

**getDescription**

String getDescription()

Return a description of this scope, if present.

Returns: A description or null

**getMetaFunctions**

Collection<MetaScope.MetaFunction> getMetaFunctions()
Return an unmodifiable list of Meta Function objects. This list is a snapshot of the current state and will not follow the state.

**Returns:**

an unmodifiable list of Meta Function objects.
public static interface MetaScope.MetaFunction

A Meta Function describes a scoped function.

<table>
<thead>
<tr>
<th>Nested Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static interface</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
</tr>
<tr>
<td>String getDescription() Return a description of this scope, if present.</td>
</tr>
<tr>
<td>String getName() Return the name of this function.</td>
</tr>
<tr>
<td>boolean isVarArgs() Answer if this is mapped to a vararg method.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>getName</td>
</tr>
<tr>
<td>String getName() Return the name of this function.</td>
</tr>
<tr>
<td>Returns: Name of the function, always non-null.</td>
</tr>
<tr>
<td>getDescription</td>
</tr>
<tr>
<td>String getDescription() Return a description of this scope, if present.</td>
</tr>
<tr>
<td>Returns: A description or null</td>
</tr>
<tr>
<td>getMetaParameters</td>
</tr>
</tbody>
</table>
Returns:
The meta arguments

isVarArgs

boolean isVarArgs()

Answer if this is mapped to a vararg method. A vararg method can be used to fill out the last argument of a function.

Returns:
true if this is for a vararg method, otherwise false

Field Summary

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAMETER_FLAG</td>
<td>40</td>
</tr>
<tr>
<td>PARAMETER_OPTION</td>
<td>39</td>
</tr>
<tr>
<td>PARAMETER_PROVIDED_BY_SESSION</td>
<td>40</td>
</tr>
<tr>
<td>PARAMETER_UNNAMED</td>
<td>39</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String[] getAliases()</td>
<td>40</td>
</tr>
<tr>
<td>String getDescription()</td>
<td>40</td>
</tr>
<tr>
<td>int getParameterType()</td>
<td>40</td>
</tr>
<tr>
<td>org.osgi.service.converter.ReifiedType getType()</td>
<td>40</td>
</tr>
<tr>
<td>String ifPresent()</td>
<td>41</td>
</tr>
<tr>
<td>String isAbsent()</td>
<td>41</td>
</tr>
</tbody>
</table>

Field Detail

PARAMETER_UNNAMED

public static final int PARAMETER_UNNAMED = 0

A parameter type without any meta information, a so called UNNAMED parameter. Unnamed parameters have no Flag or Option annotation. The UNNAMED parameters must be at the end of the parameter list for a method.

PARAMETER_OPTION

public static final int PARAMETER_OPTION = 1

The parameter type that is marked as option. An option can be omitted from the parameter list, in that case the isAbsent() must be used. Options are marked in the command line with names that start with `-`. Options must be followed by a value, for example...
foo -f (bar file) hello

PARAMETER_FLAG

public static final int PARAMETER_FLAG = 2

The parameter type is marked as a Flag. A Flag does not require a value in the command line. If the flag is used in the command line then the value is given by ifPresent(), otherwise the value is given by isAbsent().

PARAMETER_PROVIDED_BY_SESSION

public static final int PARAMETER_PROVIDED_BY_SESSION = 3

The parameter can be provided by the session, for example the Terminal object or the Command Session object.

Method Detail

getParameterType

int getParameterType()

Return the Parameter Type, either: 0, PARAMETER_FLAG, PARAMETER_OPTION, PARAMETER_PROVIDED_BY_SESSION.

Returns: the parameter type

description

String getDescription()

Provide a description of the parameter.

Returns: A description of the parameter or null of none available.

getAliases

String[] getAliases()

List the name of aliases for this parameter. An alias must start with a minus sign ('-') and must not be '--'. For example: {'-f', '--file'} If the list of aliases can be empty, in that case this must be an unnamed parameter.

Returns: Array of aliases.

type

org.osgi.service.converter.ReifiedType getType()
Return the Java type of the parameter. This is a Reified Type that can be used in the conversion model.

**Returns:**
the reified type that describes the parameter's type.

---

**isAbsent**

`String isAbsent()`

The value to use when none of the parameter aliases are used in the command line for an option or a flag.

**Returns:**
The value to use when the parameter is not used in the command line, can be `null`.

---

**ifPresent**

`String ifPresent()`

The value to use for a flag when it is used in the command line. This method is undefined for an option.

**Returns:**
the parameter value for a flag.
Class ParsingException

```
java.lang.Object
    java.lang.Throwable
        java.lang.Exception
            java.lang.RuntimeException
                org.osgi.service.command.ParsingException
```

All Implemented Interfaces:
    Serializable

```java
class ParsingException
    extends RuntimeException
```

An exception that can point at the location where the error occurred.

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ParsingException(String message, int line, int column)</code></td>
<td>42</td>
</tr>
<tr>
<td>Constructor to create a syntax exception.</td>
<td></td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getColumn()</code></td>
<td>43</td>
</tr>
<tr>
<td>Get the column number where the parser failed.</td>
<td></td>
</tr>
<tr>
<td><code>getLine()</code></td>
<td>42</td>
</tr>
<tr>
<td>Get the line number where the parser failed.</td>
<td></td>
</tr>
</tbody>
</table>

**Constructor Detail**

**ParsingException**

```java
public ParsingException(String message, int line, int column)
```

Constructor to create a syntax exception.

**Parameters:**

- `message`: A human readable message
- `line`: The line number at which the parser failed to recognize the text
- `column`: The column at which the parser failed to recognize the text

**Method Detail**

**getLine**

```java
public int getLine()
```

Get the line number where the parser failed.

**Returns:**

The line # where the parser failed.
getColumn

public int getColumn()

Get the column number where the parser failed.

Returns:
The column # where the parser failed.
public interface Terminal

The Terminal interface describes a minimal terminal that can easily be mapped to command line editing tools. A Terminal is associated with an Input Stream and an Output Stream. The Input Stream represents the keyboard and the Output Stream the screen. A terminal does not block the input, each character is returned as it is typed, no buffering or line editing takes place, characters are also not echoed. However, the Input Stream is not restricted to bytes only, it can also return translated key strokes. Integers from 1000 are used for those. Not all keys have to be supported by an implementation. A number of functions is provided to move the cursor and erase characters/lines/screens. Any text outputed to the Output Stream is immediately added to the cursor position, which is then moved forwards. The control characters (LF,CR,TAB,BS) perform their normal actions. However lines do not wrap. Text typed that is longer than the window will not be visible, it is the responsibility of the sender to ensure this does not happen. A screen is considered to be getHeight() lines that each have getWidth() characters. For cursor positioning, the screen is assumed to be starting at 0,0 and increases its position from left to right and from top to bottom. Positioning outside the screen bounds is undefined.

### Nested Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static class</td>
<td></td>
</tr>
<tr>
<td>Terminal.Attribute</td>
<td>55</td>
</tr>
</tbody>
</table>

An inner class to provide an enum for the attributes

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int BREAK</td>
<td>47</td>
</tr>
<tr>
<td>Break key</td>
<td></td>
</tr>
<tr>
<td>int CONTROL_START</td>
<td>46</td>
</tr>
<tr>
<td>Start point of control characters.</td>
<td></td>
</tr>
<tr>
<td>int CURSOR_BACKWARD</td>
<td>47</td>
</tr>
<tr>
<td>Cursors backward key.</td>
<td></td>
</tr>
<tr>
<td>int CURSOR_DOWN</td>
<td>46</td>
</tr>
<tr>
<td>Cursor down key.</td>
<td></td>
</tr>
<tr>
<td>int CURSOR_FORWARD</td>
<td>46</td>
</tr>
<tr>
<td>Cursors forward key.</td>
<td></td>
</tr>
<tr>
<td>int CURSOR_UP</td>
<td>46</td>
</tr>
<tr>
<td>Cursor up key</td>
<td></td>
</tr>
<tr>
<td>int DELETE</td>
<td>47</td>
</tr>
<tr>
<td>Delete key</td>
<td></td>
</tr>
<tr>
<td>int END</td>
<td>47</td>
</tr>
<tr>
<td>End key</td>
<td></td>
</tr>
<tr>
<td>int F1</td>
<td>48</td>
</tr>
<tr>
<td>Function key 1</td>
<td></td>
</tr>
<tr>
<td>int F10</td>
<td>49</td>
</tr>
<tr>
<td>Function key 10</td>
<td></td>
</tr>
<tr>
<td>int F11</td>
<td>49</td>
</tr>
<tr>
<td>Function key 11</td>
<td></td>
</tr>
<tr>
<td>int F12</td>
<td>49</td>
</tr>
<tr>
<td>Function key 12</td>
<td></td>
</tr>
<tr>
<td>int F2</td>
<td>48</td>
</tr>
<tr>
<td>Function key 2</td>
<td></td>
</tr>
<tr>
<td>int F3</td>
<td>48</td>
</tr>
<tr>
<td>Function key 3</td>
<td></td>
</tr>
<tr>
<td>int F4</td>
<td>48</td>
</tr>
<tr>
<td>Function key 4</td>
<td></td>
</tr>
</tbody>
</table>
### Class Terminal.Attribute

<table>
<thead>
<tr>
<th>int</th>
<th>F5</th>
<th>Function key 5</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>F6</td>
<td>Function key 6</td>
<td>48</td>
</tr>
<tr>
<td>int</td>
<td>F7</td>
<td>Function key 7</td>
<td>49</td>
</tr>
<tr>
<td>int</td>
<td>F8</td>
<td>Function key 8</td>
<td>49</td>
</tr>
<tr>
<td>int</td>
<td>F9</td>
<td>Function key 9</td>
<td>49</td>
</tr>
<tr>
<td>int</td>
<td>FUNCTION_START</td>
<td>Helper</td>
<td>48</td>
</tr>
<tr>
<td>int</td>
<td>HOME</td>
<td>Home key</td>
<td>47</td>
</tr>
<tr>
<td>int</td>
<td>INSERT</td>
<td>Insert key</td>
<td>47</td>
</tr>
<tr>
<td>int</td>
<td>PAGE_DOWN</td>
<td>Page down key</td>
<td>47</td>
</tr>
<tr>
<td>int</td>
<td>PAGE_UP</td>
<td>Page up key</td>
<td>47</td>
</tr>
<tr>
<td>long</td>
<td>REPORTS_CURSOR_POS</td>
<td>Value for getCapabilities(), if set this Terminal can report the cursor position.</td>
<td>49</td>
</tr>
<tr>
<td>long</td>
<td>REPORTS_SIZE</td>
<td>Value for getCapabilities(), if set this Terminal can report the current size, see getHeight() and getWidth().</td>
<td>50</td>
</tr>
<tr>
<td>long</td>
<td>REPORTS_SIZE_CHANGES</td>
<td>Value for getCapabilities(), if set this Terminal sends a control code when the terminal changes size.</td>
<td>49</td>
</tr>
<tr>
<td>int</td>
<td>SIZE_CHANGE</td>
<td>The window size has changed.</td>
<td>48</td>
</tr>
<tr>
<td>long</td>
<td>SUPPORTS_ATTRIBUTES</td>
<td>Value for getCapabilities(), if set this Terminal supports attributes.</td>
<td>50</td>
</tr>
<tr>
<td>long</td>
<td>SUPPORTS_BIDIRECTIONAL_SCRIPTS</td>
<td>Value for getCapabilities(), if set this Terminal will handle bidirectional scripts.</td>
<td>50</td>
</tr>
<tr>
<td>long</td>
<td>SUPPORTS_CURSOR_POS</td>
<td>Value for getCapabilities(), if set this Terminal supports setting the cursor position.</td>
<td>50</td>
</tr>
<tr>
<td>long</td>
<td>SUPPORTS_Latin_1_SUPPLEMENT</td>
<td>Value for getCapabilities(), if set this Terminal supports the LATIN_1 supplement.</td>
<td>50</td>
</tr>
<tr>
<td>long</td>
<td>SUPPORTS_UNICODE</td>
<td>Value for getCapabilities(), if set this Terminal supports the full UNICODE set.</td>
<td>50</td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>attributes(Terminal.Attribute... attr)</code></td>
<td>Set the attributes of the text to outputed next.</td>
</tr>
<tr>
<td><code>backward()</code></td>
<td>Move the cursor backward.</td>
</tr>
<tr>
<td><code>clear()</code></td>
<td>Clear the complete screen and position the cursor at 0,0.</td>
</tr>
<tr>
<td><code>down()</code></td>
<td>Move the cursor down one line, this must not cause a scroll if the cursors moves off the screen.</td>
</tr>
<tr>
<td><code>eraseEndOfLine()</code></td>
<td>Leave the cursor where it is but clear the remainder of the line.</td>
</tr>
</tbody>
</table>
void forward()
Move the cursor forward.

long getCapabilities()
Answer the capabilities of this terminal.

GetWrite getErr()
Return the associated standard error stream.

int getHeight()
Return the actual height of the screen.

int getIn()
Get a character from the input.

GetWrite getOut()
Return the associated standard output stream.

int[] getPosition()
Return the current cursor position.

int getWidth()
Return the actual width of the screen.

boolean position(int x, int y)
Position the cursor on the screen.

String readLine()
Read a complete line from the input.

void up()
Move the cursor up one line, this must not cause a scroll if the cursor moves off the screen.

Field Detail

CONTROL_START
public static final int CONTROL_START = 65536
Start point of control characters.

CURSOR_UP
public static final int CURSOR_UP = 65536
Cursor up key.

CURSOR_DOWN
public static final int CURSOR_DOWN = 65537
Cursor down key.

CURSOR_FORWARD
public static final int CURSOR_FORWARD = 65538
Cursors forward key. Usually right.
CURSOR_BACKWARD

public static final int CURSOR_BACKWARD = 65539

Cursors backward key. Usually left.

PAGE_UP

public static final int PAGE_UP = 65540

Page up key

PAGE_DOWN

public static final int PAGE_DOWN = 65541

Page down key

HOME

public static final int HOME = 65542

Home key

END

public static final int END = 65543

End key

INSERT

public static final int INSERT = 65544

Insert key

DELETE

public static final int DELETE = 65545

Delete key

BREAK

public static final int BREAK = 65546

Break key
**SIZE_CHANGE**

```java
public static final int SIZE_CHANGE = 65547
```

The window size has changed.

**FUNCTION_START**

```java
public static final int FUNCTION_START = 65792
```

`Helper`

**F1**

```java
public static final int F1 = 65793
```

Function key 1

**F2**

```java
public static final int F2 = 65794
```

Function key 2

**F3**

```java
public static final int F3 = 65795
```

Function key 3

**F4**

```java
public static final int F4 = 65796
```

Function key 4

**F5**

```java
public static final int F5 = 65797
```

Function key 5

**F6**

```java
public static final int F6 = 65798
```

Function key 6
Class Terminal.Attribute

F7

```java
class Termianl.Attribute {
    public static final int F7 = 65799

    Function key 7
}
```

F8

```java
class Termianl.Attribute {
    public static final int F8 = 65800

    Function key 8
}
```

F9

```java
class Termianl.Attribute {
    public static final int F9 = 65801

    Function key 9
}
```

F10

```java
class Termianl.Attribute {
    public static final int F10 = 65802

    Function key 10
}
```

F11

```java
class Termianl.Attribute {
    public static final int F11 = 65803

    Function key 11
}
```

F12

```java
class Termianl.Attribute {
    public static final int F12 = 65804

    Function key 12
}
```

REPORTS_CURSOR_POS

```java
class Termianl.Attribute {
    public static final long REPORTS_CURSOR_POS = 1L

    Value for getCapabilities(), if set this Terminal can report the cursor position. See position(int, int).
}
```

REPORTS_SIZE_CHANGE

```java
class Termianl.Attribute {
    public static final long REPORTS_SIZE_CHANGE = 2L

    Value for getCapabilities(), if set this Terminal sends a control code when the terminal changes size. See SIZE_CHANGE.
}
```
REPORTS_SIZE

public static final long REPORTS_SIZE = 4L

Value for `getCapabilities()`, if set this Terminal can report the current size, see `getHeight()` and `getWidth()`.

SUPPORTS_BIDIRECTIONAL_SCRIPTS

public static final long SUPPORTS_BIDIRECTIONAL_SCRIPTS = 256L

Value for `getCapabilities()`, if set this Terminal will handle bidirectional scripts. If supported, text and cursor movements must be automatically reordered to match the visualization. This will allow users to just send Unicode strings where text is in increasing memory order. Any reordering is only done on the display.

SUPPORTS_ATTRIBUTES

public static final long SUPPORTS_ATTRIBUTES = 1024L

Value for `getCapabilities()`, if set this Terminal supports attributes. If not set, `attributes(Attribute...)` will always return null.

SUPPORTS_CURSOR_POS

public static final long SUPPORTS_CURSOR_POS = 16384L

Value for `getCapabilities()`, if set this Terminal supports setting the cursor position. If not set, `position(int, int)` will always return false and not set the cursor.

SUPPORTS_LATIN_1_SUPPLEMENT

public static final long SUPPORTS_LATIN_1_SUPPLEMENT = 65536L

Value for `getCapabilities()`, if set this Terminal supports the LATIN_1 supplement. These are the UNICODE 80-FF codes. A Terminal must minimally support US-ASCII.

SUPPORTS_UNICODE

public static final long SUPPORTS_UNICODE = 131072L

Value for `getCapabilities()`, if set this Terminal supports the full UNICODE set. This does not imply Bidirectional script handling. A Terminal must minimally support US-ASCII.

Method Detail

getIn

```
int getIn()
    throws Exception
```
Get a character from the input. Characters less than 0x10000 are Unicode characters, if more it is a control code defined by the constants in this class.

**Returns:**
the current Input Stream.

**Throws:**
Exception - When character cannot be read

---

**readLine**

String *readLine() throws Exception*

Read a complete line from the input. The result will not contain any command codes, just text. Implementers can allow line editing and history handling. The string must not contain the LF or CR at the end.

**Returns:**
a new line

**Throws:**
Exception

---

**getOut**

PrintWriter *getOut()*

Return the associated standard output stream.

**Returns:**
the associated standard output stream

---

**getErr**

PrintWriter *getErr()*

Return the associated standard error stream.

**Returns:**
the associated standard error stream

---

**clear**

void *clear() throws Exception*

Clear the complete screen and position the cursor at 0,0.

**Throws:**
Exception - when the method fails

---

**eraseEndOfLine**

void *eraseEndOfLine() throws Exception*
Leave the cursor where it is but clear the remainder of the line.

**Throws:**

- Exception - when the method fails

---

**up**

```java
void up()
    throws Exception
```

Move the cursor up one line, this must not cause a scroll if the cursor moves off the screen.

**Throws:**

- Exception - when the method fails

---

**down**

```java
void down()
    throws Exception
```

Move the cursor down one line, this must not cause a scroll if the cursor moves off the screen.

**Throws:**

- Exception - when the method fails

---

**backward**

```java
void backward()
    throws Exception
```

Move the cursor backward. Must not wrap to previous line.

**Throws:**

- Exception - when the method fails

---

**forward**

```java
void forward()
    throws Exception
```

Move the cursor forward. Must not wrap to next line if the cursor becomes higher than the width.

**Throws:**

- Exception - when the method fails

---

**getWidth**

```java
int getWidth()
    throws Exception
```

Return the actual width of the screen. Some screens can change their size and this method must return the actual width if possible. If the width cannot be established a -1 must be returned. If the size changes and the terminal supports reporting these events a **SIZE_CHANGE** key must be returned.
Class Terminal.Attribute

Returns:
the width of the screen or -1.

Throws:
Exception - when the method fails

getHeight

int get Höhe Method ( )
throws Exception

Return the actual height of the screen. Some screens can change their size and this method must return
the actual height if possible. If the width cannot be established a -1 must be returned. If the size changes
and the terminal supports reporting these events a SIZE_CHANGE key must be returned.

Returns:
the height of the screen or -1.

Throws:
Exception - when the method fails

generatePosition

int[] getGeneratePosition ( )
throws Exception

Return the current cursor position. The position is returned as an array of 2 elements. The first element is
the x position and the second elements is the y position. Both are zero based.

Returns:
the current position or null if it is not possible to establish the cursor position.

Throws:
Exception - when the method fails

setPosition

boolean position ( int x,
int y )
throws IllegalArgumentException,
Exception

Position the cursor on the screen. Positioning starts at 0,0 and the maximum value is given by getWidth ( ),
getHeight ( ). The visible cursor is moved to this position and text insertion will continue from that position.

Parameters:
x - The x position, must be from 0-width
y - The y position, must be from 0-height

Returns:
true if the position could be set, otherwise false

Throws:
IllegalArgumentException - when x or y is not in range
Exception - when this method fails

attributes

Terminal.Attribute[] attributes ( Terminal.Attribute... attr )
throws Exception

Set the attributes of the text to outputed next. The method returns the current settings, which can be used
to restore the display to the previous state. These current settings are stream based and not associated
with the position of the cursor. Attributes must be completely specified, they do not inherit from the current display. If attributes are specified multiple times, the last one wins.

**Parameters:**
- attr - A number of attributes describing the output

**Returns:**
- The previous state of attributes or null if attributes are not supported.

**Throws:**
- Exception - when this method fails

---

### getCapabilities

```java
long getCapabilities()
```

Answer the capabilities of this terminal. The following capabilities can be returned:

**Returns:**
- the bitmap of capabilities
## Class Terminal.Attribute

### org.osgi.service.command

java.lang.Object

Enclosing class: Terminal

public static class Terminal.Attribute extends Object

An inner class to provide an enum for the attributes

### Field Summary

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal.Attribute BACK_BLACK</td>
<td>Black</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_BLUE</td>
<td>Blue</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_CYAN</td>
<td>Cyan</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_GREEN</td>
<td>Green</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_MAGENTA</td>
<td>Magenta</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_NONE</td>
<td>No Color, transparent.</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_RED</td>
<td>Red</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_WHITE</td>
<td>White</td>
</tr>
<tr>
<td>Terminal.Attribute BACK_YELLOW</td>
<td>Yellow</td>
</tr>
<tr>
<td>Terminal.Attribute BOLD</td>
<td>Bolden the text.</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_BLACK</td>
<td>Black</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_BLUE</td>
<td>Blue</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_CYAN</td>
<td>Cyan</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_GREEN</td>
<td>Green</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_MAGENTA</td>
<td>Magenta</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_NONE</td>
<td>No Color, transparent.</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_RED</td>
<td>Red</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_WHITE</td>
<td>White</td>
</tr>
<tr>
<td>Terminal.Attribute FORE_YELLOW</td>
<td>Yellow</td>
</tr>
</tbody>
</table>
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>toString</td>
<td>String toString()</td>
</tr>
</tbody>
</table>

Field Detail

**UNDERLINE**

public Terminal.Attribute UNDERLINE

Underline the text.

**BOLD**

public Terminal.Attribute BOLD

Bolden the text.

**REVERSED**

public Terminal.Attribute REVERSED

Reverse the text.

**FORE_NONE**

public Terminal.Attribute FORE_NONE

No Color, transparent.

**FORE_BLACK**

public Terminal.Attribute FORE_BLACK

Black

**FORE_GREEN**

public Terminal.Attribute FORE_GREEN

Green
Package org.osgi.service.command.annotations

FORE_YELLOW
public Terminal.Attribute FORE_YELLOW

Yellow

FORE_MAGENTA
public Terminal.Attribute FORE_MAGENTA

Magenta

FORE_CYAN
public Terminal.Attribute FORE_CYAN

Cyan

FORE_BLUE
public Terminal.Attribute FORE_BLUE

Blue

FORE_RED
public Terminal.Attribute FORE_RED

Red

FORE_WHITE
public Terminal.Attribute FORE_WHITE

White

BACK_NONE
public Terminal.Attribute BACK_NONE

No Color, transparent.

BACK_BLACK
public Terminal.Attribute BACK_BLACK

Black
### Package org.osgi.service.command.annotations

#### BACK_GREEN

```java
public Terminal.Attribute BACK_GREEN

Green
```

#### BACK_YELLOW

```java
public Terminal.Attribute BACK_YELLOW

Yellow
```

#### BACK_MAGENTA

```java
public Terminal.Attribute BACK_MAGENTA

Magenta
```

#### BACK_CYAN

```java
public Terminal.Attribute BACK_CYAN

Cyan
```

#### BACK_BLUE

```java
public Terminal.Attribute BACK_BLUE

Blue
```

#### BACK_RED

```java
public Terminal.Attribute BACK_RED

Red
```

#### BACK_WHITE

```java
public Terminal.Attribute BACK_WHITE

White
```

### Method Detail

#### toString

```java
public String toString()

Overrides:

toString in class Object
```
Package org.osgi.service.command.annotations

Command Annotations Package Version 1.0.

See: Description

<table>
<thead>
<tr>
<th>Annotation Types Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>60</td>
</tr>
<tr>
<td>Provide a description for this element.</td>
<td></td>
</tr>
<tr>
<td><strong>Parameter</strong></td>
<td>61</td>
</tr>
<tr>
<td>A Parameter provides the information to treat a method parameter as a flag or option.</td>
<td></td>
</tr>
</tbody>
</table>

Package org.osgi.service.command.annotations Description

Command Annotations Package Version 1.0.

The purpose of this package is to provide access to the annotations for the command shell.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.service.command.annotations; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.service.command.annotations; version="[1.0,1.1)"
```
@Retention(value=RetentionPolicy.RUNTIME)
@Target(value={
    ElementType.TYPE,
    ElementType.METHOD,
    ElementType.PARAMETER
})
public @interface Description

Provide a description for this element. This is a generic annotation that can be used describe types, methods, and parameters. The information in this annotation can end up in the Command Descriptor. The usage of the Description is like:

Version:
$Id: c97161ad7adc294d6eb706e182af0b0d547d18538 $
A Parameter provides the information to treat a method parameter as a flag or option. Options and flags always start with a minus sign (\texttt{-}, \texttt{-}) in the command line. An option always is followed by a value while a flag goes without a value. The distinction is made with the \texttt{ifPresent()} method. If this method returns \texttt{NOT_SET} then this Parameter describes an option, otherwise it is a flag because a value is provided when the flag is used. If the same name aliases are used in other flags or options than the first one in the parameter declaration wins. Flags and options must be the first parameters in the method, it is not possible to intersperse the flags and options with parameters that have none. All remaining parameters without a Parameter description are unnamed.

Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{NOT_SET}</td>
<td>Magic value to provide a default for \texttt{ifAbsent()} and \texttt{ifPresent()} to indicate it is not set.</td>
</tr>
</tbody>
</table>

Required Element Summary

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{alias}</td>
<td>Parameter name and aliases.</td>
</tr>
<tr>
<td>\texttt{ifAbsent}</td>
<td>The value of the parameter if its name is not present on the command line.</td>
</tr>
<tr>
<td>\texttt{ifPresent}</td>
<td>The value if the flag or option is used in the command line.</td>
</tr>
</tbody>
</table>

Field Detail

\texttt{NOT\_SET}

\begin{verbatim}
public static final String \texttt{NOT\_SET} = "be3831f6ddfaa48ef1c0aba9e81c6251bf0f0ca"
\end{verbatim}

Magic value to provide a default for \texttt{ifAbsent()} and \texttt{ifPresent()} to indicate it is not set.

Element Detail

\texttt{alias}

\begin{verbatim}
public abstract String[] \texttt{alias}
\end{verbatim}

Parameter name and aliases. The shell will only be able to recognize names that start with a hyphen (\texttt{-}, \texttt{-}), however, other names are allowed. If the list of aliases is empty, For example, to support the options \texttt{-f/\--files}, return (\texttt{"-f"}, \texttt{"--files"}). If the alias contains an empty array then the Parameter can be provided by the session because it is a built in value like the Command Session object or the Terminal.

Returns:

parameter names.
ifPresent

public abstract String ifPresent

The value if the flag or option is used in the command line. If this returns NOT_SET then this is an option and the value after the alias must be used. This value must always be set for a flag.

Default: "be3831f6ddfba48efe1c0aba9e81c6251bf0f0ca"

Returns: The value to use when one of the flag names is present in the command line.

ifAbsent

public abstract String ifAbsent

The value of the parameter if its name is not present on the command line. This value is effectively the default value for the parameter. If this method returns NOT_SET then an appropriate default must be chosen that is negative. That is, 0, false, null.

Returns: default value of the parameter if its name is not present on the command line.

6 Alternatives

7 Security Considerations

 Obviously, a shell language provides ample opportunities for malice. In principle, anything in the system is accessible, just like from Java. The protection against malicious behavior is based up the Java 2 security model. This allows the shell and all commands to be ignorant of any security issues, unless they want to perform operations that they have access to but a potential user has not. Such code must be executed in a doPrivileged block.

The IO processors have the responsibility for protecting against malicious users.

The specification should copy some of the text of DMT Admin because it follows the same procedures
8 Document Support

8.1 References

8.1 Author’s Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Peter Kriens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>aQute</td>
</tr>
<tr>
<td>Address</td>
<td>9c, Avenue St. Drezery</td>
</tr>
<tr>
<td>Voice</td>
<td>+33 633982260</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:Peter.Kriens@aQute.biz">Peter.Kriens@aQute.biz</a></td>
</tr>
</tbody>
</table>

8.2 Acronyms and Abbreviations

8.3 End of Document
RFC 0157 Event Admin Update

Final

9 Pages

Abstract

A collection of minor updates to the Event Admin specification.
## 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
  - 3.1 Bug 926 ............................................................................................................................... 4
  - 3.2 Bug 1357 ............................................................................................................................. 4
  - 3.3 Bug 1587 ............................................................................................................................. 4
- 4 Requirements ............................................................................................................................... 5
- 5 Technical Solution ....................................................................................................................... 5
  - 5.1 containsProperty ................................................................................................................. 5
  - 5.2 Asynchronous Unordered Events ........................................................................................ 5
  - 5.3 EventProperties .................................................................................................................. 7
- 6 Considered Alternatives .............................................................................................................. 7
  - 6.1 Replace postEventUnordered with event.intents="unordered" ............................................ 7
  - 6.2 event.intents renamed to event.delivery .............................................................................. 8
- 7 Security Considerations .............................................................................................................. 8
- 8 Document Support ....................................................................................................................... 9
  - 8.1 References .......................................................................................................................... 9
  - 8.2 Author’s Address .................................................................................................................. 9
  - 8.3 Acronyms and Abbreviations ............................................................................................... 9
  - 8.4 End of Document .................................................................................................................. 9
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>05/03/10</td>
<td>Initial draft. Includes changes from bug 926, 1357, 1372 and 1587.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BJ Hargrave</td>
</tr>
<tr>
<td>2nd draft</td>
<td>05/03/10</td>
<td>I closed bug 1372 and am removing mention of it from this RFC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BJ Hargrave</td>
</tr>
<tr>
<td>3rd draft</td>
<td>05/17/10</td>
<td>Replaced postEventUnordered method with event.intents property on the EventHandler.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BJ Hargrave</td>
</tr>
<tr>
<td>4th draft</td>
<td>05/27/10</td>
<td>Renamed event.intents to event.delivery per CPEG call.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BJ Hargrave</td>
</tr>
<tr>
<td>Final</td>
<td>08/11/10</td>
<td>Marking final for voting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BJ Hargrave</td>
</tr>
</tbody>
</table>

1 Introduction

Event Admin is specified in section 113 of the OSGi Service Compendium specification. It has proven to be a popular service and several minor requested changes have been collecting in the OSGi bugzilla system. This document collects these changes into an RFC for review and discussion for updating Event Admin for Compendium version 4.3.
2 Application Domain

Event Admin specifies a means for an event source to send events to event listeners. Event sources can create events with a topic and properties and request Event Admin to deliver the events to event listeners which have declared interest in specific event topics and/or property values. The event source can request synchronous (and unordered) delivery or asynchronous (and ordered) delivery.

3 Problem Description

3.1 Bug 926
Event objects are specified to be immutable. Event object contain a map of the event properties including the event topic. For event sources which fire many events with the same event properties, this results in a event property map created and the map argument copied into that map for each event fired. A means is needed to eliminate map copying if the event source creates and fires many identical events.

3.2 Bug 1357
The Event class acts as a Map for Event properties. It can be constructed with a Map object which can have null values. Currently the Event class has no way for a user to distinguish between an unset key and a key set with a null value.

3.3 Bug 1587
Event Admin allows for events to be delivers synchronously and asynchronously. With synchronous delivery, delivery order cannot be guaranteed without fear of deadlock issues between threads. For asynchronous delivery, order is guaranteed by the specification. This is done by effectively serializing delivery of all events asynchronously fired. It is not easy to asynchronously deliver events without regard to delivery order.
4 Requirements

1. Provide a more performant means of creating and fire multiple events with the same event properties while preserving immutability of the Event object.

2. Allow an Event receiver to distinguish between the absence of an event property and an event property with a null value.

3. Allow events to be asynchronously delivered without regard to delivery order.

5 Technical Solution

The following changes to Event Admin will require updating the package version to 1.3. The API is also updated to compiler with -target jsr14 and use generics.

5.1 containsProperty

The following method is added to Event. This allows one to distinguish between a property not being present and a property with a null value.

public final boolean containsProperty(String name)

Indicate the presence of a property.

Parameters:
  name The name of the property.

Returns:
  true if a property with the specified name is in the event. This property may have a null value. false otherwise.

Since:
  1.3

5.2 Asynchronous Unordered Events

The following constants are added to EventConstants. This allows an Event Handler to indicate that it is willing to relax the event ordering requirements so that Event Admin can optimize delivery.
**String** EVENT_DELIVERY = "event.delivery"
Service Registration property specifying the delivery qualities requested by an Event Handler service.

Event handlers MAY be registered with this property. Each value of this property is a string specifying a delivery quality for the Event handler.

The value of this property must be of type String, String[], or Collection<String>.

Since: 1.3
See Also:
- DELIVERY_ASYNC_ORDERED
- DELIVERY_ASYNC_UNORDERED

**String** DELIVERY_ASYNC_ORDERED = "async.ordered"
Event Handler delivery quality value specifying the Event Handler requires asynchronously delivered events be delivered in order. Ordered delivery is the default for asynchronously delivered events.

This delivery quality value is mutually exclusive with DELIVERY_ASYNC_UNORDERED. However, if both this value and DELIVERY_ASYNC_UNORDERED are specified for an event handler, this value takes precedence.

Since: 1.3
See Also:
- EVENT_DELIVERY

**String** DELIVERY_ASYNC_UNORDERED = "async.unordered"
Event Handler delivery quality value specifying the Event Handler does not require asynchronously delivered events be delivered in order. This may allow an Event Admin implementation to optimize asynchronous event delivery by relaxing ordering requirements.

This delivery quality value is mutually exclusive with DELIVERY_ASYNC_ORDERED. However, if both this value and DELIVERY_ASYNC_ORDERED are specified for an event handler, DELIVERY_ASYNC_ORDERED takes precedence.

Since: 1.3
See Also:
- EVENT_DELIVERY
5.3 EventProperties

The following EventProperties class is added. The class provides an unmodifiable Map<String, Object> of event properties which can be used in multiple events without requiring copying of key/value pairs.

```java
public class EventProperties implements Map<String, Object> {
    The properties for an Event An event source can create an EventProperties object if it needs to reuse the same event properties for multiple events.

    The keys are all of type String. The values are of type Object. The key "event.topics" is ignored as event topics can only be set when an Event is constructed.

    Once constructed, an EventProperties object is unmodifiable. However, the values of the map used to construct an EventProperties object are still subject to modification as they are not deeply copied.
}
```

Since: 1.3

The Event class is modified to use the EventProperties type to hold properties supplied as Map or Dictionary arguments in a constructor. If the Map argument is an EventProperties argument, then the Event constructor will simply use the EventProperties argument directly rather than make a new one. This means event sources can directly construct EventProperties objects and use them when creating Event objects to avoid any performance overhead of the Event object constructor copying properties from the arguments.

6 Considered Alternatives

6.1 Replace postEventUnordered with event.intents="unordered"

The following method is added to EventAdmin. This method is like postEvent except that no event delivery ordering is guaranteed.

```java
void postEventUnordered(Event event) {
    Initiate asynchronous, unordered delivery of an event. This method returns to the caller before delivery of the event is completed. Events received by this method may be delivered in any order.
}
```

Parameters:

- `event` The event to send to all listeners which subscribe to the topic of the event.

Throws:
SecurityException – If the caller does not have TopicPermission[topic,PUBLISH] for the topic specified in the event.

Since:

1.3

A trivial implementation of this method can be to call postEvent. However a more sophisticated implementation could use an executor with multiple threads.

6.2 event.intents renamed to event.delivery

We use event.intents to allow for future additional intents to be defined.

String EVENT_INTENTS = "event.intents"

Service Registration property specifying the intents of an Event Handler service.

Event handlers MAY be registered with this property. Each value of this property is a string specifying an intent of the Event handler.

The value of this property must be of type String, String[], or Collection<String>.

Since:

1.3

See Also:

EVENT_INTENT_UNORDERED

String EVENT_INTENT_UNORDERED = "unordered"

Event Handler intent value specifying the Event Handler does not require events be delivered in order. This may allow an Event Admin implementation to optimize event deliver by relaxing ordering requirements.

Since:

1.3

See Also:

EVENT_INTENTS

7 Security Considerations

No new security considerations.
8 Document Support

8.1 References

8.2 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>

8.3 Acronyms and Abbreviations

8.4 End of Document
Abstract

A Coordinator service makes it possible to coordinate activities between independent parties. A Coordinator can start a coordination and participants can participate by being called back at the end of the Coordination. Coordinations can be pushed on a thread local stack to act as an implicit parameter for participants that are called downstream, or can be passed explicitly via the Participant API.
## 0 Document Information

### Table of Contents

- **0 Document Information** ................................................................................................................. 2
- **Table of Contents** .............................................................................................................................. 2
- **Terminology and Document Conventions** ....................................................................................... 3
- **Revision History** ............................................................................................................................... 3
- **1 Introduction** .................................................................................................................................. 3
- **2 Application Domain** ....................................................................................................................... 4
- **3 Problem Description** ....................................................................................................................... 4
- **4 Requirements** ................................................................................................................................ 5
- **5 Use Cases** ..................................................................................................................................... 6
  - **Configuration Admin** ..................................................................................................................... 6
  - **Blueprint Configuration Admin Support** ........................................................................................ 6
- **6 Technical Solution** ....................................................................................................................... 6
  - **The Coordinator Service** ............................................................................................................. 6
    - **Normal Case** ............................................................................................................................. 7
    - **Timeout** .................................................................................................................................... 7
    - **Participants** .............................................................................................................................. 8
    - **Blocking** .................................................................................................................................. 8
    - **Failing** ...................................................................................................................................... 10
    - **Thread Safety** ........................................................................................................................... 11
    - **Coordination Exception** ........................................................................................................... 11
- **7 Command Line** ............................................................................................................................... 12
- **8 JMX API** ....................................................................................................................................... 12
- **9 Javadoc** ...................................................................................................................................... 12
- **10 Considered Alternatives** ............................................................................................................ 13
  - **Prepare phase** ............................................................................................................................. 13
  - **No Asynchronous Failure** ............................................................................................................ 13
- **11 Security Considerations** ............................................................................................................. 14
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 .

Source code is shown in this typeface.

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>06/22/10</td>
<td>Initial Release</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Peter Kriens, aQute</td>
</tr>
<tr>
<td>0.1</td>
<td>06/28/10</td>
<td>Additional use-case and comments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>David Bosschaert, Red Hat</td>
</tr>
<tr>
<td>0.2</td>
<td>08/25/10</td>
<td>Comments from Ottawa meeting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Coordinations can be nested, they no longer automatically join the current Coordination</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Coordinations can be created without a thread local association</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Coordinations can be pushed/popped on a thread local stack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- There is now the concept of a current Coordination</td>
</tr>
<tr>
<td>0.3</td>
<td>08/30/10</td>
<td>Revisions for non-blocking/explicit API use case</td>
</tr>
</tbody>
</table>
1 Introduction

During the work on R4.3 it was found that we needed a mechanism to coordinate activities that were taking place on the same thread but that were not directly in touch with each other. The key example was configuration admin and the extension to support multiple PIDs on a MS. The MS would not know when the last PID was set and would therefore likely have to do too much work by updating its internal configuration for each PID that was set.

This RFC introduces a lightweight API to coordinate activities to address the need for local coordination between services. This API is not a replacement for transactions because it does not support any persistence capabilities.

2 Application Domain

In a service oriented programming model it is hard to know what parties will participate in the execution of some function. In principle, a single call to a service can result in many other calls. Services are implemented by more or less independent actors. Java software calls methods on the same thread, this means that there is a thread context between all methods that are being called. The ThreadLocal class creates a variable space that is based on the thread.

Parties that need to coordinate their activities must establish a shared protocol. For example, Conditional Permission Admin creates an Updater object so that all updates being made can be treated atomically. This is a common pattern: Party A makes repeated updates and then “commits” all the changes in one go.

Transactions support a bracketing model where an initiator can bracket a certain amount of work, ensuring that participants have the guarantee that they're called back at the end of the transaction. In JTA, this model was even opened to additional parties with the Synchronization Registry.

The model discussed in this RFC has the following structure. A task is a piece of work that needs to be performed in a system. The task is begun by an initiator. The initiator begins the work and calls other parties. The context for the work can be passed as a specific parameter or it can be stored in a thread local stack. A party can decide to participate in the coordination or it can ignore it, participation is optional. A party can also decide to fail the work by throwing an exception. When the parties return from the work they've done the initiator can clean up.
3 Problem Description

Though transactions can be used for coordination, they carry a rather heavy overhead. To support the ACID characteristics, it is necessary to know the resource managers before the system is started to be able to recover any transactions that were in limbo. Additionally, the overhead of a typically distributed model as well as the complexity of the XAResource API makes transactions a very heavy weight solution.

The problem this RFC addresses is therefore that there is currently no lightweight way to coordinate the execution of a task, where participants can see when they are required to participate as well as when the task fails or is finally terminated. In short, this RFC needs a coordination concept that allows multiple parties to coordinate the outcome of a task.

4 Requirements

- CO0001 – Provide a solution to allow multiple parties that collaborate on the same thread to coordinate the outcome of a task initiated by an initiator.
- CO0002 – An initiator must be able to initiate a coordination and control the final outcome.
- CO0003 – Any party participating in the task must be able to make the coordination always fail when ended.
- CO0004 – Participants on the task must be informed when the coordination is terminated.
- CO0005 – Participants must know that the coordination succeeded or failed at termination.
- CO0006 – A coordination must have a timeout with a default time that is more than 30s.
- CO0007 – The timeout timer must be settable by the initiator.
- CO0008 – It must be possible to enumerate the participants.
- CO0009 – A participant must automatically be blocked when it attempts to participate on multiple coordinations, unless it indicates it is reentrant.
- CO0010 – Participants must be able to prevent blocking.
- CO0010 – Coordinations must be allowed to nest.
• CO0011 – The initiator must be informed if one of the participants throw an exception in the termination method for an ok outcome.

• CO0012 – It must be possible to enumerate the active coordinations

• CO0013 – It must be possible to fail a coordination from outside of the initiating/active thread and ensure that the active thread (and the coordinated task) is interrupted.

• CO0014 – Provide an identifying mechanism for coordinations

• CO0015 – Provide a simple way to either initiate a coordination or participate in an existing coordination.

• CO0016 – Allow parties that make the coordination fail to provide information why it failed, the rationale for this requirement for trouble shooting

• CO0017 – A Coordination must have a unique identification for management purposes

• CO0018 – Provide an optional command line interface specification.

• CO0019 – Provide an optional JMX based management interface.

• CO0020 – All coordinations can be passed as parameters in method call

5 Use Cases

Configuration Admin

Configuration Admin allows Managed Service (Factory) services to use multiple PIDs, this is a recent change. However, it turns out that these PIDs will update the MS and MSF independently from each other, creating race conditions and inconsistencies. It is very hard to know for an MS or MSF that it has gotten the last update of a set. It can wait until all its PIDs are set but there are many scenarios where a new update can interfere.

There is therefore a need for a management agent to bracket a number of updates to Configuration Admin. Updates to the MS and MSF must then wait until the bracket is closed. That is, at the time that all configurations are persisted.

Remote Service Admin

RSA manages the topology, the set of services mapping to external systems. Changing the topology can have a serious impact on the services, and those registrations and unregistrations can affect many different subsystems. In the current model a subsystem has no possibility to delay work until the topology has been completely changed, it is forced to do everything immediate. Though a delay can improve the situations it is much more complicated and provides no guarantees.
Deployment Admin

Deployment Admin takes a JAR file (a Deployment Pack) and from this DP it extracts bundles that it installs as well as resources, where the resources are processed through Resource Processors. A large number of services can be involved in installing and uninstalling a DP. Currently, all these activities are not bracketed, forcing each party to process the activities immediately.

Blueprint Configuration Admin Support

Like the previous use-case this one is also related to Configuration Admin, but at a slightly different level.

Blueprint Configuration Admin support is worked on in RFC 156. One problem with configuration admin support for Blueprint is to marry the Blueprint style bean setter approach with the bracketing required to apply configuration changes at the right time.

As an example take a bean that controls a thread pool in a system. This thread-pool has a high-watermark and a low-watermark. These watermarks are set through the Configuration Admin Service and injected into the bean using \texttt{setLowWaterMark()} and \texttt{setHighWaterMark()} injection methods. It is clearly important for the bean to know when the setting of configuration changes is finished so that it can apply the new thread pool values together as applying these new values individually would be highly inefficient or possibly even incorrect.

A coordinator service could help in this situation making a callback to a \texttt{configUpdateFinished()} or similar method once all the configuration changes had been set in the bean.

6 Technical Solution

The Coordinator Service

This RFC provides a Coordinator service. A Coordinator service allows an initiator to begin a Coordination. The initiator can now call other services. The Coordination can be passed as a parameter or it can be associated with the current thread. Each thread will have a thread local stack for nested Coordinations.

These downstream services can decide to participate on this coordination by registering themselves with the Coordination object, or the Coordinator service for convenience. When the initiator is finished, it calls the end() method on the Coordination. The end() method will terminate the Coordination by calling all registered participants to inform them that the Coordination has ended. Coordinations can fail, in that case the participants will be informed of the negative outcome and the end() method will return a code indicating why there was a failure. The basic model is depicted in the next schema, where the blank services represent arbitrary domain services.
Normal Case

After the Coordination is begun, the initiator can perform work. It is best practice to encapsulate the work with an exception handler that ensure the coordination is always properly terminated. For example:

```java
Coordination c = coordinator.begin("name");
try {
    work1();
    work2();
    if (c != null) {
        // we're the initiator!
        if (c.end() != OK)
            ... // error case
    }
} catch (SomeException e) {
    ...
} finally {
    c.terminate();
}
```

Any party calling begin(String) should call terminate() on the result of the Coordinator.begin method. When begin() is called, it is possible that there is already a Coordination going on. This Coordination will be pushed on the thread local stack.

Timeout

If a Coordination is not properly terminated (either end() or fail(Throwable) is not called) it will time out. The time out must be at least 30 seconds by default but it is allowed be overridden by the initiator with the addTimeout method:
Coordination c = coordinator.begin("hello");
try {
    coordinator.addTimeout( 5000 ); // 5 secs
}

Implementations can allow the default timeout of at least 30 secs to be overridden by configuration. The reason that at least 30 secs limit is given is to allow for out-of-the-box testing with a known timeout.

If a timeout happens, the coordination thread is set to always fail and interrupted and a small delay happens to allow the initiator to clean up by reacting on the interrupt. If the initiator can do this clean up normally, the end() method will return TIMEOUT. After this delay the coordination is forcefully failed.

### Participants

The common pattern for a participant is to participate if a current Coordination is present and to do the work immediately when there is no Coordination active. To minimize the amount of code for this common pattern best practice is to use the following pattern:

```java
void foo() {
    if ( coordinator.participate(this))
        delayWork();
    else
        immediateWork();
}
```

That is, the participate method returns a boolean that indicates if there was a current Coordination to participate with (true) or if it could not participate because there was no current coordination (false).

It is also possible to participate in a Coordination that is passed as a parameter. In this case, the Participant is invoked indirectly by the initiator (perhaps via retrieval of some filtered set of Participants registered in the service registry). In the example below, the Participant is invoked generically, and elects to participate in the coordination if a particular context object is present in the provided coordination.

```java
void participate(Coordination c) {
    CustomObject context = c.getVariable(this.getClass(), CustomObject.class);
    if ( context != null ) {
        c.participate(this); // convenience: register with coordination directly
        delayWork(context);
    } else {
        immediateWork();
    }
}
```

A participant can participate in the same Coordination multiple times but will only be called back once when the Coordination is terminated. The participate method must block if the participant object is already participating in another Coordination, unless the participant indicates it is reentrant. For non-reentrant participants, this check must be based on object identity, not equals.

The Coordinator may throw a CoordinationException from the participate() method when the participant lock times out or a deadlock is detected.

### Blocking

Participants can be written in a reentrant manner if they store all collaboration-specific state in the collaboration object and exercise all of the care required to support processing multiple thread-collaboration pairs at the same time. The collaborator service will ensure that a collaboration is active on only one thread at a time, enabling
Participants to be written for a simpler model: any non-reentrant participant should participate in only one Coordination at a time. The Coordinator will block on the participate() method if a non-reentrant participant is active in a different coordination. If a participant never wants to block, it must always register a unique object for each Coordination. Such an object is then not shared between Coordinations and can not cause blocking.

Because a participant is blocked, many solutions are quite simple to implement because access to the participants is automatically serialized. For example, the following code shows a Managed Service that merges its dictionary for each call within a Coordination. Because all calls to the Managed Service are serialized (well, only the ones inside any Coordination) a field can be used to keep the merged dictionary. This field can then be shared between different threads.

```java
volatile Dictionary merged = new Dictionary();

public void updated(Dictionary dict) {
    if (coordinator.participate(this))
        merge(dict);
    else
        direct(dict);
}

public void failed(Coordination c) {
    merged = new Dictionary(); // discard merges
}

public void ended(Coordination c) {
    direct(merged);
    merged = new Dictionary();
}
```

However, coarse grained locking is very prone to deadlocks if the locked resources are not carefully ordered. The granularity can therefore be chosen by the participants. A good example is the use of the Configuration Admin and the Coordination API. The CM can participate in different ways:

1. Only participate in a single Coordination at a time. This is very safe and very simple (the CM can act as participant) but deadlock prone. This model was shown in the previous example.

2. Create a participant per thread, implies that different updates to the same PIDs can occur and thereby throwing out any coordination scheme.

3. Create a participant per update. This is the least deadlock prone method but provides no safety.

4. Create a participant per PID. Once a PID is used it is locked until the end of the Coordination. Locking on the PID is likely the best solution.

The following code snippets elucidate the different approaches. It assumes the the CM implementation calls a schedule method with a Runnable, like:

```java
public class Scheduler {
    void schedule(String pid, Runnable r) { ... }
}
```

For case 1 (serialized Coordinations), we can lock on the scheduler.

```java
public class Scheduler {
    List<Runnable> queue = new ArrayList<Runnable>();

    void schedule(String pid, Runnable r) {
        if (coordinator.participate(this)) {
            queue.add(r);
        }
    }
```
The previous example only allows a single Coordination to use the CM. Using thread locals it is possible to create a more fine grained lock.

```java
ThreadLocal queues = new ThreadLocal();
class Queuer implements Participant {
    List<Runnable> queue = new ArrayList<Runnable>();

    public void ended(Coordination c) {
        for (Runnable r : queue )
            executor.execute(r);
    }
    public void failed(Coordination c) {
        queue.clear();
    }

    void schedule( String pid, Runnable r ) {
        if ( coordinator.getCurrentCoordination() != null ) {
            Queuer queuer = queues.get();
            if ( queuer == null ) {
                queuer = new Queuer();
                queues.set(queuer); // never blocks because new object
            }
            queuer.add(r);
        } else
            executor.execute(r);
    }
}

Though this model poses significantly less contention (and thus change for deadlock), it allows multiple threads to overlap their pid updates. The following example shows how the PID can be used to lock:

```java
final Map<String,PidQueuer> queues = new WeakHashMap<String,PidQueuer>();
class PidQueuer implements Participant {
    List<Runnable> queue = new ArrayList<Runnable>();

    public void ended(Coordination c) throws Exception {
        queue.clear();
        for ( Runnable r : queue )
            executor.execute(r);
    }

    public synchronized void failed(Coordination c) throws Exception {
        queue.clear();
    }
}

void schedule(String pid, Runnable runnable) {
    Coordinator coordinator = getCoordinator();
}
if (coordinator != null && coordinator.isActive()) {
    PidQueuer pq;
    synchronized (queues) {
        pq = queues.get(pid);
        if (pq == null) {
            pq = new PidQueuer();
            queues.put(pid, pq);
        }
        assert coordinator.participate(pq); // lock
        assert pq.queue.isEmpty();
        pq.queue.add(runnable);
    }
    else
        executor.execute(runnable);
}

The Coordination API is specifically designed to enable these kind of locking mechanisms without requiring a lot of effort from the programmer.

Failing

Any party can fail an ongoing Coordination. The Coordinator service has a alwaysFail(Throwable) method that allows work to continue on the active Coordination but ensure that the outcome is always fail. The fail state can be checked with isFailed(). Setting this state when the Coordination is terminated has no effect. If the Coordination is set to always fail, the initiator will receive FAILED back from the end() method and all participants will be called on their failed(Coordination) method during termination.

The initiator can also fail the Coordination by calling fail(Throwable). Due to race conditions, this method returns a true if the Coordination was actually terminated because of that call or if the Coordination had already been terminated. If had already been terminated, the fail method is a noop.

A Coordination can also fall because a of a time out or a management agent that calls the fail(Throwable) method on the Coordination on another thread. Such asynchronous failing must actually terminate the Coordination. An asynchronous fail mean that the failed() methods of the Participant objects is called on another thread. Participants must properly synchronize their fields to allow this asynchronous behavior.

The different fail methods provide an Throwable parameter so that the party failing can indicate why the Coordination failed to assist debugging.

Current Coordination Support

The Coordinator supports a thread local stack for Coordinations. The top of this stack is the current Coordination. The Coordinator.begin(String) method creates a Coordination and pushes it on the top of the stack. The getCurrentCoordination() method returns the current Coordination or null. Coordinations can be pushed and popped.

It is up to the actual scenarios to define how to use the Coordinator. The most common case will be to use the begin method and the concept of a current Coordination. This model does not require API changes. However, in certain cases it is better to pass the Coordination around as a parameter in a call or event object. The Coordinator API allows both patterns.
Thread Safety

A Coordination can timeout asynchronously and a management agent could asynchronously fail a Coordination. It is even possible that the ended() method is called on another thread than the initiator. An implementation must therefore provide the following guarantees:

- All objects, including the participants must be thread safe, all methods can be called from any thread.
- Either the Participant.ended(Coordination) or Participant.failed(Coordination) method is called once.
- An exception must be thrown when the end() method is called more than once.
- The Coordination.fail(Throwable) method can be called any number of times from any thread but can only have effect when the Coordination has not been terminated. fail() must not block if a Coordination is in the process of being terminated.
- Termination (via end or fail(Throwable)) must be atomic.
- It must be guaranteed that if participate(Participant) returns true that the participant is called back on either failed(Coordination) or ended(Coordination). Notice that either method can already be called back before the participate method has returned due to asynchronous termination.
- A non-reentrant Participant must only be locked on a Coordination as long as the encompassing Coordination is not terminated.

Coordination Exception

There are several places in the code where the Coordinator can throw an exception to indicate the Coordination failed. The Coordination Exception is a Runtime Exception because it is only intended to be caught by the initiator. Participants must make sure they properly clean up when this exception is thrown but should allow it to propagate. The Coordination Exception provides the following reasons:

1. TIMEOUT – The locking of the participating timed out.
2. DEADLOCK – The Coordinator detected a potential deadlock
3. UNKNOWN – An unknown reason

7 Command Line

A Coordinator can optionally register a command provider for the TSH. The following commands should be implemented by a Coordinator implementation:

```
crd list [ -f, --full ] [ <regex filter on name> ]
crd fail [ -r, --reason <reason> ] [-b, --bundle <bundle>] <coordination> ...
```
A Coordinator must provide a converter to a Coordination based on the following inputs:

- id – the Coordination id
- name – the Coordination name. Must be unique
- bundle – Must translate to all coordinations of a specific bundle

## 8 JMX API

The Coordinator has a Companion Mbean: CoordinatorMBean. This Mbean has the following methods:

1. TabularData istCoordinations( String regexFilter )
2. CompositeData getCoordination( long id )
3. boolean fail(long id, String reason)
4. void setTimeout(long id, long timeout)

The identity of the Coordination must be maintained by the Mbean. Each id must be mappable to the Coordination object. This id is not part of the coordination. Further details can be found in the Javadoc.

## 9 Javadoc

<table>
<thead>
<tr>
<th>OSGi Javadoc</th>
<th>8/31/10 12:13 PM</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Package Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.osgi.jmx.service.coordination</td>
<td>OSGi JMX Coordination Package Version 1.0.</td>
</tr>
<tr>
<td>org.osgi.service.coordination</td>
<td>Coordination Package Version 1.0.</td>
</tr>
</tbody>
</table>
Package org.osgi.jmx.service.coordination

OSGi JMX Coordination Package Version 1.0.

See:

Description

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoordinatorMBean</td>
<td>16</td>
</tr>
</tbody>
</table>

This MBean provides the management interface to the OSGi Coordinator Service

---

### Package org.osgi.jmx.service.coordination Description

OSGi JMX Coordination Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.jmx.service.coordination; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.jmx.service.coordination; version="[1.0,1.1)"
```
public interface CoordinatorMBean

This MBean provides the management interface to the OSGi Coordinator Service

Version:
$Id: cd559be4903b74181ee27ec6aaa36200616421d$

ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompositeType</td>
<td>COMPARISON, TYPE</td>
</tr>
<tr>
<td>ArrayType</td>
<td>COORDINATIONS, TYPE</td>
</tr>
<tr>
<td>Description</td>
<td>Defines a list of COORDINATION, TYPE</td>
</tr>
<tr>
<td>String</td>
<td>ID</td>
</tr>
<tr>
<td>Description</td>
<td>The key ID, used in ID, ITEM.</td>
</tr>
<tr>
<td>org.osgi.jmx.Item</td>
<td>ID, ITEM</td>
</tr>
<tr>
<td>Description</td>
<td>The item for the ID of an Coordination object.</td>
</tr>
<tr>
<td>String</td>
<td>NAME</td>
</tr>
<tr>
<td>Description</td>
<td>The key NAME, used in NAME, ITEM.</td>
</tr>
<tr>
<td>org.osgi.jmx.Item</td>
<td>NAME, ITEM</td>
</tr>
<tr>
<td>Description</td>
<td>The item for the user name for an authorization object.</td>
</tr>
<tr>
<td>String</td>
<td>OBJECTNAME</td>
</tr>
<tr>
<td>Description</td>
<td>User Admin MBean object name.</td>
</tr>
<tr>
<td>String</td>
<td>TIMEOUT</td>
</tr>
<tr>
<td>Description</td>
<td>The key TIMEOUT, used in TIMEOUT, ITEM.</td>
</tr>
<tr>
<td>org.osgi.jmx.Item</td>
<td>TIMEOUT, ITEM</td>
</tr>
<tr>
<td>Description</td>
<td>The item for the ID of an Coordination object.</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void addTimeout (long id, long timeout)</td>
<td>Set/Change the timeout of a Coordination.</td>
</tr>
<tr>
<td>boolean fail (long id, String reason)</td>
<td>Fail a Coordination.</td>
</tr>
<tr>
<td>CompositeData getCoordination (long id)</td>
<td>Get a Coordination.</td>
</tr>
<tr>
<td>TabularData listCoordinations (String regexFilter)</td>
<td>List the current coordinations.</td>
</tr>
</tbody>
</table>

Field Detail

OBJECTNAME

public static final String OBJECTNAME = "osgi.compendium:service=coordinator,version=1.0"
Interface Coordination

User Admin MBean object name.

NAME

public static final String NAME = "Name"

The key NAME, used in NAME_ITEM.

NAME_ITEM

public static final org.osgi.jmx.Item NAME_ITEM

The item for the user name for an authorization object. The key is NAME and the type is SimpleType.STRING.

ID

public static final String ID = "Id"

The key ID, used in ID_ITEM.

ID_ITEM

public static final org.osgi.jmx.Item ID_ITEM

The item for the id of an Coordination object. The key is ID and the type is SimpleType.LONG. The id must be generated by the Mbean and map to a unique Coordination (which should no be pinned in memory because of this).

TIMEOUT

public static final String TIMEOUT = "Id"

The key TIMEOUT, used in TIMEOUT_ITEM.

TIMEOUT_ITEM

public static final org.osgi.jmx.Item TIMEOUT_ITEM

The item for the id of an Coordination object. The key is ID and the type is SimpleType.LONG.

COORDINATION_TYPE

public static final CompositeType COORDINATION_TYPE
COORDINATIONS_TYPE

public static final ArrayType COORDINATIONS_TYPE

Defines a list of COORDINATION_TYPE

Method Detail

listCoordinations

TabularData listCoordinations(String regexFilter)
throws IOException

List the current coordinations. The Composite Data is typed by COORDINATIONS_TYPE.

Parameters:
regexFilter - a regular expression filter on the coordination name

Returns:
the Coordinations typed by COORDINATIONS_TYPE.

Throws:
IOException - if the operation fails

getCoordination

CompositeData getCoordination(long id)
throws IOException

Get a Coordination. The Composite Data is typed by COORDINATION_TYPE.

Parameters:
id - The id of a Coordination

Returns:
the Coordinations typed by COORDINATION_TYPE.

Throws:
IOException - if the operation fails

fail

boolean fail(long id,
String reason)
throws IOException

Fail a Coordination.

Parameters:
id - The id of the coordination to be failed.
reason - The reason the coordination should be failed. The implementation of the MBean should create a Throwable/Exception with this reason in order to fail the specified Coordination

Returns:
true if the coordination was failed by this call, otherwise false.

Throws:
IOException

See Also:
Coordination#fail(Throwable)
addTimeout

void addTimeout(long id, long timeout)
throws IOException

Set/Change the timeout of a Coordination.

Parameters:
   id - The id of the Coordination
   timeout - The nr of milliseconds for the next timeout.

Throws:
   IOException
Package org.osgi.service.coordination

Coordination Package Version 1.0.

See:

Description

## Interface Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination</td>
<td>A Coordination object is used to coordinate a number of independent participants.</td>
<td>21</td>
</tr>
<tr>
<td>Coordinator</td>
<td>A Coordinator service provides a facility to coordinate activities between different parties.</td>
<td>30</td>
</tr>
<tr>
<td>Participant</td>
<td>A Participant participates in a Coordination.</td>
<td>34</td>
</tr>
</tbody>
</table>

## Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoordinationP</td>
<td>The name parameter of the Permission is a filter expression.</td>
<td>28</td>
</tr>
<tr>
<td>ermission</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Exception Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoordinationE</td>
<td>Thrown when an implementation detects a potential deadlock situation that it cannot solve.</td>
<td>26</td>
</tr>
<tr>
<td>xception</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Package org.osgi.service.coordination Description

Coordination Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

Import-Package: org.osgi.service.coordination; version="[1.0,2.0)"

Example import for providers implementing the API in this package:

Import-Package: org.osgi.service.coordination; version="[1.0,1.1)"
public interface Coordination

A Coordination object is used to coordinate a number of independent participants. Once a Coordination is created, it can be used to add Participant objects. When the Coordination is ended, the participants are called back. A Coordination can also fail for various reasons, in that case the participants are informed of this failure.

ThreadSafe

<table>
<thead>
<tr>
<th>Field Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int FAILED</td>
<td>22</td>
</tr>
<tr>
<td>int OK</td>
<td>21</td>
</tr>
<tr>
<td>int PARTIALLY_ENDED</td>
<td>22</td>
</tr>
<tr>
<td>int TIMEOUT</td>
<td>22</td>
</tr>
</tbody>
</table>

Field Detail

OK

public static final int OK = 0
Return value of `end()`. The Coordination ended normally, no participant threw an exception.

### PARTIALLY_ENDED

```java
class Coordination {
    public static final int PARTIALLY_ENDED = 1;
}
```

Return value of `end()`. The Coordination did not end normally, a participant threw an exception making the outcome unclear.

### FAILED

```java
class Coordination {
    public static final int FAILED = 2;
}
```

Return value of `end()`. The Coordination was set to always fail (`fail(Throwable)`).

### TIMEOUT

```java
class Coordination {
    public static final int TIMEOUT = 3;
}
```

Return value of `end()`. The Coordination failed because it had timed out.

### Method Detail

#### getName

```java
public String getName()
```

Return the name of this Coordination. The name is given in the `Coordinator.begin(String)` or `Coordinator.create(String)` method.

**Returns:**

the name of this Coordination

#### fail

```java
public boolean fail(Throwable reason)
```

Fail and then end this Coordination while returning the outcome. Any participants will be called on their `Participant.failed(Coordination)` method. Participants must assume that the Coordination failed and should discard and cleanup any work that was processed during this Coordination. The fail method must terminate the current Coordination before any of the failed methods are called. That is, the `Participant.failed(Coordination)` methods must be running outside the current coordination, no participants can be added during the termination phase. A fail method must return silently when the Coordination has already finished.

**Parameters:**

reason - Throwable describing the reason of the failure for documentation

**Returns:**

true if the Coordination was still active, otherwise false
**terminate**

`boolean terminate()`

If the Coordination is terminated then return, otherwise set the Coordination to fail. This method enables the following fail-safe pattern to ensure Coordinations are properly terminated.

```java
Coordination c = coordinator.begin("show_fail");
try {
    work1();
    work2();
    if (end() != OK)
        log("...");
} catch (SomeException e) {
    ...
} finally {
    c.terminate();
}
```

With this pattern, it is easy to ensure that the coordination is always terminated.

**Returns:**
- `true` if this method actually terminated the coordination (that is, it was not properly ended).
- `false` if the Coordination was already properly terminate by an `end()` or `fail(Throwable)` method.

**end**

`int end()`

Throws `IllegalStateException`

End the current Coordination. Any participants will be called on their `Participant.ended(Coordination)` method. This `end()` method indicates that the Coordination has properly terminated and any participants should The end method must terminate the current Coordination before any of the `Participant.ended(Coordination)` methods is called. That is, the `Participant.ended(Coordination)` methods must be running outside the current coordination, no participants can be added during the termination phase. This method returns the outcome of the Coordination:

1. `OK` - Correct outcome, no exceptions thrown
2. `PARTIALLY_ENDED` - One of the participants threw an exception
3. `FAILED` - The Coordination was set to always fail

**Returns:**
- `OK`, `PARTIALLY_ENDED`, `FAILED`

**Throws:**
- `IllegalStateException` - when the Coordination is already terminated.

**getParticipants**

`Collection<Participant> getParticipants()`

Return the current list of participants that joined the Coordination. This list is only valid as long as the Coordination has not been terminated. That is, after `end()` or `fail(Throwable)` is called this method will return an empty list.
isFailed

boolean isFailed()

Returns:
true if this Coordination has failed, false otherwise.

addTimeout

void addTimeout(long timeOutInMs)

Add a minimum timeout for this Coordination. If this timeout expires, then the Coordination will fail and the
initiating thread will be interrupted. This method must only be called on an active Coordination, that is,
before end() or fail(Throwable) is called. If the current deadline is arriving later than the given timeout
then the timeout is ignored.

Parameters:
timeOutInMs - Number of ms to wait, zero means forever.

Throws:
SecurityException - This method requires the or  action for the CoordinationPermission.

participate

boolean participate(Participant p)

Add a Participant to this Coordination. If this method returns true then there was a current Coordination
and the participant has successfully joined it. If there was no current Coordination then false is returned.
Once a Participant is participating it is guaranteed to receive a call back on either the
Participant.ended(Coordination) or Participant.failed(Coordination) method when the
Coordination is terminated. A participant can be added to the Coordination multiple times but it must only
be called back once when the Coordination is terminated. A Participant can only participate at a single
Coordination, if it attempts to block at another Coordination, then it will block until prior Coordinations are
finished. Notice that in edge cases the call back can happen before this method returns. The ordering of
the call-backs must follow the order of participation. If participant is participating multiple times the first time
it participates defines this order.

Returns:
true if the Coordination was active, otherwise false.

Throws:
CoordinationException - This exception should normally not be caught by the caller but allowed
to bubble up to the initiator of the coordination, it is therefore a RuntimeException. It signals that
this participant could not participate the current coordination. This can be cause by the following
reasons:

1. CoordinationException.DEADLOCK_DETECTED
2. CoordinationException.TIMEOUT
3. CoordinationException.UNKNOWN
getVariables

Map<Class<?>,?> getVariables()

A utility map associated with the current Coordination. Each coordination carries a map that can be used for communicating between different participants. To namespace of the map is a class, allowing for private data to be stored in the map by using implementation classes or shared data by interfaces.

Returns:
The map
Class CoordinationException

```
java.lang.Object
    java.lang.Throwable
        java.lang.Exception
            java.lang.RuntimeException
                org.osgi.service.coordination.CoordinationException
```

All Implemented Interfaces:
    Serializable

**public class CoordinationException**

extends RuntimeException

Thrown when an implementation detects a potential deadlock situation that it cannot solve. The name of the current coordination is given as argument.

### Field Summary

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEADLOCK_DETECTED</td>
<td></td>
<td>Adding a participant caused a deadlock.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td></td>
<td>The Coordination took too long to finish.</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>0</td>
<td>Unknown reason for this exception.</td>
</tr>
</tbody>
</table>

### Constructor Summary

```
CoordinationException(String message, String name, int reason)
```

Create a new Coordination Exception.

### Method Summary

```
String getName()  
int getReason()  
```

### Field Detail

**UNKNOWN**

```
public static final int UNKNOWN = 0  
```

Unknown reason for this exception.
### Class CoordinationException

#### DEADLOCK_DETECTED

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

Adding a participant caused a deadlock.

#### TIMEOUT

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

The Coordination took too long to finish.

### Constructor Detail

**CoordinationException**

```java
public CoordinationException(String message,
                              String name,
                              int reason)
```

Create a new Coordination Exception.

**Parameters:**
- `message` - The message
- `name` - The name of the Coordination
- `reason` - The reason for the exception.

### Method Detail

#### getName

```java
public String getName()
```

Answer the name of the Coordination associated with this exception.

**Returns:**
- the Coordination name

#### getReason

```java
public int getReason()
```

Answer the reason.

**Returns:**
- the reason
public class CoordinationPermission extends BasicPermission

The name parameter of the Permission is a filter expression. It asserts the bundle that is associated with the coordination. Additionally, the following attributes can be asserted:

1. coordination.name - The name of the coordination

Coordinator field summary:

- alwaysFail(String)
- begin(String)
- getCoordinations()
- isActive()
- isFailed()
- participate(Participant)
- participateOrBegin(Participant)

Coordination field summary:

- end()
- fail(String)
- getName()
- getParticipants()
- isFailed()
- setTimeout(long)
- terminate()

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String</td>
<td></td>
</tr>
<tr>
<td>ADMIN</td>
<td>29</td>
</tr>
</tbody>
</table>
| The action string | admin. | 29
| INITIATE       | 29   |
| String         |      |
| INITIATE       | 29   |
| String         |      |
| PARTICIPATE    | 29   |
| String         |      |

Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoordinationPermission(String filterExpression, String actions)</td>
<td>29</td>
</tr>
<tr>
<td>The name parameter specifies a filter condition.</td>
<td></td>
</tr>
<tr>
<td>CoordinationPermission(org.osgi.framework.Bundle bundle, String coordinationName, String actions)</td>
<td>29</td>
</tr>
<tr>
<td>The verification permission</td>
<td></td>
</tr>
</tbody>
</table>
Field Detail

INITIATE

public static final String INITIATE = "initiate"

Initiate a Coordination. An owner of this permission can initiate, end, fail, and terminate a Coordination.

ADMIN

public static final String ADMIN = "admin"

The action string admin.

PARTICIPATE

public static final String PARTICIPATE = "participate"

The action string participate.

Constructor Detail

CoordinationPermission

public CoordinationPermission(String filterExpression,
                          String actions)

The name parameter specifies a filter condition. The filter asserts the bundle that initiated the Coordination. An implicit grant is made for a bundle's own coordinations.

Parameters:
  filterExpression - A filter expression asserting the bundle associated with the coordination.
  actions - A comma separated combination of INITIATE, ADMIN, PARTICIPATE.

CoordinationPermission

public CoordinationPermission(Bundle bundle,
                          String coordinationName,
                          String actions)

The verification permission

Parameters:
  bundle - The bundle that will be the target of the filter expression.
  coordinationName - The name of the coordination or null
  actions - The set of actions required, which is a combination of INITIATE, ADMIN, PARTICIPATE.
public interface Coordinator

A Coordinator service provides a facility to coordinate activities between different parties. The Coordinator is a factory of Coordination objects. Coordination objects can be created. Once created, they can be pushed on a thread local stack `push(Coordination)` as an implicit parameter for calls to other parties, or they can be passed as an argument. The current top of the thread local stack can be obtained with `getCurrentCoordination()`. The `participate(Participant)` method on this service or the `Coordination.participate(Participant)` method can be used to participate in a Coordination. Participants can only participate in a single Coordination, if a Participant object is added to two different Coordinations it will block until any prior Coordination has been ended. A Coordination can end correctly when the `Coordination.end()` method is called or when it fails. If the Coordination ends correctly, all its participants are called on the `Participant.ended(Coordination)` method, otherwise the `Participant.failed(Coordination)` is called. A Coordination can fail because it times out or it is explicitly failed. A Coordination will timeout after an implementation defined amount of time that must be higher than 30 seconds unless overridden with configuration. This time can be set on a per Coordination basis with the `Coordination.addTimeout(long)` method. The typical usage of the Coordinator service is as follows:

```java
Coordination coordination = coordinator.begin("mycoordination");
try {
    doWork();
    if ( coordination.end() != Coordination.OK )
        log("failed");
} finally {
    coordination.terminate();
}
```

In the `doWork()` method, code can be called that requires a callback at the end of the Coordination. This code is for a Participant.

```java
void doWork() {
    if ( coordinator.participate(this) ) {
        beginWork();
    } else {
        beginWork();
        finishWork();
    }
}
void ended() {
    finishWork();
}
void failed() {
    undoWork();
}
```

Life cycle. All Coordinations that are begun through this service must automatically fail before this service is ungotten.

ThreadSafe

### Method Summary

<table>
<thead>
<tr>
<th>boolean alwaysFail(Throwables reason)</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always fail the current Coordination, if exists.</td>
<td>33</td>
</tr>
</tbody>
</table>
### Interface Coordinator

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>begin</code> (String name)</td>
<td>Begin a new Coordination and push it on the thread local stack with <code>push(Coordination)</code>.</td>
</tr>
<tr>
<td><code>create</code> (String name)</td>
<td>Create a new Coordination that is not associated with the current thread.</td>
</tr>
<tr>
<td><code>getCoordinations()</code></td>
<td>Provide a list of all Coordination objects.</td>
</tr>
<tr>
<td><code>getCurrentCoordination()</code></td>
<td>Return the current Coordination.</td>
</tr>
<tr>
<td><code>participate</code> (Participant participant)</td>
<td>Participate in the current Coordination or return false if there is none.</td>
</tr>
<tr>
<td><code>participateOrBegin</code> (Participant isActive)</td>
<td>Participate if there is an active Coordination otherwise initiate a new Coordination.</td>
</tr>
<tr>
<td><code>pop</code> ()</td>
<td>Pop the top of the thread local stack of Coordinations.</td>
</tr>
<tr>
<td><code>push</code> (Coordination c)</td>
<td>Associate the given Coordination object with a thread local stack.</td>
</tr>
</tbody>
</table>

#### Method Detail

**create**

```java
Coordination create(String name)
```

Create a new Coordination that is not associated with the current thread.

**Parameters:**
- name - The name of this coordination, a name does not have to be unique.

**Returns:**
- The new Coordination object or `null`

**Throws:**
- SecurityException - This method requires the action, no bundle check is done.

**begin**

```java
Coordination begin(String name)
```

Begin a new Coordination and push it on the thread local stack with `push(Coordination)`.

**Parameters:**
- name - The name of this coordination, a name does not have to be unique.

**Returns:**
- A new Coordination object

**Throws:**
- SecurityException - This method requires the action, no bundle check is done.

**push**

```java
Coordination push(Coordination c)
```
Interface Coordinator

Associate the given Coordination object with a thread local stack. The top of the thread local stack is returned with the `getCurrentCoordination()` method. To remove the Coordination from the top call `pop()`.

**Parameters:**
- c - The Coordination to push

**Returns:**
- c (for the builder pattern purpose)

```
pop
```

`Coordination pop()`

Pop the top of the thread local stack of Coordinations. If no current Coordination is present, return `null`.

**Returns:**
- The top of the stack or `null`

```
participate
```

`boolean participate(Participant participant)`

Participate in the current Coordination or return false if there is none. This method calls `getCurrentCoordination()`, if it is null, it will return false. Otherwise it will call `Coordination.participate(Participant)` and return the result of that method.

**Parameters:**
- participant - The participant of the Coordination

**Returns:**
- true if there was a current Coordination that could be successfully used to participate, otherwise false.

**Throws:**
- `CoordinationException` - This exception should normally not be caught by the caller but allowed to bubble up to the initiator of the coordination, it is therefore a `RuntimeException`. It signals that this participant could not participate the current coordination. This can be cause by the following reasons:
  1. `CoordinationException.DEADLOCK_DETECTED`
  2. `CoordinationException.TIMEOUT`
  3. `CoordinationException.UNKNOWN`
- `SecurityException` - This method requires the action for the current Coordination, if any.

```
participateOrBegin
```

`Coordination participateOrBegin(Participant ifActive)`

Participate if there is an active Coordination otherwise initiate a new Coordination. This method is a short cut that atomically checks if there is a current Coordination. It either returns a new current Coordination object if no current Coordination exists or it adds the participant to the current Coordination. Notice that this method can block until the participant is free to participate on the current or new Coordination. This method
Interface Coordinator

makes it simple to start a new Coordination or to participate in an existing Coordination. See
`begin(String)` and `participate(Participant)` for the details of those methods.

**Parameters:**
- `ifActive` - The participant

**Returns:**
- `null` if there is a current Coordination otherwise a newly initiated Coordination.

**Throws:**
- `SecurityException` - This method requires the action for the current Coordination, if any.
  Otherwise it requires to create a new coordination.

### alwaysFail

```java
boolean alwaysFail(Throwable reason)
```

Always fail the current Coordination, if exists. Must fail the current Coordination and return `true` or return
`false` if there is no current Coordination.

**Parameters:**
- `reason` - Throwable describing why the collaboration must always fail for debugging or `null`.

**Returns:**
- `true` if there was a current Coordination and `false` if not.

### getCurrentCoordination

```java
Coordination getCurrentCoordination()
```

Return the current Coordination. The current Coordination is the top of the thread local stack of
Coordinations. If the stack is empty, there is no current Coordination.

**Returns:**
- `null` when the thread local stack is empty, otherwise the top of the thread local stack of Coordinations.

### getCoordinations

```java
Collection<Coordination> getCoordinations()
```

Provide a list of all Coordination objects. Answer a read only list of active Coordination. This list must be a
mutable snapshot of the current situation. Changes to the list must not affect the original. Coordination
objects are capabilities and designed to be used only on the Coordination thread. The returned list must
only contain the Coordinations for which the caller has, without this permission an empty list must be
returned.

**Returns:**
- a list of Coordination objects filter by.
public interface Participant

A Participant participates in a Coordination. A Participant can participate in a Coordination by calling Coordinator.participate(Participant) or Coordinator.participateOrBegin(Participant). After successfully initiating the participation, the Participant is called back when the Coordination is terminated. If a Coordination ends with the Coordination.end() method, then all the participants are called back on their ended(Coordination) method. If the initiator decides to fail the Coordination (or another party has called Coordinator.alwaysFail(Throwable)) then the failed(Coordination) method is called back. Participants are required to be thread safe for the ended(Coordination) method and the failed(Coordination) method. Both methods can be called on another thread. A Coordinator service must block a Participant when it tries to participate in multiple Coordinations.

ThreadSafe

<table>
<thead>
<tr>
<th>Method Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void ended(Coordination c)</td>
<td>34</td>
</tr>
<tr>
<td>The Coordination is being ended.</td>
<td></td>
</tr>
<tr>
<td>void failed(Coordination c)</td>
<td>34</td>
</tr>
<tr>
<td>The Coordination has failed and the participant is informed.</td>
<td></td>
</tr>
</tbody>
</table>

**Method Detail**

**failed**

```java
void failed(Coordination c)
throws Exception
```

The Coordination has failed and the participant is informed. A participant should properly discard any work it has done during the active coordination.

**Parameters:**
- c - The Coordination that does the callback

**Throws:**
- Exception - Any exception thrown should be logged but is further ignored and does not influence the outcome of the Coordination.

**ended**

```java
void ended(Coordination c)
throws Exception
```

The Coordination is being ended.

**Parameters:**
- c - The Coordination that does the callback
10 Considered Alternatives

Prepare phase
The original design had an isAgreed() method on the Participant. Before the end() method decided to pass or fail it would consult all the participants. If any participant decided to return false or throw an exception, the end method would assume a VETO and call failed on all participants.

This phase was discarded because it did not provide sufficient value for the given use cases.

No Asynchronous Failure
The original design did not allow the fail(String) method to be called except on the Coordination thread. This significantly simplified the implementation. However, the result is that bad code could easily leave a Coordination hanging, and thereby locking the participants. It was therefore decided that a timeout must really fail all participants, regardless of any cooperation of the initiator, despite the significant added complexity that this causes in the implementation.

XA Resources
During one of the conference calls it was suggested to skip the Coordinator and instead use JTA with XA Resources.

JTA and XA Resources are not lightweight solutions, one of the primary requirements. JTA is primary and all encompassing focus is to provide durable consistency, exactly this part is missing from the Coordinator requirements. Implementing an XA Resource is significant piece of work because an XA Resource object represent the resource manager and not the actual participation on a specific Coordination/Transaction. The XA Resource must be able to handle many different concurrent scenarios and must multiplex transaction activities via the Xid. For the design goal of XA Resources this is not a major problem because it is small in comparison to maintaining the ACID model, especially in the light of recoverability after a crash. Compare the interfaces of the Participant and the XAResource:

```java
donotdisplay
void commit(Xid xid, boolean onePhase)  
    Commits the global transaction specified by xid.
void end(Xid xid, int flags)  
    Ends the work performed on behalf of a transaction branch.
void forget(Xid xid)  
    Tells the resource manager to forget about a heuristically completed transaction branch.
int getTransactionTimeout()  
    Obtains the current transaction timeout value set for this XAResource instance.
boolean isSameRM(XAResource xares)
```
This method is called to determine if the resource manager instance represented by the target object is the same as the resource manager instance represented by the parameter xares.

```java
int prepare(Xid xid)
```

Ask the resource manager to prepare for a transaction commit of the transaction specified in xid.

```java
Xid[] recover(int flag)
```

Obtains a list of prepared transaction branches from a resource manager.

```java
void rollback(Xid xid)
```

Informs the resource manager to roll back work done on behalf of a transaction branch.

```java
boolean setTransactionTimeout(int seconds)
```

Sets the current transaction timeout value for this XAResource instance.

```java
void start(Xid xid, int flags)
```

Starts work on behalf of a transaction branch specified in xid.

Additionally, application servers make every attempt to hide transactions for the application developer and are very cautious letting third parties participate in transactions. It is a primary design goal of the Coordinator that many services leverage the Coordinator.

Coordinations instead are a very lightweight model to allow different service implementations to collaborate. If XA Resource were used for this then it would be unlikely that it would be used.

## 11Security Considerations

This specification provides a Coordination Permission. This permission can enforce the name of the coordination as well as the initiating bundle. The permission therefore uses a filter as name, as defined in the filter based permissions section in the core specification. There is one additional parameter for the filter:

```java
coordination.name
```

The following actions are defined:

1. INITIATE – Required to initiate a Coordination.
2. PARTICIPATE – Required to participate a Coordination.
3. ADMIN – Required to administrate a Coordinator

The permissions are indicated in the following table:

```
<table>
<thead>
<tr>
<th>Coordinator</th>
<th>INITIATE</th>
<th>PARTICIPATE</th>
<th>ADMIN</th>
<th>NONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>alwaysFail(String)</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>begin(String)</td>
<td>+</td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>getCoordinations()</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>isActive()</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>isFailed()</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>participate(Participant)</td>
<td>+</td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>participateOrBegin(Participant)</td>
<td>+ and</td>
<td>+</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>end()</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>fail(String)</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>getName()</td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
</tbody>
</table>
```
Interface Participant

*     getParticipants()                                              +      -
*     isFailed()                                                            +
*     setTimeout(long)                       +                       +      -
*     terminate()                                                           +

12Document Support

References

Author's Address

Name | Peter Kriens  
---- | ------- 
Company | aQute    
Address | 9c, Avenue St. Drezery 
Voice | +33-698332260 
e-mail | Peter Kriens@aQute.biz 

Name | David Bosschaert 
---- | ------- 
Company | Red Hat 
Address | 6700 Cork Airport Business Park, Kinsale Road, Cork, Ireland 
Voice | +353861704529 
e-mail | david@redhat.com

Acronyms and Abbreviations

End of Document
CM Combined Modifications

Draft
64 Pages

Abstract

This RFC combines the changes of RFC 144 Configuration Admin Extension and RFC 150 Configuration Enhancements. It defines the requires changes to the CM specification to support permissions enforced scoping, coordinated PID updates, and a more user friendly way of receiving the updated configurations for Managed Services and their factories.
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 Removal of Location Restrictions........................................................................................ 5
  5.2 Secure Segmentation Model............................................................................................... 6
    5.2.1 API Changes............................................................................................................... 8
  5.3 Coordination of Multiple PIDs............................................................................................ 8
    5.3.1 Change Counter on Configuration............................................................................. 9
  5.4 User Friendly Configuration................................................................................................. 9
    5.4.1 Synopsis..................................................................................................................... 9
    5.4.2 Configurable............................................................................................................... 10
    5.4.3 PID Interfaces............................................................................................................. 10
    5.4.4 Meta Information......................................................................................................... 11
    5.4.5 Meta Annotations......................................................................................................... 12
    5.4.6 Singleton Configurable.............................................................................................. 13
    5.4.7 Configurable Factory................................................................................................. 14
    5.4.8 Metatyping.................................................................................................................. 15
    5.4.9 Security....................................................................................................................... 16
  5.5 VersionedTargeted PIDs...................................................................................................... 16
    5.5.1 Version Ranges.......................................................................................................... 16
    5.5.2 Registering a ManagedService for a Versioned PID.................................................. 17
    5.5.3 Managed Service Factories........................................................................................ 18
    5.5.4 Backward Compatibility.............................................................................................. 18
    5.5.5 Alternative................................................................................................................... 18

6 Command Line API...................................................................................................................... 18

7 JMX API......................................................................................................................................... 19

8 Initial Spec Chapter..................................................................................................................... 19
1 Introduction

This RFC combines the proposed changes of RFC 144 and RFC 150. The reason this RFC is made because both RFCs contained a lot of historic information and adding one to the other felt wrong. This RFC is therefore a fresh start based on the teachings of these prior RFCs.

This RFC therefore merges three distinct functions:
• Coordinated updates
• Permissions for groups
• An easier way to receive updates

2 Application Domain

In CM as it is today there are two important roles. The first is the configurer, the party that provides does the configuring. The second role is the configuree, the party that is being configured.

The current Configuration Admin design provides locations. Locations are used to make it harder for a bundle to receive configurations that it was not supposed to receive, even without the use of Java security. The location worked as follows:

When a configuration is created it is bound to a location. The location can be bound explicitly in the call that creates the configuration when the location is specified, or it happens implicitly with the callers bundle location when no location is specified. When explicitly bound, it can also be set null. When the location is null, the first bundle that is updated will make the configuration “learn” the location. After the first update, the configuration becomes bound to that bundle. A bundle can only be updated with a configuration when the configuration's location exactly matches the bundle that receives the update's location.

The current Configuration Admin based on the model that a bundle has a unique PID under which it receives its configuration. PIDs are not supposed to be shared between bundles and configurations are property based and are therefore not type safe.

3 Problem Description

Configuration Admin is quite popular and plays a role in many OSGi specifications, even though it requires some getting used to. The reason it is often not well understood is because it reverses the normal model where the bundle actively gets its configuration when it needs it. In Configuration Admin, the model is that the Configuration Admin service always calls the bundle. This role reversal is similar to change dependency injection brought to the software industry.

However, a number of problems are present in CM.
1. The rather rigid model of a PID per bundle is too constrained. In practice, a configuration can be shared between bundles. For example, a configuration PID could define a PID for system configurations, describing global properties to be used by multiple bundles. There is clearly a tendency to decouple the PID from the bundle and make them an identifier for a cohesive aspect of the total configuration.

2. In a secure environment that hosts bundles from different providers, there is a need to ensure that these different providers can share configuration between bundles but also ensure that no configuration information leaks to other providers. There is a need for a group model where providers can become members of a group; no configurations should be crossing groups.

3. Turning PIDs in shared identifiers creates the need to receive the update of a group of PIDs in a single callback instead of a number of separated callbacks that then need coordination.

4. The current Configuration Admin provides a single complete set of properties for each update based on a PID. In practice, the receiver has to process these properties into their correct types and is often only interested in the change since last time it was called.

5. There is an issue today when multiple versions of the same bundle, which is configured through Configuration Admin, are present in the framework as they will share the same PID. This can cause unwanted results. It's quite possible that the configuration data schema has changed between the bundle versions and even if the data schema hasn't changed it's highly likely that the configuration for each will differ. A mechanism to version the PIDs can help in situations like this.

4 Requirements

Consult the Requirements sections of RFC 144 and RFC 150

5 Technical Solution

5.1 Removal of Location Restrictions

A PID is no longer restricted to a single bundle. Any bundle can use any PID and the Configuration Admin must update all bundles that register such a PID. This implies the following changes to the current specification:
1. Configuration Admin must no longer verify the location against the bundle’s location when it updates a bundle, the location must be ignored.

2. If the configuration’s location is null when a bundle is updated then this must remain null, the location must no longer be learned from the first target bundle.

Multiple bundles can now register the same PID so there is now an ordering issue between bundles. However, no ordering can be consistently applied because during an update phase a background thread will update the managed services, during the time this takes, target services can come and go. There is therefore no order specified.

### 5.2 Secure Group Model

The location in the configuration restricted a PID to a specific bundle. In a way, the location scoped the PID, albeit in a rather poor way. By sharing PIDs, the meaning of the location must change to simultaneously support being used by multiple bundles. The meaning of the location is therefore widened to allow multiple bundles, it is therefore now called a group. Groups are enforced using Java 2 security, without security they are ignored. Bundles can use configurations belonging to a group when they have an appropriate Configuration Permission for that group.

Group names are unrestricted strings. They can be locations (as they are in the current spec), names of providers, layer names (like system, user, application), numbers, etc., wildcard characters in the configurations location name have no special meaning. For example, com.example.* is a literal string and not a wildcard expression. It is the responsibility of the operator to define a consistent set of groups and ensure that they are properly used.

To support groups, the Configuration Permission is augmented with the following:

1. The name parameter, which provides the name of a resource, becomes the group name. Currently, only ‘*’ is allowed and it was never expressed what the name parameter was. That is the Configuration Permission now protects a group (was location). It is allowed to end the name parameter with a asterisk (‘*’ \u002A) to indicate a wildcard. A wildcard matches zero or more characters at that position. An actual configuration location of null is matched against any Configuration Permission name.

2. Add an UPDATED action. This action is required when a bundle wants to be updated for a specific PID. A bundle must only be updated when the group of the corresponding configuration matches the permission name.

3. Add a READ action. This action has been added to protect the getLocation method that provides access to the group.

4. Add a PLUGIN action to be used by Plugins.

Every bundle is granted an implicit permission for:

```
ConfigurationPermission <<LOCATION>>, CONFIGURE|READ|UPDATED
```

Where <<LOCATION>> stands for the location of the bundle. This rule makes the model equal to the current management model.

<table>
<thead>
<tr>
<th>group</th>
<th>Report</th>
<th>CONFIGURE</th>
<th>READ</th>
<th>UPDATED</th>
<th>PLUGIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>createFactoryConfiguration(String) &lt;&lt;LOCATION&gt;&gt;</td>
<td>&lt;&lt;LOCATION&gt;&gt;</td>
<td>&lt;never&gt;</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
createFactoryConfiguration(String pid, String loc)
getConfiguration(String)
getConfiguration(String pid, String loc)
listConfigurations(String)
getPid()
getProperties()
(Configuration)
update(Dictionary)
delete()
getFactoryPid()
update()
setBundleLocation(String loc)
getGroup()
modifyConfiguration
(ManagedService)
updated(Dictionary)
updated(String, Dictionary)
deleted(String)
configurationEvent

<table>
<thead>
<tr>
<th>Method</th>
<th>Report 1</th>
<th>Report 2</th>
<th>CONFIGURE</th>
<th>READ</th>
<th>UPDATED</th>
<th>PLUGIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>createFactoryConfiguration</td>
<td>loc</td>
<td>Except.</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getConfiguration</td>
<td>null</td>
<td>&lt;never&gt;</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getConfiguration</td>
<td>loc</td>
<td>Except.</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>listConfigurations</td>
<td>c : list</td>
<td>Filtered</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getPid()</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getProperties()</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Configuration)</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>delete()</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getFactoryPid()</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>update()</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>setBundleLocation</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getGroup()</td>
<td>c location</td>
<td>Except.</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>modifyConfiguration</td>
<td>c location</td>
<td>Filtered</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>(ManagedService)</td>
<td>c location</td>
<td>Filtered</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>updated(Dictionary)</td>
<td>c location</td>
<td>Filtered</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>updated(String, Dictionary)</td>
<td>c location</td>
<td>Filtered</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deleted(String)</td>
<td>c location</td>
<td>Filtered</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>configurationEvent</td>
<td>c location</td>
<td>Filtered</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Violations of the group model can be reported in two different ways:

1. Except - A Security Exception is thrown when no permission is implied.
2. Filtered – The target never sees the configuration because it is filtered based on permissions. The filtering decision must be based on the Bundle.hasPermission method, where the bundle is the bundle that will receive the configuration.

The second column defines where the group that is tested against the permission(s) comes from:

- `<LOCATION>` - The caller bundle's location
- `loc` - The loc parameter in the call
- `c : list | c location` - If the answer is a list, verify each configuration's location as group. Ignore if the location has no permission. That is configuration objects that do not have a matching group must be ignored, no security exception should be thrown.
- `c location` - The group is the location of the associated configuration object.
5.2.1 API Changes

It is proposed to deprecate the `getLocation` and the `get/create` methods that take a location. Instead the following methods should take over their role:

- `void setGroup(String)` – Assigns a segment to a configuration
- `String getGroup()` - Gets the group of a configuration.

5.3 Coordination of Multiple PIDs

The changes proposed by the earlier sections enable a model where a Managed Service (MS) is registered with multiple PIDs that each specify a fragment of the total configuration. For example, a MS could be registered with the following PIDs:

- `com.acme.system.pid`
- `com.acme.accounting.pid`
- `com.acme.application.a.pid`

Each PID defines a number of properties that different components in the system need to know. However, this model breaks the atomicity of the update: it is possible that `com.acme.accounting.pid` is updated but `com.acme.system.pid` and `com.acme.application.a.pid` not. Or they are updated shortly after one another. It is impossible for the MS to know when all updates have happened. This is a serious problem because processing the configuration properties can be an expensive operation, especially if an existing subsystem must be stopped to effectuate the new configuration.

One solution is to provide a new API for handing over all configuration objects at once but that is a rather heavy handed approach. A better solution is to leverage the Coordination API. The Coordination API allows different parties to collaborate without prior agreement (except to use the Coordination). It is therefore proposed to use the Coordinator Service in the following way.

Any of the following methods must begin a Coordination if no Coordination is active or otherwise participate the current Coordination:

- `getConfiguration(String pid, String loc)`
- `(Configuration) update(Dictionary dic)`
- `delete()`
- `update()`

If no Coordination is active, the specification remains unchanged. Any resulting update to the MS is updated on a different thread some undefined time in the future. If a Coordination is active, the Configuration Admin must not persistently store the configuration, schedule any background updates, nor send out any events. All events and updates must be queued until the participation callback ended is called. When failed is called, all queued up data must be ignored.

A PID must only be used on a single Coordination, when it is used on multiple Coordinations, the second thread must block. For this reason, a configurer should sort the PIDs it wants to update and update them in ascending order to prevent deadlocks.

Configuration objects with locked PIDs used outside a Coordination must block on all their methods that involve mutable state when locked on a Coordination. It is the responsibility of the management agent to prevent deadlocks to ensure that outside a Coordination changes do not disturb the management policy. To minimize deadlocks, it is strongly recommended to update multiple PIDs in ascending lexical order.

If the Coordination ends correctly the following must happen in the following order:
1. All changes must be made persistent inside the ended method. When a Configuration object changes, its version count (getChangeCount(), new method) must increase with one or more. Failures must be reported as exceptions, this will inform the initiator of the coordination that there was no clean end.

2. All events and updates are scheduled to run on another thread. This thread must first send out all events in the order they occurred and then sequentially update all the (registered) Managed Services in their natural service order as defined by service.ranking.

The key guarantee that Configuration Admin gives with a Coordination is that events are not delivered before all Configuration objects within a Coordination are persisted. When an event is received or a updated method is called, the persistent state contains all the changes, there are no race conditions.

The primary use case for this extension is to simplify the processing of multiple configurations. A Managed Service can now at a callback retrieve all its PIDs from Configuration Admin, merge their Dictionaries, and check if the configuration was changed from the previous time. The first time after a Coordination has ended, this will likely be the case if there was any change. The other events can then be ignored because the getChangeCount() method of the Configuration object will return the same value.

5.3.1 Factories and Coordination

Factory configuration updates must also be coordinated in the same way as MS.

5.3.2 Change Counter on Configuration

It is proposed to add an int getChangeCount() to the Configuration Admin Configuration object. This counter is incremented by one or more whenever the Configuration object is persisted. That is, at any moment in observable time the change counter is at least one higher than the persisted value. This will make it very quick for the configurer to detect that a configuration has been changed.

5.4 User Friendly Configuration

Managing PIDs and processing properties is not the most interesting code to write. It is also very error prone and hard because the properties can be in many different types though the configuree only needs a single type. Doing the conversion properly is very error prone. This section therefore proposes an extension service to Configuration Admin that makes the Configuration Admin service significantly easier to use. Though the Configuration Admin could implement these, this is a separate service that can also be built on top of existing Configuration Admin impls. The proposed model provides an alternative service for Managed Service and Managed Service Factory.

The basic service diagram is depicted in the following figure:
5.4.1 Synopsis

A single piece of work that needs to be configured can be configured by registering a Configurable\(<C>\) service, this maps to the Managed Service. A Managed Service Factory maps to a ConfigurableFactory\(<C>\) service. The Configurable\(<C>\) services do not accept a Dictionary or Map but instead take an object that implements a domain specific interface, which is the generic parameter \(C\). The name of this interface is treated as the PID and the method names are the names of the properties. The interface can implement other interfaces that also count as PIDs, thereby effectively merging the properties of different Configuration objects. The ConfigurableFactory\(<C>\) service provides Configurable\(<C>\) objects instructed by the creation of factory Configurations in Configuration Admin. Guarantees are given that the target bundles are only called when changes have been made and that callbacks to a single service are never concurrent.

5.4.2 Configurable

The basic concept of this service is a Configurable. There are two possibilities, the bundle knows exactly what work to do and just needs some parameters for it, that is, the configuration is a singleton. The other case is when the bundle can perform the work many times but need another party to tell when and with what parameters. These two use cases map to the Managed Service (one configurable) and Managed Service Factory (as many as needed, but directed by the Configuration Admin).

For example, a bundle can be prepared to act as a single web server when it is started. It therefore has only Configurable. Another bundle could provide a number of Web servers, each on their own port, document, and security settings. An administrator could then use a factory to start/stop different web servers matching its needs. The single piece of work case is the more traditional approach, which is usually well known by most developers. The second approach is called configuration driven design. It is no longer the developer that drives the actual deployed system, it is the deployer. Configuration Driven Design is very natural for OSGi based systems, only when a specific piece of work is really a singleton should the singleton model be used. The singleton model is popular because of tradition and simplicity but many designs become more flexible when they choose the configuration driven design model.

5.4.3 PID Interfaces

The Configurable interface has a generic parameter, which is not constrained: any interface will do. This parameter must define an interface that represent the configuration parameters while its name specifies the PID. It is an error to specify a class, and this should be logged. Such a PID interface is interpreted in the following way:

- The fully qualified name of the interface is the PID.
- The method names are the property names. Method names are not interpreted with the bean design pattern. That is, no set\(Xxx\) method, just the \(xxx()\) method for the \(xxx\) property.
These interfaces are called **PID interfaces**. A PID interface must have at least one property defined to act as a PID interface. The following code shows an example of a configuration interface for a web server:

```java
package com.example.web;
public interface WebConf { // PID = com.example.web.WebConf
    int port();              // property key = port
    String host();           // property key = host
}
```

In many cases, it can be very useful to get a composite configuration, that is, the configuration should be the merge of a number of PIDs. For example, a web server could have the need for some system wide properties. This facility is provided by extending the primary PID interface with any number of additional PID interfaces. This provides an ordered number of PIDs. The order is defined recursively, when the interface comes first and then all the interfaces it extends in declaration order recursively. For example:

```java
public interface MyWebConf extends WebConf, SysConf, AppConf {
}
public interface SysConf, NetworkConf {
    String email();
    String user_name();
}
public interface AppConf {
    String application_name();
    int thread_pool_size();
}
```

### there was a discussion that empty interfaces are not PID interfaces. This, however, conflicts with factories. Like to keep it as is.

In this case, the PIDs will be: MyWebConf, WebConf, SysConf, NetworkConf, and AppConf (prefixed with their appropriate packages). If the ordered list of PIDs contains duplicates then the first must remain and the later ones must be removed. This order is important because it defines the order in which the properties of the Configuration Admin are merged. Configurations with PIDs that are earlier will override the configurations of later PIDs when the property key is the same.

There are a number of limitations in using method names as property names because property names have a significantly more relaxed character set and Java reserved keywords cannot be used as method names. Therefore, if a property cannot be found in the configuration properties the configurer must attempt the modify the name of the method before it uses it as a property key:

- Property names often have a dot (‘.’) as separator. The dot is an invalid Java identifier character and can therefore not be used inside the name of a PID interface method. The replacement character is the underscore (‘_’). That is, a method with the name ‘a_b_c’ will match ‘a_b_c’ as well as ‘a.b.c’. A private property like .private must be defined as _private. Though camel case would be more natural in many cases, it creates a number of ambiguities when abbreviations are used.

- Java reserved keywords (if, else, static, etc) cannot be used as method names. These names can be escaped with a dollar sign at the start. For example, a method name like $new must match ‘new’.

Each method has a return type. The return type is used to convert the property. The property can be any of the basic types + collections and arrays. An implementation of this service must use the Aggregate Converter (RFC 147) service if present. If no such service is present, it must at least implement the basic conversions as described in the Aggregate Converter service, including any generic information if present.
5.4.4 Meta Information

The type safe model to access configuration properties of the PID interfaces is the preferred access model. However, there are special cases where a configuree (the Work implementer) wants to get some more detailed view of the underlying configuration. The configuree can get access to this information through the Meta interface. A PID interface that implements the Meta interface will automatically get access to the methods that provide information about the configuration. The Meta interface must not act as a PID. The following methods are provided:

- Map<String, ? > getProperties() - Return the properties for a callback to the setup(Object) method. The contents are the merged contents of all involved PIDs. The returned properties are not modifiable. The map is case insensitive.
- Map<String, ? > getPreviousProperties() - Return the properties from the previous callback to the setup(Object) method. The contents are the merged contents of all involved PIDs. The returned properties are not modifiable. The map is case insensitive.
- boolean isChanged(String... names) - Return true if any of the given property names has a different value than the last callback.
- Collection<String> getChangedKeys() - Return a collection of property names of the properties that are changed from the previous callback.
- List<String> getPids() - Return an unmodifiable collection of the PIDs that are involved in the associated PID interface.
- String getFactoryPid() - Return the factory PID or null if this PID interface is not related to a factory configuration.

5.4.5 Meta Annotations

The Meta interface also provides a number of annotations to annotate the PID Interfaces as well as put extra constraints on the property values:

- PID (TYPE) – This annotations overrides the default PID name. The default PID name is the fully qualified dotted name of the interface but this annotation allows any PID name to take its place. An empty string as PID indicates that an implemented interface should not be mapped to a PID. This is intended to be used in grouping other interfaces and other cases where the interface has no meaning with respect to the Configuration Admin service.
- Default (METHOD) – Provides a default (String) value if the value is not specified in the configuration properties. If no default is specified, the property is mandatory.
- Separated (METHOD) – Allows the source property to be a string separated by a separator. If the source property is not a string, it must be converted first to a string. The default separator is a whitespace bounded comma ("\s*,\s*") but this can be overridden. If the conversion fails the configuration must not be delivered and a log message should be generated.
- Pattern (METHOD) – Provides a number of regular expression that must match the property before it can be delivered, all patterns must match. The property must be converted to a string before the check is done. If not all patterns match, the configuration must not be delivered and a message should be logged.
• Range (METHOD) – Provides a range check on the return value. The value must match the range. The semantics of the range check differ on the return type:
  ◦ Subclass of Number – Numeric range check
  ◦ String – The length of the string must fall within the range
  ◦ Implements Collection – The size of the collection
  ◦ Arrays – The length of the array.

• Label – Provide a label for the property

• Description – Provide a description of the property

• Option (METHOD) – Provides a list of option values with optional labels, the property must match to one of the option values or the configuration must not be delivered. If the property is an array or collection, all values must map to one of the options. If the property or its elements are not strings, then they must be converted first to strings before checking.

5.4.6 Singleton Configurable

A class implementing a single piece of work should extend the Work interface and register as a Work<?> service. The Work interface looks like:

```java
public  interface Configurable<C> {
    void setup( C configuration );
    void deleted();
}
```

For example, this is how it can be used with the web server example:

```java
public class WebServer implements Configurable<WebConf> {
    public void setup( WebConf configuration ) { … }
    public void deleted() { … }
    … // other methods
}
```

The configurer will detect the registration of a Work service and then analyze the implementation class. Through reflection it can find the generic parameter of the Configurable<C> interface, C in this case. C is treated as a PID interface. This interface is then analyzed for multiple PIDs. There must be at least one PID specified (annotations can indicate that an interface’s PID must not be used), if this is not the case a message should be logged.

The configurer must then update the Configurable<C> service. If no configuration is available, or if some of the configuration is faulty as indicated by the constraints, then the Configurable<C>.deleted() method is called to signal that the Configurable<C> service is tracked but no configuration is available for it. Otherwise it must call setup() and provide it with an implementation of the PID interface. Such an object is trivially made with the Proxy class. The configurer must then track the PIDs in Configuration Admin. If one of the configurations change, the configurer must update the Configurable<C> service only when at least one of the properties has a different value. A Configurable<C> service must only be updated when the following requirements are met:

• All properties are either defined by Config Admin or have a default value.

• All properties as defined by the PID interfaces match their constraints (including default values)
At least one property has been changed since the previous callback.

No setup or delete call must be in progress.

The configurer must ensure that group commits (configuring multiple PIDs) are properly handled if a Coordinator is present, this is handled in the Coordinator section. With a Coordinator, it is guaranteed that the first PID update has all other PID's configuration already persisted. For non Coordinator cases, it is also advised to have a slight delay (~100 ms) between the detection of a changed configuration and the update of the Configurable<C> service to ensure the multiple PIDs are in a consistent state. Otherwise, a configurer could update multiple PIDs but then calls the Configurable multiple times for what is supposed to be the same update. How this is achieved is up to an implementation. However, if a Coordinator is present (and the Configuration Admin supports the Coordinator) then the setup call must see a consistent set of configurations.

If the Configurable<C> service is unregistered the configurer must clean up any data structures related to this service but it must no longer attempt to call setup() or deleted() on this service.

There is no limit to reusing PIDs, the same PIDs can be reused any number of times by different bundles.

### 5.4.7 Configurable Factory

In many cases it is desirable to leave the initiative to the configurer and allow it (or more precise the management agent) to create the Configurable <C> objects, analogous to the Managed Service Factory. In that case a configuee must register a Configurable Factory<C>, where C is the same generic parameter used for the Configurable <C> interface: the PID interface. The Configurable Factory<C> interface is a factory that can provide new Configurable <C> objects on demand with the newInstance() method.

```java
public interface ConfigurableFactory<C> {
    Configurable <C> newInstance();
}
```

The C generic parameter is the PID interface. Also here, this interface is analyzed for additional PIDs and there must at least be one PID specified. Unlike a singleton piece of work, a factory cannot merge multiple PIDs because there is a life cycle involved. A factory therefore must treat the first PID different from the other PIDs. The first PID is the primary PID and it specifies the factory PID. If the configurer detects a new instance of that factory PID, it will ask the factory for a new instance of a Configurable <C> object. This object is linked to the instance PID of the new factory configuration and will track its life cycle. That is, if the configuration is updated, the setup() method must be called and if it is deleted, the deleted() method must be called.

For example:

```java
void init() {  
    context.registerService( ConfigurableFactory.class.getName(),
        new Configurable Factory<WebConf>() {  
            Configurable <WebConf> newInstance() {  
                return new WebConfigurable ();
            }  
        },
        null );
}
```

The MyWebConf PID is now treated as a factory PID. If a new configuration is created for that factory PID it gets a synthesized PID, for example AB45-CFA9. The configurer will detect the new configuration and will ask the factory for a new Configurable <C> object. This object is now treated as if it was registered as a service with the instance PID AB45-CFA9. If additional PIDs are specified beyond and above the factory PID then these are treated as PIDs of Configuration objects, identical to a Configurable <C> service.
For example:

```java
public interface MyWebConf extends WebConf, SysConf, AppConf {
}
```

If C is a MyWebConf interface (as in the fore last example), then fully qualified name of MyWebConf will map to the factory PID and the fully qualified names of SysConf and AppConf map to singleton configuration PIDs. The resulting Configurable <C> object must be treated identical to a Configurable <C> service that would have been registered with a primary interface that maps to the instance PID of the factory configuration and the additional PIDs. In the example, this would conceptually look like:

```java
@PID("AB45-CFA9")
public interface MyWebConf extends WebConf, SysConf, AppConf { ... }
```

This implies all the semantics: merging of the properties, checking constraints, etc. However, because this object is not registered as a service, the life cycle of all instances are bounded by the WorkFactory<C> service. If this factory service becomes unregistered, all its associated Configurable <C> objects must be discarded as if they were conceptually unregistered.

A factory PID can be registered multiple times. The configurer must create a Configurable <C> for each registered factory and track the life cycles of these Configurable <C> objects independently. The ordering between the different factories is undefined.

### 5.4.8 Metatyping

The PID interfaces can be used for Meta Typing, the annotations provide sufficient information to create rich metatyping if a Meta Type service is available. A simple solution is for tools like bnd to create the Meta Type information from the PID interfaces. However, it would be beneficial if the configurer could also create the metatyping information based on inspecting the PID interfaces. There is an upcoming change in the Meta Type specification that allows third parties to provide Meta Type information on behalf of others.

Therefore, a bundle containing interface PIDs can therefore define the following bundle header:

```plaintext
Bundle-PIDInterfaces ::= '*' | fqn ( ',' fqn )* 
```

If the wild card is specified, the configurer must automatically provide Meta Type information for any PID interface it detects through the Configurable <C> or Configurable Factory<C> interfaces. If the header contains fqns, the interfaces specified in this header must be loadable from the bundle that specifies them to ensure class space consistency. A configurer must then register a Meta Type Provider on behalf of that bundle that provides Meta Type access to these interfaces.

This model requires the next version of the Meta Type Service as proposed in bug #1641.

### 5.4.9 Security

In a secure environment, the configurer must ensure that any security constraints as defined by Configuration Admin are enforced. A configuree must have all the permissions as if it registered Managed Services or Managed Services Factory for the appropriate PIDs.

### 5.5 Targeted PIDs

Now PIDs can be reused by multiple configuration targets, a need is created to restrict them, or at least to allow exceptions. It is therefore proposed that whenever a MS(F) needs a configuration object, it must look this up in a similar way as locale based properties. The pattern for looking up a Configuration object is:

```plaintext
<PID>({ '-' <bsn> ( '-' <version> ( '-' <id> )? )? } )?
```
Configuration Admin must first attempt the longest string and then work its way backward. This allows a configurer to override a PID for a specific bundle. For example, if a configurer wants to override the MyWebConf PID for bundles with bsn=com.acme.web, then it should configure:

```
MyWebConf-com.acme.web
```

If the PID is not found in getConfiguration() the PID must be created as is. These rules are valid for both MS and MSF. This model is sufficiently backwards compatible with the existing model. However, managers can now override configurations for specific bundles.

### 5.5.1 Alternative

An alternative proposal is to use a filter instead of the bsn/version/id triplet. The filter would use the definition in 2.4.2 Filter based Permissions in the Core Specification. The reasons this alternative has not been chosen are:

1. The chosen solution is easy to use because the extra arguments (bsn/version/id) are known, the 4 possible names are easy to construct and can then be looked up in an optimized way, for example with hashing. A filter will require a prefix match and then running each filter on the bundle.

2. The filter based permissions do not support versions

3. It creates the problem that multiple filters might match

---

### 6 Command Line API

The scope for configuration admin is “cm” for Configuration Manager. No extra commands need to be provided except for the commands present on the interfaces.

---

### 7 JMX API

Add the getChangeCounter method to the Configuration MBean
8 Javadoc
<table>
<thead>
<tr>
<th>Package Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.osgi.service.cm</code></td>
<td>19</td>
</tr>
<tr>
<td>Configuration Admin Package Version 1.4.</td>
<td></td>
</tr>
<tr>
<td><code>org.osgi.service.configurable</code></td>
<td>47</td>
</tr>
<tr>
<td>Formatter Package Version 1.0.</td>
<td></td>
</tr>
</tbody>
</table>
Package org.osgi.service.cm

Configuration Admin Package Version 1.4.

See: Description

### Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration</strong></td>
<td>The configuration information for a ManagedService or ManagedServiceFactory object.</td>
<td>20</td>
</tr>
<tr>
<td><strong>ConfigurationAdmin</strong></td>
<td>Service for administering configuration data.</td>
<td>25</td>
</tr>
<tr>
<td><strong>ConfigurationListener</strong></td>
<td>Listener for Configuration Events.</td>
<td>35</td>
</tr>
<tr>
<td><strong>ConfigurationPlugin</strong></td>
<td>A service interface for processing configuration dictionary before the update.</td>
<td>40</td>
</tr>
<tr>
<td><strong>ManagedService</strong></td>
<td>A service that can receive configuration data from a Configuration Admin service.</td>
<td>42</td>
</tr>
<tr>
<td><strong>ManagedServiceFactory</strong></td>
<td>Manage multiple service instances.</td>
<td>44</td>
</tr>
</tbody>
</table>

### Class Summary

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConfigurationEvent</strong></td>
<td>A Configuration Event.</td>
<td>29</td>
</tr>
<tr>
<td><strong>ConfigurationPermission</strong></td>
<td>Indicates a bundle's authority to configure bundles.</td>
<td>36</td>
</tr>
</tbody>
</table>

### Exception Summary

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConfigurationException</strong></td>
<td>An Exception class to inform the Configuration Admin service of problems with configuration data.</td>
<td>32</td>
</tr>
</tbody>
</table>

---

### Package org.osgi.service.cm Description

Configuration Admin Package Version 1.4.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

Import-Package: org.osgi.service.cm; version="[1.4,2.0)"

Example import for providers implementing the API in this package:

Import-Package: org.osgi.service.cm; version="[1.4,1.5)"
## Interface Configuration

**org.osgi.service.cm**

### public interface Configuration

The configuration information for a ManagedService or ManagedServiceFactory object. The Configuration Admin service uses this interface to represent the configuration information for a ManagedService or for a service instance of a ManagedServiceFactory.

A Configuration object contains a configuration dictionary and allows the properties to be updated via this object. Bundles wishing to receive configuration dictionaries do not need to use this class - they register a ManagedService or ManagedServiceFactory. Only administrative bundles, and bundles wishing to update their own configurations need to use this class.

The properties handled in this configuration have case insensitive String objects as keys. However, case is preserved from the last set key/value.

A configuration can be *bound* to a group (Bundle.getGroup() and setGroup(String)). Groups are checked with ConfigurationPermission, a configuration can only be used to update a target when they have the permission.

If a configuration's location is null, it is not yet bound to a location. It will become bound to the location of the first bundle that registers a ManagedService or ManagedServiceFactory object with the corresponding PID.

The same Configuration object is used for configuring both a Managed Service Factory and a Managed Service. When it is important to differentiate between these two the term "factory configuration" is used.

### Version:

$Id: 89dc37425f1382bb25407f9b59670739b3f70f8c $ 

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void delete()</td>
<td>Delete this Configuration object.</td>
<td>22</td>
</tr>
<tr>
<td>boolean equals(Object other)</td>
<td>Equality is defined to have equal PIDs Two Configuration objects are equal when their PIDs are equal.</td>
<td>23</td>
</tr>
<tr>
<td>String getBundleLocation()</td>
<td>Deprecated. Locations are now handled by groups</td>
<td>23</td>
</tr>
<tr>
<td>int getChangeCount()</td>
<td>Answer the change count.</td>
<td>24</td>
</tr>
<tr>
<td>String getFactoryPid()</td>
<td>For a factory configuration return the PID of the corresponding Managed Service Factory, else return null.</td>
<td>22</td>
</tr>
<tr>
<td>String getGroup()</td>
<td>Return the group this Configuration belong to or null.</td>
<td>23</td>
</tr>
<tr>
<td>String getPid()</td>
<td>Get the PID for this Configuration object.</td>
<td>21</td>
</tr>
<tr>
<td>Dictionary getProperties()</td>
<td>Return the properties of this Configuration object.</td>
<td>21</td>
</tr>
<tr>
<td>int hashCode()</td>
<td>Hash code is based on PID.</td>
<td>24</td>
</tr>
<tr>
<td>void setBundleLocation(String group)</td>
<td>Deprecated. Locations are now handled by groups, see setGroup(String)</td>
<td>23</td>
</tr>
<tr>
<td>void setGroup(String group)</td>
<td>Set the current group for this Configuration object or null.</td>
<td>23</td>
</tr>
</tbody>
</table>
Interface Configuration

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>update()</td>
<td>void update()</td>
<td>Update the Configuration object with the current properties.</td>
</tr>
<tr>
<td>update(Dictionary properties)</td>
<td>void update(Dictionary properties)</td>
<td>Update the properties of this Configuration object.</td>
</tr>
</tbody>
</table>

Method Detail

**getPid**

String getPid()

Get the PID for this Configuration object. This method requires [<group>,READ|CONFIGURE] permission. The tested group is the group of this Configuration object.

Returns: the PID for this Configuration object.

Throws: 
- IllegalArgumentException - if this configuration has been deleted
- SecurityException - if the caller does not have the proper permission.

**getProperties**

Dictionary getProperties()

Return the properties of this Configuration object. The Dictionary object returned is a private copy for the caller and may be changed without influencing the stored configuration. The keys in the returned dictionary are case insensitive and are always of type String. If called just after the configuration is created and before update has been called, this method returns null. This method requires [<group>,READ|CONFIGURE] permission. The tested group is the group of this Configuration object.

Returns: A private copy of the properties for the caller or null. These properties must not contain the "service.bundleLocation" property or the "service.configuration.group" property.

Throws: 
- IllegalArgumentException - if this configuration has been deleted
- SecurityException - if the caller does not have the proper permission.

**update**

void update(Dictionary properties) throws IOException

Update the properties of this Configuration object. Stores the properties in persistent storage after adding or overwriting the following properties:

- "service.pid": is set to be the PID of this configuration.
- "service.factoryPid": if this is a factory configuration it is set to the factory PID else it is not set.

These system properties are all of type String. If the corresponding Managed Service/Managed Service Factory is registered, its updated method must be called asynchronously. Else, this callback is delayed until aforementioned registration occurs. Also initiates an asynchronous call to all ConfigurationListener with a ConfigurationEvent.CM_UPDATED event. This method requires [<group>,CONFIGURE] permission. The tested group is the group of this Configuration object.

Parameters: 
- properties - the new set of properties for this configuration

Throws: 
- IOException - if update cannot be made persistent
**IllegalArgumentException** - if the Dictionary object contains invalid configuration types or contains case variants of the same key name.

**IllegalStateException** - if this configuration has been deleted

**SecurityException** - if the caller does not have the proper permission.

---

**delete**

```java
void delete() throws IOException
```

Delete this Configuration object. Removes this configuration object from the persistent store. Notify asynchronously the corresponding Managed Service or Managed Service Factory. A ManagedService object is notified by a call to its updated method with a null properties argument. A ManagedServiceFactory object is notified by a call to its deleted method. This method requires \(<\text{group}>\).CONFIGURE permission. The tested group is the group of this Configuration object. Also initiates an asynchronous call to all ConfigurationListeners with a ConfigurationEvent.CM_DELETED event.

**Throws:**

- IOException - If delete fails
- IllegalStateException - if this configuration has been deleted
- SecurityException - if the caller does not have the proper permission.

---

**getFactoryPid**

```java
String getFactoryPid()
```

For a factory configuration return the PID of the corresponding Managed Service Factory, else return null. This method requires \(<\text{group}>\).CONFIGURE|READ permission. The tested group is the group of this Configuration object.

**Returns:**

- factory PID or null

**Throws:**

- IllegalStateException - if this configuration has been deleted
- SecurityException - if the caller does not have the proper permission.

---

**update**

```java
void update() throws IOException
```

Update the Configuration object with the current properties. Initiate the updated callback to the Managed Service or Managed Service Factory with the current properties asynchronously.

This is the only way for a bundle that uses a Configuration Plugin service to initiate a callback. For example, when that bundle detects a change that requires an update of the Managed Service or Managed Service Factory via its ConfigurationPlugin object. This method requires \(<\text{group}>\).CONFIGURE permission. The tested group is the group of this Configuration object.

**Throws:**

- IOException - if update cannot access the properties in persistent storage
- IllegalStateException - if this configuration has been deleted
- SecurityException - if the caller does not have the proper permission.

**See Also:**

- ConfigurationPlugin
**setBundleLocation**

```java
void setBundleLocation(String group)
```

*Deprecated.* *Locations are now handled by groups,* see **setGroup(String)**

Alias for `{link #setGroup()}`.

**Parameters:**
- group - The group name

---

**getBundleLocation**

```java
String getBundleLocation()
```

*Deprecated.* *Locations are now handled by groups*

Alias for **getGroup()**.

**Returns:**
- the group name.

---

**getGroup**

```java
String getGroup()
```

Return the group this Configuration belong to or `null`. This method requires `<group>,CONFIGURE|READ` permission. The tested group is the group of this Configuration object.

**Returns:**
- the current group or `null`

**Throws:**
- `SecurityException` - if the caller does not have the proper permission.

---

**setGroup**

```java
void setGroup(String group)
```

Set the current group for this Configuration object or `null`. This method requires `<group>,CONFIGURE` permission. The tested group is the group of this Configuration object.

**Parameters:**
- group - The new current group.

**Throws:**
- `SecurityException` - if the caller does not have the proper permission.

---

**equals**

```java
boolean equals(Object other)
```

Equality is defined to have equal PIDs Two Configuration objects are equal when their PIDs are equal.

**Overrides:**
- `equals in class Object`

**Parameters:**
- other - Configuration object to compare against
Interface Configuration

Returns:
true if equal, false if not a Configuration object or one with a different PID.

hashCode

int hashCode()

Hash code is based on PID. The hashcode for two Configuration objects must be the same when the Configuration PID's are the same.

Overrides:
hashCode in class Object

Returns:
hash code for this Configuration object

getChangeCount

int getChangeCount()

Answer the change count. The change count is incremented every time the configuration is stored persistently.

Returns:
The change count
public interface ConfigurationAdmin

Service for administering configuration data. The main purpose of this interface is to store bundle configuration data persistently. This information is represented in Configuration objects. The actual configuration data is a Dictionary of properties inside a Configuration object. There are two principally different ways to manage configurations. First there is the concept of a Managed Service, where configuration data is uniquely associated with an object registered with the service registry. Next, there is the concept of a factory where the Configuration Admin service will maintain 0 or more Configuration objects for a Managed Service Factory that is registered with the Framework. The first concept is intended for configuration data about "things/services" whose existence is defined externally, e.g. a specific printer. Factories are intended for "things/services" that can be created any number of times, e.g. a configuration for a DHCP server for different networks. Bundles that require configuration should register a Managed Service or a Managed Service Factory in the service registry. A registration property named service.pid (persistent identifier or PID) must be used to identify this Managed Service or Managed Service Factory to the Configuration Admin service. When the ConfigurationAdmin detects the registration of a Managed Service, it checks its persistent storage for a configuration object whose service.pid property matches the PID service property (service.pid) of the Managed Service. If found, it calls ManagedService.updated() method with the new properties. The implementation of a Configuration Admin service must run these call-backs asynchronously to allow proper synchronization. When the Configuration Admin service detects a Managed Service Factory registration, it checks its storage for configuration objects whose service.factoryPid property matches the PID service property of the Managed Service Factory. For each such Configuration objects, it calls the ManagedServiceFactory.updated method asynchronously with the new properties. The calls to the updated method of a ManagedServiceFactory must be executed sequentially and not overlap in time. In general, bundles having permission to use the Configuration Admin service can only access and modify their own configuration information. Accessing or modifying the configuration of another bundle requires ConfigurationPermission[*,CONFIGURE]. A Configuration can be associated with a group. Managed Service (Factory) services must have the ConfigurationPermission.UPDATEDConfigurationPermission to be updated by Configuration Admin. The null group indicates that no security check is necessary. The method descriptions of this class refer to a concept of "the calling bundle". This is a loose way of referring to the bundle which obtained the Configuration Admin service from the service registry. Implementations of ConfigurationAdmin must use a org.osgi.framework.ServiceFactory to support this concept.

Version:
$Id: a2d6c57503f7ee9308f85d38dc8f7219ac3906ab $
Field Detail

SERVICE_FACTORYPID

public static final String SERVICE_FACTORYPID = "service.factoryPid"

Configuration property naming the Factory PID in the configuration dictionary. The property's value is of type String.

Since: 1.1

SERVICE_BUNDLELOCATION

public static final String SERVICE_BUNDLELOCATION = "service.bundleLocation"

Configuration property naming the location of the bundle that is associated with a Configuration object. This property can be searched for but must not appear in the configuration dictionary for security reason. The property's value is of type String.

Since: 1.1

SERVICE_CONFIGURATION_GROUP

public static final String SERVICE_CONFIGURATION_GROUP = "service.configuration.group"

Configuration property naming the group of the bundle that is associated with a Configuration object. This property can be searched for but must not appear in the configuration dictionary for security reason. The property's value is of type String.

Since: 1.4

Method Detail

createFactoryConfiguration

Configuration createFactoryConfiguration(String factoryPid)
throws IOException

Create a new factory Configuration object with a new PID. The properties of the new Configuration object are null until the first time that its Configuration.update(Dictionary) method is called. It is not required that the factoryPid maps to a registered Managed Service Factory. This method requires [group,CONFIGURE] permission.

Parameters:
factoryPid - PID of factory (not null).

Returns:
A new Configuration object.

Throws:
IOException - if access to persistent storage fails.
SecurityException - if caller does not have the proper permission.
createFactoryConfiguration

```java
Configuration createFactoryConfiguration(String factoryPid,
                                        String group)
    throws IOException
```

Create a new factory Configuration object with a new PID. The properties of the new Configuration object are null until the first time that its `Configuration.update(Dictionary)` method is called. It is not required that the `factoryPid` maps to a registered Managed Service Factory. This method requires \[<group>,CONFIGURE\] permission.

**Parameters:**
- `factoryPid` - PID of factory (not null).
- `group` - A bundle group, or null.

**Returns:**
a new Configuration object.

**Throws:**
- IOException - if access to persistent storage fails.
- SecurityException - if caller does not have the proper permission.

getConfiguration

```java
Configuration getConfiguration(String pid,
                                 String group)
    throws IOException
```

Get an existing Configuration object from the persistent store, or create a new Configuration object.

If a Configuration with this PID already exists in Configuration Admin service return it. The group parameter is ignored in this case.

Else, return a new Configuration object. This method requires \[<group>,CONFIGURE|READ\] permission.

**Parameters:**
- `pid` - Persistent identifier.
- `group` - The bundle group string, or null.

**Returns:**
An existing or new Configuration object.

**Throws:**
- IOException - if access to persistent storage fails.
- SecurityException - if caller does not have the proper permission.

getConfiguration

```java
Configuration getConfiguration(String pid)
    throws IOException
```

Get an existing or new (link Configuration) object from the persistent store. If the Configuration object for this PID does not exist, create a new Configuration object for that PID, where properties are null. This method requires \[<group>,CONFIGURE|READ\] permission. The group is derived from the group set on the Configuration object if it exists, otherwise it is null.

**Parameters:**
- `pid` - persistent identifier.

**Returns:**
an existing or new Configuration matching the PID.

**Throws:**
- IOException - if access to persistent storage fails.
- SecurityException - if the caller does not have the proper permission.
listConfigurations

```java
Configuration[] listConfigurations(String filter)
    throws IOException,
            org.osgi.framework.InvalidSyntaxException
```

List the current Configuration objects which match the filter and for which the caller has permission.

Only Configuration objects with non-null properties are considered current. That is, Configuration.getProperties() is guaranteed not to return null for each of the returned Configuration objects. Only Configuration objects where the caller has the permission [group],READ|CONFIGURE] for must be returned.

The syntax of the filter string is as defined in the org.osgi.framework.Filter class. The filter can test any configuration properties including the following:

- `service.pid`-String - the PID under which this is registered
- `SERVICE_FACTORYPID`-String - the factory if applicable
- `SERVICE_BUNDLELOCATION`-String - the bundle location
- `SERVICE_CONFIGURATION_GROUP`-String - the group of the Configuration object

The filter can also be null, meaning that all Configuration objects should be returned.

Parameters:
- filter - A filter string, or null to retrieve all Configuration objects.

Returns:
- All matching Configuration objects and for which there is permission, or null if there aren't any.

Throws:
- IOException - if access to persistent storage fails
- org.osgi.framework.InvalidSyntaxException - if the filter string is invalid
public class ConfigurationEvent
extends Object

A Configuration Event.

ConfigurationEvent objects are delivered to all registered ConfigurationListener service objects. ConfigurationEvents must be asynchronously delivered in chronological order with respect to each listener. A type code is used to identify the type of event. The following event types are defined:

- CM_UPDATED
- CM_DELETED

Additional event types may be defined in the future. Security Considerations. ConfigurationEvent objects do not provide Configuration objects, so no sensitive configuration information is available from the event. If the listener wants to locate the Configuration object for the specified PID, it must use ConfigurationAdmin. However, if the Configuration object has a set group, then the receiving bundle must have the [<group>,READ|CONFIGURE] Configuration Permission.

Since: 1.2
Version:  $Id: b0efbb26d23b0931d901cac9c16f0a49b06a530b $
See Also: ConfigurationListener

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int CM_DELETED</td>
<td>30</td>
</tr>
<tr>
<td>static int CM_UPDATED</td>
<td>30</td>
</tr>
</tbody>
</table>

CM_DELETED: A Configuration has been deleted.
CM_UPDATED: A Configuration has been updated.

Constructor Summary

ConfigurationEvent(org.osgi.framework.ServiceReference reference, int type, String factoryPid, String pid)

Constructs a ConfigurationEvent object from the given ServiceReference object, event type, and pids.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getFactoryPid()</td>
<td>30</td>
</tr>
<tr>
<td>Returns the factory pid of the associated configuration.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getPid()</td>
<td>31</td>
</tr>
<tr>
<td>Returns the pid of the associated configuration.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.osgi.framework.ServiceReference getReference()</td>
<td>31</td>
</tr>
<tr>
<td>Return the ServiceReference object of the Configuration Admin service that created this event.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>int getType()</td>
<td>31</td>
</tr>
<tr>
<td>Return the type of this event.</td>
<td></td>
</tr>
</tbody>
</table>
Class ConfigurationEvent

Field Detail

CM_UPDATED

public static final int CM_UPDATED = 1

A Configuration has been updated.

This ConfigurationEvent type that indicates that a Configuration object has been updated with new properties. An event is fired when a call to Configuration.update(Dictionary) successfully changes a configuration.

The value of CM_UPDATED is 1.

CM_DELETED

public static final int CM_DELETED = 2

A Configuration has been deleted.

This ConfigurationEvent type that indicates that a Configuration object has been deleted. An event is fired when a call to Configuration.delete() successfully deletes a configuration.

The value of CM_DELETED is 2.

Constructor Detail

ConfigurationEvent

public ConfigurationEvent(org.osgi.framework.ServiceReference reference,
int type,
String factoryPid,
String pid)

Constructs a ConfigurationEvent object from the given ServiceReference object, event type, and pids.

Parameters:
reference - The ServiceReference object of the Configuration Admin service that created this event.
type - The event type. See getType().
factoryPid - The factory pid of the associated configuration if the target of the configuration is a ManagedServiceFactory. Otherwise null if the target of the configuration is a ManagedService.
pid - The pid of the associated configuration.

Method Detail

getFactoryPid

public String getFactoryPid()

Returns the factory pid of the associated configuration.

Returns:
Returns the factory pid of the associated configuration if the target of the configuration is a ManagedServiceFactory. Otherwise null if the target of the configuration is a ManagedService.
getPid

public String getPid()

    Returns the pid of the associated configuration.

    Returns:  
    Returns the pid of the associated configuration.

getype

public int getType()

    Return the type of this event.

    The type values are:
    - CM_UPDATED
    - CM_DELETED

    Returns:  
    The type of this event.

getReference

public org.osgi.framework.ServiceReference getReference()

    Return the ServiceReference object of the Configuration Admin service that created this event.

    Returns:  
    The ServiceReference object for the Configuration Admin service that created this event.
Class ConfigurationException

public class ConfigurationException extends Exception

An Exception class to inform the Configuration Admin service of problems with configuration data.

Version:
$Id: c9fb6fb10fb8fc75291a073348fa0f4e56f248a7 $
ConfigurationException

public ConfigurationException(String property,
                                      String reason,
                                      Throwable cause)

Create a ConfigurationException object.

Parameters:
  property - name of the property that caused the problem, null if no specific property was the
  cause
  reason - reason for failure
  cause - The cause of this exception.

Since: 1.2

Method Detail

getProperty

public String getProperty()

Return the property name that caused the failure or null.

Returns: name of property or null if no specific property caused the problem

getReason

public String getReason()

Return the reason for this exception.

Returns: reason of the failure

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: 1.2

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: 1.2
public interface ConfigurationListener

Listener for Configuration Events. When a ConfigurationEvent is fired, it is asynchronously delivered to a
ConfigurationListener.

ConfigurationListener objects are registered with the Framework service registry and are notified with a
ConfigurationEvent object when an event is fired.

ConfigurationListener objects can inspect the received ConfigurationEvent object to determine its type, the
pid of the Configuration object with which it is associated, and the Configuration Admin service that fired the
event.

Security Considerations. Bundles wishing to monitor configuration events will require
ServicePermission[ConfigurationListener,REGISTER] to register a ConfigurationListener service.

Since: 1.2
Version: $Id: bc0872c4df2541cba4060a0036f8aeb24a608051 $

Method Summary

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>configurationEvent</td>
<td>35</td>
</tr>
</tbody>
</table>

configurationEvent

void configurationEvent(ConfigurationEvent event)

Receives notification of a Configuration that has changed and that is associated with a
group for which the listener’s bundle has [<group>,READ|CONFIGURE] Configuration Permission.

Parameters:
- event - The ConfigurationEvent.
Class ConfigurationPermission

org.osgi.service.cm

java.lang.Object
  java.security.Permission
    java.security.BasicPermission
      org.osgi.service.cm.ConfigurationPermission

All Implemented Interfaces:
  Guard, Serializable

final public class ConfigurationPermission
  extends BasicPermission

Indicates a bundle’s authority to configure bundles. This permission has only a single action: CONFIGURE.

Since: 1.2
Version: $Id: 6b918f50972aaa9890b9a2d613ddd08598509280 $
ThreadSafe

Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static String CONFIGURE</td>
<td>37</td>
</tr>
<tr>
<td>The action string &quot;configure&quot;.</td>
<td></td>
</tr>
<tr>
<td>static String PLUGIN</td>
<td>37</td>
</tr>
<tr>
<td>The action string &quot;plugin&quot;.</td>
<td></td>
</tr>
<tr>
<td>static String READ</td>
<td>37</td>
</tr>
<tr>
<td>The action string &quot;read&quot;.</td>
<td></td>
</tr>
<tr>
<td>static String UPDATED</td>
<td>37</td>
</tr>
<tr>
<td>The action string &quot;updated&quot;.</td>
<td></td>
</tr>
</tbody>
</table>

Constructor Summary

ConfigurationPermission(String name, String actions)
  Create a new ConfigurationPermission.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals(Object obj)</td>
<td>38</td>
</tr>
<tr>
<td>Determines the equality of two ConfigurationPermission objects.</td>
<td></td>
</tr>
<tr>
<td>String getActions()</td>
<td>38</td>
</tr>
<tr>
<td>Returns the canonical string representation of the ConfigurationPermission actions.</td>
<td></td>
</tr>
<tr>
<td>int hashCode()</td>
<td>38</td>
</tr>
<tr>
<td>Returns the hash code value for this object.</td>
<td></td>
</tr>
<tr>
<td>boolean implies(Permission p)</td>
<td>37</td>
</tr>
<tr>
<td>Determines if a ConfigurationPermission object &quot;implies&quot; the specified permission.</td>
<td></td>
</tr>
<tr>
<td>Permission Collection newPermissionCollection()</td>
<td>38</td>
</tr>
<tr>
<td>Returns a new PermissionCollection object suitable for storing ConfigurationPermissions.</td>
<td></td>
</tr>
</tbody>
</table>
Field Detail

CONFIGURE

public static final String CONFIGURE = "configure"

The action string "configure".

READ

public static final String READ = "read"

The action string "read". This action protects the Configuration.getGroup() method that provides access to the group.

UPDATED

public static final String UPDATED = "updated"

The action string "updated". This action is required when a bundle wants to be updated for a specific PID. A bundle must only be updated when the group of the corresponding Configuration object matches the permission name.

PLUGIN

public static final String PLUGIN = "plugin"

The action string "plugin". Required by Configuration Plugins.

Constructor Detail

ConfigurationPermission

public ConfigurationPermission(String name, String actions)

Create a new ConfigurationPermission. The name of the permission represents the group name. The wildcard character ("*") is allowed to match wildcards as defined in BasicPermission. The actions are:

1. CONFIGURE
2. READ
3. UPDATED
4. PLUGIN

Parameters:
name - Name represents a group name or null.
actions - configure (canonical order).

Method Detail

implies

public boolean implies(Permission p)
Determine if a `ConfigurationPermission` 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.

```java
public boolean equals(Object obj)
```

Determines the equality of two `ConfigurationPermission` objects.

Two `ConfigurationPermission` objects are equal.

** Overrides:**
`equals in class BasicPermission`

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

** Returns:**
true if obj is equivalent to this `ConfigurationPermission`; false otherwise.

```java
public int hashCode()
```

Returns the hash code value for this object.

** Overrides:**
`hashCode in class BasicPermission`

** Returns:**
Hash code value for this object.

```java
public String getActions()
```

Returns the canonical string representation of the `ConfigurationPermission` actions.

Always returns present `ConfigurationPermission` actions in the following order: CONFIGURE

** Overrides:**
`getActions in class BasicPermission`

** Returns:**
Canonical string representation of the `ConfigurationPermission` actions.

```java
public PermissionCollection newPermissionCollection()
```

Returns a new `PermissionCollection` object suitable for storing `ConfigurationPermission`s.

** Overrides:**
`newPermissionCollection in class BasicPermission`
Returns:
   A new PermissionCollection object.
Interface ConfigurationPlugin

```java
public interface ConfigurationPlugin
```

A service interface for processing configuration dictionary before the update. A bundle registers a `ConfigurationPlugin` object in order to process configuration updates before they reach the Managed Service or Managed Service Factory. The Configuration Admin service will detect registrations of Configuration Plugin services and must call these services every time before it calls the `ManagedService` or `ManagedServiceFactoryupdated` method. The Configuration Plugin service thus has the opportunity to view and modify the properties before they are passed to the Managed Service or Managed Service Factory.

Configuration Plugin (plugin) services have full read/write access to all configuration information. Therefore, bundles using this facility should be trusted. Access to this facility should be limited with `ServicePermission[ConfigurationPlugin,REGISTER]`. Additionally, a Plugin must have the Configuration Permission `[<group>,PLUGIN]` for every Configuration object they see, where the group is the Configuration object's group. Implementations of a Configuration Plugin service should assure that they only act on appropriate configurations.

The `Integerservice.cmRanking` registration property may be specified. Not specifying this registration property, or setting it to something other than an `Integer`, is the same as setting it to the `Integer` zero. The `service.cmRanking` property determines the order in which plugins are invoked. Lower ranked plugins are called before higher ranked ones. In the event of more than one plugin having the same value of `service.cmRanking`, then the Configuration Admin service arbitrarily chooses the order in which they are called.

By convention, plugins with `service.cmRanking < 0` or `service.cmRanking > 1000` should not make modifications to the properties. The Configuration Admin service has the right to hide properties from plugins, or to ignore some or all the changes that they make. Any such behavior is entirely implementation defined.

A plugin may optionally specify a `cm.target` registration property whose value is the PID of the Managed Service or Managed Service Factory whose configuration updates the plugin is intended to intercept. The plugin will then only be called with configuration updates that are targeted at the Managed Service or Managed Service Factory with the specified PID. Omitting the `cm.target` registration property means that the plugin is called for all configuration updates.

Version:

```
$Id: 3b134aa21524b8f9cd461f040a8899334eb07d2a $
```

### Field Summary

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String CM_RANKING</td>
<td>41</td>
</tr>
<tr>
<td>String CM_TARGET</td>
<td>40</td>
</tr>
</tbody>
</table>

CM_RANKING
A service property to specify the order in which plugins are invoked.

CM_TARGET
A service property to limit the Managed Service or Managed Service Factory configuration dictionaries a Configuration Plugin service receives.

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void modifyConfiguration(org.osgi.framework.ServiceReference reference, Dictionary properties)</td>
<td>41</td>
</tr>
</tbody>
</table>

View and possibly modify the a set of configuration properties before they are sent to the Managed Service or the Managed Service Factory.

### Field Detail

**CM_TARGET**

```java
public static final String CM_TARGET = "cm.target"
```
A service property to limit the Managed Service or Managed Service Factory configuration dictionaries a Configuration Plugin service receives. This property contains a String[] of PIDs. A Configuration Admin service must call a Configuration Plugin service only when this property is not set, or the target service’s PID is listed in this property.

**CM_RANKING**

```java
public static final String CM_RANKING = "service.cmRanking"
```

A service property to specify the order in which plugins are invoked. This property contains an Integer ranking of the plugin. Not specifying this registration property, or setting it to something other than an Integer, is the same as setting it to the Integer zero. This property determines the order in which plugins are invoked. Lower ranked plugins are called before higher ranked ones.

Since: 1.2

### Method Detail

**modifyConfiguration**

```java
void modifyConfiguration(org.osgi.framework.ServiceReference reference, Dictionary properties)
```

View and possibly modify the a set of configuration properties before they are sent to the Managed Service or the Managed Service Factory. The Configuration Plugin services are called in increasing order of their service.cmRanking property. If this property is undefined or is a non-Integer type, 0 is used. This method should not modify the properties unless the service.cmRanking of this plugin is in the range 0 \lt;= service.cmRanking \lt;= 1000. If this method throws any Exception, the Configuration Admin service must catch it and should log it. Only Configuration objects for which the plugin has Configuration Permission are passed.

**Parameters:**

- reference - reference to the Managed Service or Managed Service Factory
- properties - The configuration properties. This argument must not contain the "service.bundleLocation" property. The value of this property may be obtained from the Configuration.getBundleLocation method.
public interface ManagedService

A service that can receive configuration data from a Configuration Admin service. A Managed Service is a service that needs configuration data. Such an object should be registered with the Framework registry with the service.pid property set to some unique identifier called a PID. If the Configuration Admin service has a Configuration object corresponding to this PID, it will callback the updated() method of the ManagedService object, passing the properties of that Configuration object. If it has no such Configuration object, then it calls back with a null properties argument. Registering a Managed Service will always result in a callback to the updated() method provided the Configuration Admin service is, or becomes active. This callback must always be done asynchronously. Else, every time that either of the updated() methods is called on that Configuration object, the ManagedService.updated() method with the new properties is called. If the delete() method is called on that Configuration object, ManagedService.updated() is called with a null for the properties parameter. All these callbacks must be done asynchronously. The following example shows the code of a serial port that will create a port depending on configuration information.

class SerialPort implements ManagedService {
    ServiceRegistration registration;
    Hashtable configuration;
    CommPortIdentifier id;

    synchronized void open(CommPortIdentifier id, BundleContext context) {
        this.id = id;
        registration = context.registerService(
            ManagedService.class.getName(),
            this,
            getDefaults()
        );
    }

    Hashtable getDefaults() {
        Hashtable defaults = new Hashtable();
        defaults.put( "port", id.getName() );
        defaults.put( "product", "unknown" );
        defaults.put( "baud", "9600" );
        defaults.put( Constants.SERVICE_PID,
            "com.acme.serialport." + id.getName() );
        return defaults;
    }

    public synchronized void updated(
        Dictionary configuration
    ) {  
        if ( configuration == null )
            registration.setProperties( getDefaults() );
        else {
            setSpeed( configuration.get("baud") );
            registration.setProperties( configuration );
        }
    }
}

As a convention, it is recommended that when a Managed Service is updated, it should copy all the properties it does not recognize into the service registration properties. This will allow the Configuration Admin service to set properties on services which can then be used by other applications. A Managed Service must only be called if its bundle has the [group,UPDATED] Configuration Permission for the Configuration Object.
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void updated(Dictionary properties)</code></td>
<td></td>
<td>Update the configuration for a Managed Service.</td>
</tr>
</tbody>
</table>

### Method Detail

**updated**

```java
void updated(Dictionary properties)
    throwsConfigurationException
```

Update the configuration for a Managed Service.

When the implementation of `updated(Dictionary)` detects any kind of error in the configuration properties, it should create a new `ConfigurationException` which describes the problem. This can allow a management system to provide useful information to a human administrator.

If this method throws any other `Exception`, the Configuration Admin service must catch it and should log it.

The Configuration Admin service must call this method asynchronously which initiated the callback. This implies that implementors of Managed Service can be assured that the callback will not take place during registration when they execute the registration in a synchronized method.

**Parameters:**
- `properties` - A copy of the Configuration properties, or `null`. This argument must not contain the "service.bundleLocation" property. The value of this property may be obtained from the `Configuration.getBundleLocation` method.

**Throws:**
- `ConfigurationException` - when the update fails
Manage multiple service instances. Bundles registering this interface are giving the Configuration Admin service the ability to create and configure a number of instances of a service that the implementing bundle can provide. For example, a bundle implementing a DHCP server could be instantiated multiple times for different interfaces using a factory. Each of these service instances is represented, in the persistent storage of the Configuration Admin service, by a factory Configuration object that has a PID. When such a Configuration is updated, the Configuration Admin service calls the ManagedServiceFactory updated method with the new properties. When updated is called with a new PID, the Managed Service Factory should create a new factory instance based on these configuration properties. When called with a PID that it has seen before, it should update that existing service instance with the new configuration information. In general it is expected that the implementation of this interface will maintain a data structure that maps PIDs to the factory instances that it has created. The semantics of a factory instance are defined by the Managed Service Factory. However, if the factory instance is registered as a service object with the service registry, its PID should match the PID of the corresponding Configuration object (but it should not be registered as a Managed Service!). An example that demonstrates the use of a factory. It will create serial ports under command of the Configuration Admin service.

```java
class SerialPortFactory
    implements ManagedServiceFactory {
        ServiceRegistration registration;
        Hashtable ports;
        void start(BundleContext context) {
            Hashtable properties = new Hashtable();
            properties.put( Constants.SERVICE_PID,
                "com.acme.serialportfactory" );
            registration = context.registerService(
                ManagedServiceFactory.class.getName(),
                this,
                properties
            );
        }
        public void updated( String pid,
            Dictionary properties ) {
            String portName = (String) properties.get("port");
            SerialPortService port =
                (SerialPort) ports.get( pid );
            if ( port == null ) {
                port = new SerialPortService();
                ports.put( pid, port );
                port.open();
            }
            if ( port.getPortName().equals(portName) )
                return;
            port.setPortName( portName );
        }
        public void deleted( String pid ) {
            SerialPortService port =
                (SerialPort) ports.get( pid );
            port.close();
            ports.remove( pid );
        }
    }
```

A Managed Service Factory must only be called if its bundle has the [<group>,UPDATED] Configuration Permission for the Configuration Object.

**Version:**

```
$Id: 107d3da81f09a25309cc8052073057afdd2d36e3 $
```
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>deleted(String pid)</code></td>
<td>Remove a factory instance.</td>
</tr>
<tr>
<td><code>getName()</code></td>
<td>Return a descriptive name of this factory.</td>
</tr>
<tr>
<td><code>updated(String pid, Dictionary properties)</code></td>
<td>Create a new instance, or update the configuration of an existing instance.</td>
</tr>
</tbody>
</table>

### Method Detail

#### getName

```java
String getName()
```

Return a descriptive name of this factory.

**Returns:** the name for the factory, which might be localized

#### updated

```java
void updated(String pid, Dictionary properties)
```

Create a new instance, or update the configuration of an existing instance. If the PID of the Configuration object is new for the Managed Service Factory, then create a new factory instance, using the configuration properties provided. Else, update the service instance with the provided properties.

If the factory instance is registered with the Framework, then the configuration properties should be copied to its registry properties. This is not mandatory and security sensitive properties should obviously not be copied.

If this method throws any `Exception`, the Configuration Admin service must catch it and should log it.

When the implementation of updated detects any kind of error in the configuration properties, it should create a new `ConfigurationException` which describes the problem.

The Configuration Admin service must call this method asynchronously. This implies that implementors of the ManagedServiceFactory class can be assured that the callback will not take place during registration when they execute the registration in a synchronized method.

**Parameters:**
- `pid`: The PID for this configuration.
- `properties`: A copy of the configuration properties. This argument must not contain the `service.bundleLocation` property. The value of this property may be obtained from the `Configuration.getBundleLocation` method.

**Throws:**
- `ConfigurationException`: when the configuration properties are invalid.

#### deleted

```java
void deleted(String pid)
```

Remove a factory instance. Remove the factory instance associated with the PID. If the instance was registered with the service registry, it should be unregistered.

If this method throws any `Exception`, the Configuration Admin service must catch it and should log it.
The Configuration Admin service must call this method asynchronously.

**Parameters:**
- `pid` - the PID of the service to be removed
Package org.osgi.service.configurable

Formatter Package Version 1.0.

See: Description

Interface Summary

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable&lt;T&gt;</td>
<td>The Configurable&lt;T&gt; interface is registered by any party that needs configuration settings of type &lt;T&gt;, where &lt;T&gt; is an interface that represents the type of the configuration properties.</td>
<td>48</td>
</tr>
<tr>
<td>ConfigurableFactory</td>
<td>A Configurable Factory can create Configurable objects.</td>
<td>50</td>
</tr>
<tr>
<td>Meta</td>
<td>The Meta interface provides access to the properties that underly a PID interface and their delta status.</td>
<td>51</td>
</tr>
</tbody>
</table>

Annotation Types Summary

<table>
<thead>
<tr>
<th>Annotation Type</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta.Default</td>
<td>If no value is supplied provide the given default value.</td>
<td>54</td>
</tr>
<tr>
<td>Meta.Description</td>
<td>Provide a description of this element.</td>
<td>55</td>
</tr>
<tr>
<td>Meta.Key</td>
<td>Use the given name instead of the method name to access the property.</td>
<td>56</td>
</tr>
<tr>
<td>Meta.Label</td>
<td>Provide a label for this property.</td>
<td>57</td>
</tr>
<tr>
<td>Meta.Mandatory</td>
<td>This property must have a value.</td>
<td>58</td>
</tr>
<tr>
<td>Meta.Options</td>
<td>Provide a list of options.</td>
<td>59</td>
</tr>
<tr>
<td>Meta.Pattern</td>
<td>A regular expression that must match the source property.</td>
<td>60</td>
</tr>
<tr>
<td>Meta.PID</td>
<td>Override the default PID name with a given PID name.</td>
<td>61</td>
</tr>
<tr>
<td>Meta.Range</td>
<td>Verifies that the source object follows the given range:</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>1. Number - The number must fall in the given range</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. String - The length of the string must fall with the range</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Collection - The size of the collection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Arrays - The length of the array</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The source object must first be converted to the return type before the check is done.</td>
<td>62</td>
</tr>
<tr>
<td>Meta.Separated</td>
<td>The value is a comma separated list.</td>
<td>63</td>
</tr>
</tbody>
</table>

Package org.osgi.service.configurable Description

Formatter Package Version 1.0.

The purpose of this package is to provide access to a Converter service. Bundles wishing to use this package must list the package in the Import-Package header of the bundle’s manifest. For example:

Consumers: Import-Package: org.osgi.service.configurable; version="[1.0,2.0)"
Providers: Import-Package: org.osgi.service.configurable; version="[1.0,1.1)"
Interface Configurable

org.osgi.service.configurable

Type Parameters:

T - The type of settings requested of the PID interface. The type T must be an interface, it can however, extend other interfaces, where these interfaces can extend other interfaces to create a composite of configuration objects.

public interface Configurable

The Configurable<T> interface is registered by any party that needs configuration settings of type <T>, where T is an interface that represents the type of the configuration properties. Each method on this interface is a configuration property name (or has some mapping to a configuration property). The Configurable service will be called with an instance of type T, where each method then is translated into a property lookup and appropriately converted. The Configurable interface can be both a service as well as being created by a Configurable Factory service. This maps to the concept of a Managed Service (Configurable) and a Managed Service Factory (Configurable Factory). By registering a Configurable service the registrar indicates it is in control of the cardinality, which is always 1. With a Configurable Factory, the cardinality is driven by the number of created Factory Configurations. A configuration type T can be backed by multiple PIDs. These PIDs are calculated from the T type's class information. The PIDs are an ordered list of all T and its extended interfaces, recursively. The order is depth first and in the order of declaration. For example, the following shows an example of a setup type:

```java
package com.example;
import com.example.sys.*;

public interface WebSetup extends SystemSetup {
    int port();
    String host();
}
```

The previous example uses a type T that is backed by two PIDs: com.example.WebSetup and com.example.sys.SystemSetup. If any of the two PIDs changes, the Configurable service is updated. For example, an Http service needs to be configured with its information. The Http Service implementation can then register the following Configurable service:

```java
Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>void deleted()</td>
<td>49</td>
</tr>
<tr>
<td>void setup(T settings)</td>
<td>48</td>
</tr>
</tbody>
</table>

Method Detail

setup

void setup(T settings)

Configure the Configurable with the given settings.

Parameters:

settings - The settings according to the required type.
deleted

void deleted()

Is called when no configuration is present when first registered or when the associated configuration is deleted.
Interface ConfigurableFactory

org.osgi.service.configurable

Type Parameters:

T - The type of interface used to map to the configuration properties. See Configurable for more information about this type.

public interface ConfigurableFactory

A Configurable Factory can create Configurable objects. Configurable objects represent some kind of task that needs some setup information. By registering a Configurable Factory service, a bundle can be used to create these Configurable objects under control of the Configuration Admin. A Configurable Factory service is associated by a generic parameter type T. This type T is the type used to configure the configurable, it also defines the factory PID and a number of associated PIDs.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable&lt;T&gt; newInstance()</td>
<td>Create a new instance of a Configurable object.</td>
</tr>
<tr>
<td>void noConfigurations()</td>
<td>Inform the factory that no configurations exist.</td>
</tr>
</tbody>
</table>

Method Detail

newInstance

Configurable&lt;T&gt; newInstance()

Create a new instance of a Configurable object. This object will be configured by the configurer with the Configuration Admin data as defined by its interface PIDs.

Returns:

A new Configurable object

noConfigurations

void noConfigurations()

Inform the factory that no configurations exist. This method is called at start up when no configurations can be found.
The Meta interface provides access to the properties that underly a PID interface and their delta status. Any PID interface can implement this interface. This interface will not count as a PID interface, that is, its PID will not map to a configuration record. Implementing this interface is optional.

### Nested Class Summary

<table>
<thead>
<tr>
<th>Nested Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>static @interface Meta.Default</td>
<td>54</td>
</tr>
<tr>
<td>If no value is supplied provide the given default value.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Description</td>
<td>55</td>
</tr>
<tr>
<td>Provide a description of this element.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Key</td>
<td>56</td>
</tr>
<tr>
<td>Use the given name instead of the method name to access the property.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Label</td>
<td>57</td>
</tr>
<tr>
<td>Provide a label for this property.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Mandatory</td>
<td>58</td>
</tr>
<tr>
<td>This property must have a value.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Options</td>
<td>59</td>
</tr>
<tr>
<td>Provide a list of options.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Pattern</td>
<td>60</td>
</tr>
<tr>
<td>A regular expression that must match the source property.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.PID</td>
<td>61</td>
</tr>
<tr>
<td>Override the default PID name with a given PID name.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Range</td>
<td>62</td>
</tr>
<tr>
<td>Verifies that the source object follows the given range:</td>
<td></td>
</tr>
<tr>
<td>1. Number - The number must fall in the given range</td>
<td></td>
</tr>
<tr>
<td>2. String - The length of the string must fall with the range</td>
<td></td>
</tr>
<tr>
<td>3. Collection - The size of the collection</td>
<td></td>
</tr>
<tr>
<td>4. Arrays - The length of the array</td>
<td></td>
</tr>
<tr>
<td>The source object must first be converted to the return type before the check is done.</td>
<td></td>
</tr>
<tr>
<td>static @interface Meta.Separated</td>
<td>63</td>
</tr>
<tr>
<td>The value is a comma separated list.</td>
<td></td>
</tr>
</tbody>
</table>

### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection&lt;String&gt; getChangedKeys()</td>
<td>52</td>
</tr>
<tr>
<td>Return a collection of property names of the properties that are changed from the previous callback.</td>
<td></td>
</tr>
<tr>
<td>String getFactoryPid()</td>
<td>53</td>
</tr>
<tr>
<td>Return the factory PID or null if this PID interface is not related to a factory configuration.</td>
<td></td>
</tr>
<tr>
<td>List&lt;String&gt; getPids()</td>
<td>52</td>
</tr>
<tr>
<td>Return an unmodifiable collection of the PIDs that are involved in the associated PID interface.</td>
<td></td>
</tr>
<tr>
<td>Map&lt;String,?&gt; getPreviousProperties()</td>
<td>52</td>
</tr>
<tr>
<td>Return the properties from the previous callback to the Configurable.setup(Object) method.</td>
<td></td>
</tr>
<tr>
<td>Map&lt;String,?&gt; getProperties()</td>
<td>52</td>
</tr>
<tr>
<td>Return the properties for a callback to the Configurable.setup(Object) method.</td>
<td></td>
</tr>
</tbody>
</table>
boolean `isChanged(String... names)`  
Return true if any of the given property names has a different value than the last callback.

### Method Detail

#### getProperties

```java
Map<String,?> getProperties()
```

Return the properties for a callback to the `Configurable.setup(Object)` method. The contents are the merged contents of all involved PIDs. The returned properties are not modifiable. The map is case insensitive.

**Returns:**  
the merged properties

#### getPreviousProperties

```java
Map<String,?> getPreviousProperties()
```

Return the properties from the previous callback to the `Configurable.setup(Object)` method. The contents are the merged contents of all involved PIDs. The returned properties are not modifiable. The map is case insensitive.

**Returns:**  
the merged properties of the previous callback

#### isChanged

```java
boolean isChanged(String... names)
```

Return true if any of the given property names has a different value than the last callback.

**Parameters:**  
- names - An array of property names

**Returns:**  
true if any of the referred properties is different from the previous callback.

#### getChangedKeys

```java
Collection<String> getChangedKeys()
```

Return a collection of property names of the properties that are changed from the previous callback.

**Returns:**  
A collection of changed property names

#### getPids

```java
List<String> getPids()
```

Return an unmodifiable collection of the PIDs that are involved in the associated PID interface.
Returns:
a collection of the pids that are involved in this PID interface.

getFactoryPid

String getFactoryPid()

Return the factory PID or null if this PID interface is not related to a factory configuration.

Returns:
The factory PID or null
Annotation Type Meta.Default

org.osgi.service.configurable

Enclosing class:

Meta

@Target(value=ElementType.METHOD)
@Retention(value=RetentionPolicy.RUNTIME)
public static @interface Meta.Default

If no value is supplied provide the given default value.

<table>
<thead>
<tr>
<th>Required Element Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String value</td>
<td>54</td>
</tr>
</tbody>
</table>

Element Detail

value

public abstract String value
Annotation Type Meta.Description

org.osgi.service.configurable

Enclosing class:
Meta

@Target(value=ElementType.METHOD)
@Retention(value=RetentionPolicy.RUNTIME)
public static @interface Meta.Description

Provide a description of this element.

<table>
<thead>
<tr>
<th>Required Element Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String value</td>
<td>55</td>
</tr>
</tbody>
</table>

Element Detail

value

public abstract String value
Annotation Type Meta.Key

org.osgi.service.configurable

Enclosing class:

Meta

@Target(value=ElementType.METHOD)
@Retention(value= RetentionPolicy.RUNTIME)
public static @interface Meta.Key

Use the given name instead of the method name to access the property. It is strongly recommended to use the automatic mapping to simplify understanding the system but in certain cases it can be beneficial to re-map the name

<table>
<thead>
<tr>
<th>Required Element Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String value</td>
<td>56</td>
</tr>
</tbody>
</table>

Element Detail

value

public abstract String value
Annotation Type Meta.Label

org.osgi.service.configurable

Enclosing class:

Meta

@Target(value=ElementType.METHOD)
@Retention(value=RetentionPolicy.RUNTIME)
public static @interface Meta.Label

Provide a label for this property.

---

### Required Element Summary

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Required</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>String</td>
<td></td>
<td>57</td>
</tr>
</tbody>
</table>

---

### Element Detail

value

class Meta {
  public abstract String value
}
Annotation Type Meta.Mandatory

@Target(value=ElementType.METHOD)
@Retention(value=RetentionPolicy.RUNTIME)
public static @interface Meta.Mandatory

This property must have a value. If this value is not set, the configuration is not delivered. Implementations should log if configurations are set where a mandatory field is absent.
Annotation Type `Meta.Options`

```java
class Meta {
    @Target(value=ElementType.METHOD)
    @Retention(value=RetentionPolicy.RUNTIME)
    public static @interface Meta.Options {
        // Methods for @interface
    }
}
```

Provide a list of options. The source property is converted to a string with the conversion model and then checked if it falls with the list of options.

### Required Element Summary

<table>
<thead>
<tr>
<th>Element</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>labels</td>
<td>59</td>
</tr>
<tr>
<td>value</td>
<td>59</td>
</tr>
</tbody>
</table>

### Element Detail

**value**

```java
public abstract String[] value
```

**labels**

```java
public abstract String[] labels
```

Default:

```java
{}
```
Annotation Type Meta.Pattern

Enclosing class:

<table>
<thead>
<tr>
<th>Required Element Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String[] value</td>
<td>60</td>
</tr>
</tbody>
</table>

A regular expression that must match the source property. A configuration must not be given to the configuree unless the pattern matches. To match the pattern it must be converted to a string first with the default conversion model.

public abstract String[] value
Annotation Type Meta.PID

```
org.osgi.service.configurable
```

Enclosing class:

```
Meta
```

```java
@Target(value=ElementType.TYPE)
@Retention(value=RetentionPolicy.RUNTIME)
public static @interface Meta.PID
```

Override the default PID name with a given PID name. The default PID name is the name of the class.

---

<table>
<thead>
<tr>
<th>Required Element Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>String value</td>
<td>61</td>
</tr>
</tbody>
</table>

**Element Detail**

**value**

```
public abstract String value
```
@Target(value=ElementType.METHOD)
@Retention(value=RetentionPolicy.RUNTIME)
public static @interface Meta.Range

Verifies that the source object follows the given range:

1. Number - The number must fall in the given range
2. String - The length of the string must fall within the range
3. Collection - The size of the collection
4. Arrays - The length of the array

The source object must first be converted to the return type before the check is done. The start() value is inclusive and the end() value is by default not inclusive: [start(), end()). The range can be made inclusive by setting inclusive=true.

<table>
<thead>
<tr>
<th>Required Element Summary</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>double end</td>
<td>62</td>
</tr>
<tr>
<td>double start</td>
<td>62</td>
</tr>
</tbody>
</table>

Element Detail

**start**

public abstract double start

Default: 0.0

**end**

public abstract double end

Default: 1.7976931348623157E308
Annotation Type Meta.Separated

Enclosing class:
Meta

@Target(value=ElementType.METHOD)
@Retention(value=RetentionPolicy.RUNTIME)
public static @interface Meta.Separated

The value is a comma separated list. If the property is not a string, it must be converted to a string first. In this conversion, arrays and collections must use the given separator in the join. The resulting string is converted to the target type that must be a collection or array.

Required Element Summary

<table>
<thead>
<tr>
<th>String</th>
<th>value</th>
</tr>
</thead>
</table>

Element Detail

value

public abstract String value

Default:
"\s*,\s*"

Java API documentation generated with DocFlex/Doclet v1.5.6.

DocFlex/Doclet is both a multi-format Javadoc doclet and a free edition of DocFlex/Javadoc. If you need to customize your Javadoc without writing a full-blown doclet from scratch, DocFlex/Javadoc may be the only tool able to help you! Find out more at www.docflex.com

9 Initial Spec Chapter

10 Considered Alternatives

See RFC 150 and RFC 144.
11 Security Considerations

Security issues have been discussed in the specifications.

12 Document Support

12.1 References

[3]. RFC 144 Config Admin Extensions https://www.osgi.org/members/RFC/0144

12.2 Author’s Address

Name: Peter Kriens
Company: aQute
Address: 9c, Avenue St. Drezery
Voice: +33-467542167
e-mail: Peter.Kriens@aQute.biz

Name: David Bosschaert
Company: Red Hat
Address: 6700 Cork Airport Business Park, Kinsale Road, Cork, Ireland
Voice: +353861704529
e-mail: david@redhat.com

12.3 Acronyms and Abbreviations

12.4 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.