Framework Update

Final

183 Pages

Abstract

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

0 Document Information

0.1 Table of Contents

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.

Revision	Date	Comments
Initial	11/07/09	Initial draft. BJ Hargrave
2nd draft	01/08/10	Updated based upon comments from CPEG. Added “adapt” model and redid PackageAdmin and StartLevel to exploit. Tracker API is also updated.
3rd draft	01/12/10	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.
4th draft	02/25/10	Minor javadoc updates from implementation experience.
5th draft	07/06/10	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.
6th draft	07/22/10	Added capabilities support for RFC 154. The changes now model exported packages and required bundles as capabilities.
7th draft	07/23/10	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.
8th draft	07/27/10	After CPEG meeting, removed FrameworkWiring methods getExportedPackages and getBundles. Corrected spelling of Capability.getAttributes method.
Final Draft	08/30/10	Updated Javadoc in preparation for RFC voting.

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 will 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 are compile time is very beneficial. Since generics only apply to compile time, they are “erased” at runtime. That is, at runtime, the types are “raw” and the generic information is not part of the Java type system.

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

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

J2ME Foundation 1.1 is based upon the Java 1.4 language and derives from the Java 1.4 SE class libraries.

ee.minimum is a subset of the Java 1.4 SE class libraries and the J2ME Foundation 1.1 class libraries.

3 Problem Description

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

There is pressure to being 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.mimimum

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

OSGi Javadoc

8/30/10 8:49 AM

Package Summary		Page
org.osgi.framework	Framework Package Version 1.6.	8
org.osgi.framework.startlevel	Framework Start Level Package Version 1.0.	143
org.osgi.framework.wiring	Framework Wiring Package Version 1.0.	149
org.osgi.util.tracker	Tracker Package Version 1.5.	164

Package org.osgi.framework

Framework Package Version 1.6.

See:
Description

Interface Summary		Page
AllServiceListener	A `ServiceEvent` listener that does not filter based upon package wiring.	16
Bundle	An installed bundle in the Framework.	17
BundleActivator	Customizes the starting and stopping of a bundle.	36
BundleContext	A bundle's execution context within the Framework.	38
BundleListener	A `BundleEvent` listener.	62
BundleReference	A reference to a Bundle.	67
Configurable	**Deprecated.** *As of 1.2.*	68
Constants	Defines standard names for the OSGi environment system properties, service properties, and Manifest header attribute keys.	69
Filter	An RFC 1960-based Filter.	99
FrameworkListener	A `FrameworkEvent` listener.	107
ServiceFactory	Allows services to provide customized service objects in the OSGi environment.	124
ServiceListener	A `ServiceEvent` listener.	126
ServiceReference	A reference to a service.	131
ServiceRegistration	A registered service.	135
SynchronousBundleListener	A synchronous `BundleEvent` listener.	137

Class Summary		Page
AdminPermission	A bundle's authority to perform specific privileged administrative operations on or to get sensitive information about a bundle.	10
BundleEvent	An event from the Framework describing a bundle lifecycle change.	52
BundlePermission	A bundle's authority to require or provide a bundle or to receive or attach fragments.	63
FrameworkEvent	A general event from the Framework.	102
FrameworkUtil	Framework Utility class.	108
PackagePermission	A bundle's authority to import or export a package.	114
ServiceEvent	An event from the Framework describing a service lifecycle change.	118
ServicePermission	A bundle's authority to register or get a service.	127
Version	Version identifier for bundles and packages.	138

Exception Summary		Page
BundleException	A Framework exception used to indicate that a bundle lifecycle problem occurred.	56
InvalidSyntaxException	A Framework exception used to indicate that a filter string has an invalid syntax.	111

ServiceException	A service exception used to indicate that a service problem occurred.	*121*

Package org.osgi.framework Description

Framework Package Version 1.6.

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)"
```

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:

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

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	Page
static String **CLASS** The action string `class`.	11
static String **CONTEXT** The action string `context`.	13
static String **EXECUTE** The action string `execute`.	12
static String **EXTENSIONLIFECYCLE** The action string `extensionLifecycle`.	12
static String **LIFECYCLE** The action string `lifecycle`.	12
static String **LISTENER** The action string `listener`.	12
static String **METADATA** The action string `metadata`.	12
static String **RESOLVE** The action string `resolve`.	12
static String **RESOURCE** The action string `resource`.	13
static String **STARTLEVEL** The action string `startlevel`.	13

Constructor Summary	Page
AdminPermission() Creates a new `AdminPermission` object that matches all bundles and has all actions.	13
AdminPermission(String filter, String actions) Create a new AdminPermission.	13
AdminPermission(Bundle bundle, String actions) Creates a new requested `AdminPermission` object to be used by the code that must perform `checkPermission`.	14

Method Summary	Page
boolean **equals**(Object obj) Determines the equality of two `AdminPermission` objects.	15
String **getActions**() Returns the canonical string representation of the `AdminPermission` actions.	14
int **hashCode**() Returns the hash code value for this object.	15
boolean **implies**(Permission p) Determines if the specified permission is implied by this object.	14
PermissionCollection **newPermissionCollection**() Returns a new `PermissionCollection` object suitable for storing `AdminPermission`s.	15

Field Detail

CLASS

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

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

Since:
1.3

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 `AdminPermission`s.

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

org.osgi.framework
All Superinterfaces:
EventListener, ServiceListener

```
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

Methods inherited from interface org.osgi.framework.ServiceListener
serviceChanged

Interface Bundle

org.osgi.framework
All Superinterfaces:
Comparable<Bundle>

```
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 `Bundle`s 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

Field Summary		Page
int	**ACTIVE** The bundle is now running.	20
int	**INSTALLED** The bundle is installed but not yet resolved.	19
int	**RESOLVED** The bundle is resolved and is able to be started.	19
int	**SIGNERS_ALL** Request that all certificates used to sign the bundle be returned.	21
int	**SIGNERS_TRUSTED** Request that only certificates used to sign the bundle that are trusted by the framework be returned.	21
int	**START_ACTIVATION_POLICY** The bundle start operation must activate the bundle according to the bundle's declared activation policy.	21

int	**START_TRANSIENT** The bundle start operation is transient and the persistent autostart setting of the bundle is not modified.	20
int	**STARTING** The bundle is in the process of starting.	20
int	**STOP_TRANSIENT** The bundle stop is transient and the persistent autostart setting of the bundle is not modified.	21
int	**STOPPING** The bundle is in the process of stopping.	20
int	**UNINSTALLED** The bundle is uninstalled and may not be used.	19

Method Summary		Page
A	**adapt**(Class<A> type) Adapt a bundle to the specifed type.	34
Enumeration<URL>	**findEntries**(String path, String filePattern, boolean recurse) Returns entries in this bundle and its attached fragments.	32
BundleContext	**getBundleContext**() Returns this bundle's BundleContext.	33
long	**getBundleId**() Returns this bundle's unique identifier.	27
File	**getDataFile**(String filename) Creates a File object for a file in the persistent storage area provided for this bundle by the Framework.	34
URL	**getEntry**(String path) Returns a URL to the entry at the specified path in this bundle.	32
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.	31
Dictionary<String,String>	**getHeaders**() Returns this bundle's Manifest headers and values.	27
Dictionary<String,String>	**getHeaders**(String locale) Returns this bundle's Manifest headers and values localized to the specified locale.	29
long	**getLastModified**() Returns the time when this bundle was last modified.	32
String	**getLocation**() Returns this bundle's location identifier.	28
ServiceReference<?>[]	**getRegisteredServices**() Returns this bundle's ServiceReference list for all services it has registered or null if this bundle has no registered services.	28
URL	**getResource**(String name) Find the specified resource from this bundle's class loader.	29
Enumeration<URL>	**getResources**(String name) Find the specified resources from this bundle's class loader.	31
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.	28
Map<X509Certificate, List<X509Certificate>>	**getSignerCertificates**(int signersType) Return the certificates for the signers of this bundle and the certificate chains for those signers.	34
int	**getState**() Returns this bundle's current state.	22

String	**getSymbolicName**() Returns the symbolic name of this bundle as specified by its Bundle-SymbolicName manifest header.	*30*
Version	**getVersion**() Returns the version of this bundle as specified by its Bundle-Version manifest header.	*34*
boolean	**hasPermission**(Object permission) Determines if this bundle has the specified permissions.	*29*
Class<?>	**loadClass**(String name) Loads the specified class using this bundle's class loader.	*30*
void	**start**() Starts this bundle with no options.	*23*
void	**start**(int options) Starts this bundle.	*22*
void	**stop**() Stops this bundle with no options.	*24*
void	**stop**(int options) Stops this bundle.	*24*
void	**uninstall**() Uninstalls this bundle.	*26*
void	**update**() Updates this bundle.	*26*
void	**update**(InputStream input) Updates this bundle from an InputStream.	*25*

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

```
public static final int STARTING = 8
```

The bundle is in the process of starting.

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

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

The value of STARTING is 0x00000008.

STOPPING

```
public static final int STOPPING = 16
```

The bundle is in the process of stopping.

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

The value of STOPPING is 0x00000010.

ACTIVE

```
public static final int ACTIVE = 32
```

The bundle is now running.

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

The value of ACTIVE is 0x00000020.

START_TRANSIENT

```
public static final int START_TRANSIENT = 1
```

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

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

Since:
1.4
See Also:
start(int)

START_ACTIVATION_POLICY

```
public static final int START_ACTIVATION_POLICY = 2
```

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

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

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

STOP_TRANSIENT

```
public static final int STOP_TRANSIENT = 1
```

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

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

Since:
1.4
See Also:
stop(int)

SIGNERS_ALL

```
public static final int SIGNERS_ALL = 1
```

Request that all certificates used to sign the bundle be returned.

Since:
1.5
See Also:
getSignerCertificates(int)

SIGNERS_TRUSTED

```
public static final int SIGNERS_TRUSTED = 2
```

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

Since:
1.5

See Also:
getSignerCertificates(int)

Method Detail

getState

```
int getState()
```

Returns this bundle's current state.

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

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

---

start

```
void start(int options)
    throws BundleException
```

Starts this bundle.

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

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

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

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

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

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

- A bundle event of type BundleEvent.STOPPING is fired.
- Any services registered by this bundle must be unregistered.
- Any services used by this bundle must be released.
- Any listeners registered by this bundle must be removed.
- This bundle's state is set to RESOLVED.
- A bundle event of type BundleEvent.STOPPED is fired.
- A BundleException is then thrown.

9. If this bundle's state is UNINSTALLED, because this bundle was uninstalled while the BundleActivator.start method was running, a BundleException is thrown.
10. This bundle's state is set to ACTIVE.
11. A bundle event of type BundleEvent.STARTED is fired.

Preconditions

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

Postconditions, no exceptions thrown

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

Postconditions, when an exception is thrown

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

Parameters:

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

Throws:

BundleException - If this bundle could not be started. This could be because a code dependency could not be resolved or the specified BundleActivator could not be loaded or threw an exception or this bundle is a fragment.

IllegalStateException - If this bundle has been uninstalled or this bundle tries to change its own state.

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

Since:

1.4

---

start

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

Starts this bundle with no options.

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

Throws:

BundleException - If this bundle could not be started. This could be because a code dependency could not be resolved or the specified BundleActivator could not be loaded or threw an exception or this bundle is a fragment.

IllegalStateException - If this bundle has been uninstalled or this bundle tries to change its own state.

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

See Also:

start(int)

---

stop

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

Stops this bundle.

The following steps are required to stop a bundle:

1. If this bundle's state is `UNINSTALLED` then an `IllegalStateException` is thrown.
2. If this bundle is in the process of being activated or deactivated then this method must wait for activation or deactivation to complete before continuing. If this does not occur in a reasonable time, a `BundleException` is thrown to indicate this bundle was unable to be stopped.
3. If the `STOP_TRANSIENT` option is not set then 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 `UNINSTALLED`, because this bundle was uninstalled while the `BundleActivator.stop` method was running, a `BundleException` must be thrown.
12. This bundle's state is set to `RESOLVED`.
13. A bundle event of type `BundleEvent.STOPPED` is fired.

Preconditions

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

Postconditions, no exceptions thrown

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

Postconditions, when an exception is thrown

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

Parameters:

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

Throws:

`BundleException` - If this bundle's `BundleActivator` threw an exception or this bundle is a fragment.

`IllegalStateException` - If this bundle has been uninstalled or this bundle tries to change its own state.

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

Since:

1.4

---

stop

```
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

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

Updates this bundle from an `InputStream`.

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

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

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

The following steps are required to update a bundle:

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

Preconditions

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

Postconditions, no exceptions thrown

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

Postconditions, when an exception is thrown

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

Parameters:

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

Throws:

`BundleException` - If the input stream cannot be read or the update fails.
`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

```
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

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

Uninstalls this bundle.

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

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

The following steps are required to uninstall a bundle:

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

Preconditions

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

Postconditions, no exceptions thrown

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

Postconditions, when an exception is thrown

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

Throws:

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

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 `instanceofjava.security.Permission`.

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

---

getResource

```
URL getResource(String name)
```

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

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

Parameters:
`name` - The name of the resource. See `ClassLoader.getResource` for a description of the format of a resource name.

Returns:
A URL to the named resource, or `null` if the resource could not be found or if this bundle is a fragment bundle or if the caller does not have the appropriate `AdminPermission[this,RESOURCE]`, and the Java Runtime Environment supports permissions.

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

Since:
1.1

See Also:
`getEntry()`, `findEntries()`

---

getHeaders

```
Dictionary<String,String> getHeaders(String locale)
```

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

This method performs the same function as `Bundle.getHeaders()` except the manifest header values are localized to the specified locale.

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

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

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

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

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

Returns:
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:

```
// 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 `X509Certificate`s for the signers of this bundle and the `X509Certificate` chains for those signers. The keys of the `Map` are the `X509Certificate`s 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 specifed 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 specifed type.

Since:
1.6

---

getDataFile

```
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

Interface BundleActivator

org.osgi.framework

```
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()`.

Version:
$Id: 1b73057bd270ab07f0a16430dba16e5132eea24f $
NotThreadSafe

Method Summary		Page
void	**start**(BundleContext context) Called when this bundle is started so the Framework can perform the bundle-specific activities necessary to start this bundle.	36
void	**stop**(BundleContext context) Called when this bundle is stopped so the Framework can perform the bundle-specific activities necessary to stop the bundle.	37

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.

---

stop

```
void stop(BundleContext context)
          throws Exception
```

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

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

Parameters:

`context` - The execution context of the bundle being stopped.

Throws:

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

Interface BundleContext

org.osgi.framework
All Superinterfaces:
BundleReference

```
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		Page
void	**addBundleListener**(BundleListener listener) Adds the specified `BundleListener` object to the context bundle's list of listeners if not already present.	43
void	**addFrameworkListener**(FrameworkListener listener) Adds the specified `FrameworkListener` object to the context bundle's list of listeners if not already present.	43
void	**addServiceListener**(ServiceListener listener) Adds the specified `ServiceListener` object to the context bundle's list of listeners.	42
void	**addServiceListener**(ServiceListener listener, String filter) Adds the specified `ServiceListener` object with the specified `filter` to the context bundle's list of listeners.	42
Filter	**createFilter**(String filter) Creates a `Filter` object.	50

ServiceReference<?>[]	**getAllServiceReferences**(String clazz, String filter) Returns an array of ServiceReference objects.	46
Bundle	**getBundle**() Returns the Bundle object associated with this BundleContext.	40
Bundle	**getBundle**(long id) Returns the bundle with the specified identifier.	41
Bundle[]	**getBundles**() Returns a list of all installed bundles.	41
File	**getDataFile**(String filename) Creates a File object for a file in the persistent storage area provided for the bundle by the Framework.	50
String	**getProperty**(String key) Returns the value of the specified property.	39
S	**getService**(ServiceReference<S> reference) Returns the service object referenced by the specified ServiceReference object.	49
ServiceReference<S>	**getServiceReference**(Class<S> clazz) Returns a ServiceReference object for a service that implements and was registered under the specified class.	47
ServiceReference<?>	**getServiceReference**(String clazz) Returns a ServiceReference object for a service that implements and was registered under the specified class.	47
Collection<ServiceReference<S>>	**getServiceReferences**(Class<S> clazz, String filter) Returns a collection of ServiceReference objects.	48
ServiceReference<?>[]	**getServiceReferences**(String clazz, String filter) Returns an array of ServiceReference objects.	46
Bundle	**installBundle**(String location) Installs a bundle from the specified location identifier.	41
Bundle	**installBundle**(String location, InputStream input) Installs a bundle from the specified InputStream object.	40
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.	45
ServiceRegistration<?>	**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.	45
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.	44
void	**removeBundleListener**(BundleListener listener) Removes the specified BundleListener object from the context bundle's list of listeners.	43
void	**removeFrameworkListener**(FrameworkListener listener) Removes the specified FrameworkListener object from the context bundle's list of listeners.	44
void	**removeServiceListener**(ServiceListener listener) Removes the specified ServiceListener object from the context bundle's list of listeners.	42
boolean	**ungetService**(ServiceReference<?> reference) Releases the service object referenced by the specified ServiceReference object.	49

Method Detail

getProperty

```
String getProperty(String key)
```

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

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

Parameters:
`key` - The name of the requested property.
Returns:
The value of the requested property, or `null` if the property is undefined.
Throws:
`SecurityException` - If the caller does not have the appropriate `PropertyPermission` to read the property, and the Java Runtime Environment supports permissions.

---

getBundle

```
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)
              throws BundleException
```

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

```
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==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 `ServiceEvent` s 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

```
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

```
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

`void` **`addBundleListener`**`(BundleListener listener)`

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

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

Parameters:
`listener` - The `BundleListener` to be added.
Throws:
`IllegalStateException` - If this BundleContext is no longer valid.
`SecurityException` - If listener is a `SynchronousBundleListener` and the caller does not have the appropriate `AdminPermission[context bundle,LISTENER]`, and the Java Runtime Environment supports permissions.
See Also:
`BundleEvent`, `BundleListener`

---

removeBundleListener

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

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

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

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

---

addFrameworkListener

`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==listener)`, this method does nothing.

Parameters:
`listener` - The `FrameworkListener` object to be added.
Throws:
`IllegalStateException` - If this BundleContext is no longer valid.
See Also:
`FrameworkEvent`, `FrameworkListener`

---

removeFrameworkListener

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

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

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

Parameters:
`listener` - The `FrameworkListener` object to be removed.
Throws:
`IllegalStateException` - If this BundleContext is no longer valid.

---

registerService

```
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 `instanceof` all the specified class names.
2. The Framework adds the following service properties to the service properties from the specified `Dictionary` (which may be `null`):
   A property named `Constants.SERVICE_ID` identifying the registration number of the service
   A property named `Constants.OBJECTCLASS` containing all the specified classes.
   Properties with these names in the specified `Dictionary` will be ignored.
3. The service is added to the Framework service registry and may now be used by other bundles.
4. A service event of type `ServiceEvent.REGISTERED` is fired.
5. A `ServiceRegistration` object for this registration is returned.

Parameters:
`clazzes` - The class names under which the service can be located. The class names in this array will be stored in the service's properties under the key `Constants.OBJECTCLASS`.
`service` - The service object or a `ServiceFactory` object.
`properties` - The properties for this service. The keys in the properties object must all be `String` objects. See `Constants` for a list of standard service property keys. Changes should not be made to this object after calling this method. To update the service's properties the `ServiceRegistration.setProperties()` method must be called. The set of properties may be `null` if the service has no properties.
Returns:
A `ServiceRegistration` object for use by the bundle registering the service to update the service's properties or to unregister the service.
Throws:
`IllegalArgumentException` - If one of the following is true:

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.

See Also:
ServiceRegistration, ServiceFactory

---

registerService

```
ServiceRegistration<?> 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.

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

Parameters:
clazz - The class name under which the service can be located.
service - The service object or a ServiceFactory object.
properties - The properties for this service.
Returns:
A ServiceRegistration object for use by the bundle registering the service to update the service's properties or to unregister the service.
Throws:
IllegalStateException - If this BundleContext is no longer valid.
See Also:
registerService(String[], Object, Dictionary)

---

registerService

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

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

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

Type Parameters:
S - Type of Service.
Parameters:
clazz - The class name under which the service can be located.
service - The service object or a ServiceFactory object.
properties - The properties for this service.
Returns:
A ServiceRegistration object for use by the bundle registering the service to update the service's properties or to unregister the service.
Throws:
IllegalStateException - If this BundleContext is no longer valid.
Since:
1.6
See Also:
registerService(String[], Object, Dictionary)

---

getServiceReferences

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

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

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

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

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

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

Parameters:

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

Returns:

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

Throws:

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

---

getAllServiceReferences

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

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

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

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

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

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

Parameters:
`clazz` - The class name with which the service was registered or `null` for all services.
`filter` - The filter expression or `null` for all services.
Returns:
An array of `ServiceReference` objects or `null` if no services are registered which satisfy the search.
Throws:
`InvalidSyntaxException` - If the specified `filter` contains an invalid filter expression that cannot be parsed.
`IllegalStateException` - If this BundleContext is no longer valid.
Since:
1.3

---

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

```
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 `ungetService(ServiceReference)` the context bundle's use count for that service is decremented by one.

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

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

The following steps are required to get the service object:

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

Type Parameters:

`S` - Type of Service.

Parameters:

`reference` - A reference to the service.

Returns:

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

Throws:

`SecurityException` - If the caller does not have the `ServicePermission` to get the service using at least one of the named classes the service was registered under and the Java Runtime Environment supports permissions.

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

`IllegalArgumentException` - If the specified `ServiceReference` was not created by the same framework instance as this `BundleContext`.

See Also:

`ungetService(ServiceReference)`, `ServiceFactory`

---

ungetService

`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`

---

getDataFile

```
File getDataFile(String filename)
```

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

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

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

Parameters:
`filename` - A relative name to the file to be accessed.
Returns:
A `File` object that represents the requested file or `null` if the platform does not have file system support.
Throws:
`IllegalStateException` - If this BundleContext is no longer valid.

---

createFilter

```
Filter createFilter(String filter)
        throws InvalidSyntaxException
```

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

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

Parameters:
`filter` - The filter string.
Returns:
A `Filter` object encapsulating the filter string.

Throws:

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

Since:

1.1

See Also:

"Framework specification for a description of the filter string syntax.", `FrameworkUtil.createFilter(String)`

Class BundleEvent

org.osgi.framework

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

All Implemented Interfaces:
Serializable

```
public class BundleEvent
extends EventObject
```

An event from the Framework describing a bundle lifecycle change.

`BundleEvent` objects are delivered to `SynchronousBundleListener`s and `BundleListener`s 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:
$Id: ac96d1ab8c5b491539af238ad41071b5bb9aef95 $
See Also:
BundleListener, SynchronousBundleListener
Immutable

Field Summary		Page
static int	**INSTALLED** The bundle has been installed.	53
static int	**LAZY_ACTIVATION** The bundle will be lazily activated.	55
static int	**RESOLVED** The bundle has been resolved.	54
static int	**STARTED** The bundle has been started.	53
static int	**STARTING** The bundle is about to be activated.	54
static int	**STOPPED** The bundle has been stopped.	53
static int	**STOPPING** The bundle is about to deactivated.	54
static int	**UNINSTALLED** The bundle has been uninstalled.	53
static int	**UNRESOLVED** The bundle has been unresolved.	54
static int	**UPDATED** The bundle has been updated.	53

Constructor Summary	Page
BundleEvent(int type, Bundle bundle) Creates a bundle event of the specified type.	55

Method Summary		Page
Bundle	**getBundle**() Returns the bundle which had a lifecycle change.	55
int	**getType**() Returns the type of lifecyle event.	55

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 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

```
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

```
public BundleEvent(int type,
                   Bundle bundle)
```

Creates a bundle event of the specified type.

Parameters:
`type` - The event type.
`bundle` - The bundle which had a lifecycle change.

Method Detail

getBundle

```
public Bundle getBundle()
```

Returns the bundle which had a lifecycle change. This bundle is the source of the event.

Returns:
The bundle that had a change occur in its lifecycle.

---

getType

```
public int getType()
```

Returns the type of lifecyle 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. `BundleException`s 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 $

Field Summary		Page
static int	**ACTIVATOR_ERROR** The bundle activator was in error.	58
static int	**DUPLICATE_BUNDLE_ERROR** The install or update operation failed because another already installed bundle has the same symbolic name and version.	59
static int	**INVALID_OPERATION** The operation was invalid.	57
static int	**MANIFEST_ERROR** The bundle manifest was in error.	58
static int	**NATIVECODE_ERROR** The bundle could not be resolved due to an error with the Bundle-NativeCode header.	58
static int	**RESOLVE_ERROR** The bundle was not resolved.	58
static int	**SECURITY_ERROR** The operation failed due to insufficient permissions.	58
static int	**START_TRANSIENT_ERROR** The start transient operation failed because the start level of the bundle is greater than the current framework start level	59
static int	**STATECHANGE_ERROR** The operation failed to complete the requested lifecycle state change.	58
static int	**UNSPECIFIED** No exception type is specified.	57
static int	**UNSUPPORTED_OPERATION** The operation was unsupported.	57

Constructor Summary	Page
BundleException(String msg) Creates a BundleException with the specified message.	59
BundleException(String msg, int type) Creates a BundleException with the specified message and type.	60
BundleException(String msg, int type, Throwable cause) Creates a BundleException with the specified message, type and exception cause.	59
BundleException(String msg, Throwable cause) Creates a BundleException with the specified message and exception cause.	59

	Method Summary	Page
Throwable	**getCause**() Returns the cause of this exception or null if no cause was set.	60
Throwable	**getNestedException**() Returns the cause of this exception or null if no cause was specified when this exception was created.	60
int	**getType**() Returns the type for this exception or UNSPECIFIED if the type was unspecified or unknown.	61
Throwable	**initCause**(Throwable cause) Initializes the cause of this exception to the specified value.	60

Field Detail

UNSPECIFIED

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

No exception type is specified.

Since:
1.5

UNSUPPORTED_OPERATION

```
public static final int UNSUPPORTED_OPERATION = 1
```

The operation was unsupported.

Since:
1.5

INVALID_OPERATION

```
public static final int INVALID_OPERATION = 2
```

The operation was invalid.

Since:
1.5

MANIFEST_ERROR

```
public static final int MANIFEST_ERROR = 3
```

The bundle manifest was in error.

Since:
1.5

RESOLVE_ERROR

```
public static final int RESOLVE_ERROR = 4
```

The bundle was not resolved.

Since:
1.5

ACTIVATOR_ERROR

```
public static final int ACTIVATOR_ERROR = 5
```

The bundle activator was in error.

Since:
1.5

SECURITY_ERROR

```
public static final int SECURITY_ERROR = 6
```

The operation failed due to insufficient permissions.

Since:
1.5

STATECHANGE_ERROR

```
public static final int STATECHANGE_ERROR = 7
```

The operation failed to complete the requested lifecycle state change.

Since:
1.5

NATIVECODE_ERROR

```
public static final int NATIVECODE_ERROR = 8
```

The bundle could not be resolved due to an error with the Bundle-NativeCode header.

Since:
1.5

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.

Since:
1.5

BundleException

```
public BundleException(String msg,
                       int type)
```

Creates a `BundleException` with the specified message and type.

Parameters:
`msg` - The message.
`type` - The type for this exception.
Since:
1.5

Method Detail

getNestedException

```
public Throwable getNestedException()
```

Returns the cause of this exception or `null` if no cause was specified when this exception was created.

This method predates the general purpose exception chaining mechanism. The `getCause()` method is now the preferred means of obtaining this information.

Returns:
The result of calling `getCause()`.

---

getCause

```
public Throwable getCause()
```

Returns the cause of this exception or `null` if no cause was set.

Overrides:
`getCause` in class `Throwable`
Returns:
The cause of this exception or `null` if no cause was set.
Since:
1.3

---

initCause

```
public Throwable initCause(Throwable cause)
```

Initializes the cause of this exception to the specified value.

Overrides:
`initCause` in class `Throwable`
Parameters:
`cause` - The cause of this exception.
Returns:
This exception.
Throws:
`IllegalArgumentException` - If the specified cause is this exception.
`IllegalStateException` - If the cause of this exception has already been set.

Since:
1.3

---

getType

```
public int getType()
```

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

Returns:
The type of this exception.
Since:
1.5

Interface BundleListener

org.osgi.framework
All Superinterfaces:
EventListener
All Known Subinterfaces:
SynchronousBundleListener

```
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. `BundleListener`s are called with a `BundleEvent` object when a bundle has been installed, resolved, started, stopped, updated, unresolved, or uninstalled.

Version:
$Id: 77cdaebd3ac97c6798fc3043957abd1bd6d01ccb $
See Also:
`BundleEvent`
NotThreadSafe

Method Summary		Page
void	**bundleChanged**(BundleEvent event) Receives notification that a bundle has had a lifecycle change.	62

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

```
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		Page
static String	**FRAGMENT** The action string `fragment`.	64
static String	**HOST** The action string `host`.	64
static String	**PROVIDE** The action string `provide`.	64
static String	**REQUIRE** The action string `require`.	64

Constructor Summary	Page
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		Page
boolean	**equals**(Object obj) Determines the equality of two `BundlePermission` objects.	65
String	**getActions**() Returns the canonical string representation of the `BundlePermission` actions.	65

int	**hashCode**() Returns the hash code value for this object.	*66*
boolean	**implies**(Permission p) Determines if the specified permission is implied by this object.	*65*
PermissionCollection	**newPermissionCollection**() Returns a new PermissionCollection object suitable for storing BundlePermission objects.	*65*

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 `BundlePermssion` 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

```
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

```
public String getActions()
```

Returns the canonical string representation of the `BundlePermission` actions.

Always returns present `BundlePermission` actions in the following order: `provide`, `require`, `host`, `fragment`.

Overrides:
`getActions` in class `BasicPermission`
Returns:
Canonical string representation of the `BundlePermission` actions.

---

newPermissionCollection

```
public PermissionCollection newPermissionCollection()
```

Returns a new `PermissionCollection` object suitable for storing `BundlePermission` objects.

Overrides:
`newPermissionCollection` in class `BasicPermission`
Returns:
A new `PermissionCollection` object.

---

equals

```
public boolean equals(Object obj)
```

Determines the equality of two `BundlePermission` objects. This method checks that specified bundle has the same bundle symbolic name and `BundlePermission` actions as this `BundlePermission` object.

Overrides:
`equals` in class `BasicPermission`

Parameters:
`obj` - The object to test for equality with this `BundlePermission` object.

Returns:
`true` if `obj` is a `BundlePermission`, and has the same bundle symbolic name and actions as this `BundlePermission` object; `false` otherwise.

---

hashCode

```
public int hashCode()
```

Returns the hash code value for this object.

Overrides:
`hashCode` in class `BasicPermission`

Returns:
A hash code value for this object.

Interface BundleReference

org.osgi.framework
All Known Subinterfaces:
BundleContext, BundleRevision, BundleStartLevel, BundleWiring, BundleWirings, FrameworkStartLevel, FrameworkWiring

---

```
public interface BundleReference
```

A reference to a Bundle.

Since:
1.5
Version:
$Id: adbba3dfc5ac9af0b534c3008ffab1b29984c4bb $
ThreadSafe

Method Summary		Page
Bundle	**getBundle**() Returns the Bundle object associated with this BundleReference.	67

Method Detail

getBundle

```
Bundle getBundle()
```

Returns the Bundle object associated with this BundleReference.

Returns:
The Bundle object associated with this BundleReference.

Interface Configurable

org.osgi.framework

`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 $

Method Summary		Page
`Object`	`getConfigurationObject()` **Deprecated.** *As of 1.2.*	68

Method Detail

getConfigurationObject

`Object` **`getConfigurationObject`**`()`

Deprecated. *As of 1.2. Please use Configuration Admin service.*

Returns this service's configuration object.

Services implementing `Configurable` should take care when returning a service configuration object since this object is probably sensitive.

If the Java Runtime Environment supports permissions, it is recommended that the caller is checked for some appropriate permission before returning the configuration object.

Returns:
The configuration object for this service.
Throws:
`SecurityException` - If the caller does not have an appropriate permission and the Java Runtime Environment supports permissions.

Interface Constants

org.osgi.framework

`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 $

Field Summary		Page
String	**ACTIVATION_LAZY** Bundle activation policy declaring the bundle must be activated when the first class load is made from the bundle.	86
String	**BUNDLE_ACTIVATIONPOLICY** Manifest header identifying the bundle's activation policy.	86
String	**BUNDLE_ACTIVATOR** Manifest header attribute identifying the bundle's activator class.	76
String	**BUNDLE_CATEGORY** Manifest header identifying the bundle's category.	74
String	**BUNDLE_CLASSPATH** Manifest header identifying a list of directories and embedded JAR files, which are bundle resources used to extend the bundle's classpath.	74
String	**BUNDLE_CONTACTADDRESS** Manifest header identifying the contact address where problems with the bundle may be reported; for example, an email address.	76
String	**BUNDLE_COPYRIGHT** Manifest header identifying the bundle's copyright information.	74
String	**BUNDLE_DESCRIPTION** Manifest header containing a brief description of the bundle's functionality.	74
String	**BUNDLE_DOCURL** Manifest header identifying the bundle's documentation URL, from which further information about the bundle may be obtained.	76
String	**BUNDLE_LOCALIZATION** Manifest header identifying the base name of the bundle's localization entries.	80
String	**BUNDLE_LOCALIZATION_DEFAULT_BASENAME** Default value for the `Bundle-Localization` manifest header.	80
String	**BUNDLE_MANIFESTVERSION** Manifest header identifying the bundle manifest version.	81
String	**BUNDLE_NAME** Manifest header identifying the bundle's name.	74
String	**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.	75
String	**BUNDLE_NATIVECODE_LANGUAGE** Manifest header attribute identifying the language in which the native bundle code is written specified in the Bundle-NativeCode manifest header.	78

String	**BUNDLE_NATIVECODE_OSNAME** Manifest header attribute identifying the operating system required to run native bundle code specified in the Bundle-NativeCode manifest header).	77
String	**BUNDLE_NATIVECODE_OSVERSION** Manifest header attribute identifying the operating system version required to run native bundle code specified in the Bundle-NativeCode manifest header).	78
String	**BUNDLE_NATIVECODE_PROCESSOR** Manifest header attribute identifying the processor required to run native bundle code specified in the Bundle-NativeCode manifest header).	77
String	**BUNDLE_REQUIREDEXECUTIONENVIRONMENT** Manifest header identifying the required execution environment for the bundle.	78
String	**BUNDLE_SYMBOLICNAME** Manifest header identifying the bundle's symbolic name.	78
String	**BUNDLE_SYMBOLICNAME_ATTRIBUTE** Manifest header attribute identifying the symbolic name of a bundle that exports a package specified in the Import-Package manifest header.	82
String	**BUNDLE_UPDATELOCATION** Manifest header identifying the location from which a new bundle version is obtained during a bundle update operation.	77
String	**BUNDLE_VENDOR** Manifest header identifying the bundle's vendor.	76
String	**BUNDLE_VERSION** Manifest header identifying the bundle's version.	76
String	**BUNDLE_VERSION_ATTRIBUTE** Manifest header attribute identifying a range of versions for a bundle specified in the `Require-Bundle` or `Fragment-Host` manifest headers.	81
String	**DYNAMICIMPORT_PACKAGE** Manifest header identifying the packages that the bundle may dynamically import during execution.	75
String	**EXCLUDE_DIRECTIVE** Manifest header directive identifying a list of classes to exclude in the exported package..	84
String	**EXPORT_PACKAGE** Manifest header identifying the packages that the bundle offers to the Framework for export.	75
String	**EXPORT_SERVICE** **Deprecated.** *As of 1.2.*	75
String	**EXTENSION_BOOTCLASSPATH** Manifest header directive value identifying the type of extension fragment.	86
String	**EXTENSION_DIRECTIVE** Manifest header directive identifying the type of the extension fragment.	85
String	**EXTENSION_FRAMEWORK** Manifest header directive value identifying the type of extension fragment.	86
String	**FRAGMENT_ATTACHMENT_ALWAYS** Manifest header directive value identifying a fragment attachment type of always.	79
String	**FRAGMENT_ATTACHMENT_DIRECTIVE** Manifest header directive identifying if and when a fragment may attach to a host bundle.	79
String	**FRAGMENT_ATTACHMENT_NEVER** Manifest header directive value identifying a fragment attachment type of never.	80
String	**FRAGMENT_ATTACHMENT_RESOLVETIME** Manifest header directive value identifying a fragment attachment type of resolve-time.	79
String	**FRAGMENT_HOST** Manifest header identifying the symbolic name of another bundle for which that the bundle is a fragment.	81
String	**FRAMEWORK_BEGINNING_STARTLEVEL** Specifies the beginning start level of the framework.	92

String	**FRAMEWORK_BOOTDELEGATION** Framework environment property identifying packages for which the Framework must delegate class loading to the parent class loader of the bundle.	88
String	**FRAMEWORK_BUNDLE_PARENT** Specifies the parent class loader type for all bundle class loaders.	92
String	**FRAMEWORK_BUNDLE_PARENT_APP** Specifies to use the application class loader as the parent class loader for all bundle class loaders.	93
String	**FRAMEWORK_BUNDLE_PARENT_BOOT** Specifies to use of the boot class loader as the parent class loader for all bundle class loaders.	93
String	**FRAMEWORK_BUNDLE_PARENT_EXT** Specifies to use the extension class loader as the parent class loader for all bundle class loaders.	93
String	**FRAMEWORK_BUNDLE_PARENT_FRAMEWORK** Specifies to use the framework class loader as the parent class loader for all bundle class loaders.	93
String	**FRAMEWORK_COMMAND_ABSPATH** Specified the substitution string for the absolute path of a file.	91
String	**FRAMEWORK_EXECPERMISSION** Specifies an optional OS specific command to set file permissions on extracted native code.	91
String	**FRAMEWORK_EXECUTIONENVIRONMENT** Framework environment property identifying execution environments provided by the Framework.	88
String	**FRAMEWORK_JARURLS** Specifies that an returned URLs from bundle class loaders must be a jar: or file: URL if set to any value.	95
String	**FRAMEWORK_LANGUAGE** Framework environment property identifying the Framework implementation language (see ISO 639 for possible values).	87
String	**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.	91
String	**FRAMEWORK_OS_NAME** Framework environment property identifying the Framework host-computer's operating system.	87
String	**FRAMEWORK_OS_VERSION** Framework environment property identifying the Framework host-computer's operating system version number.	87
String	**FRAMEWORK_PROCESSOR** Framework environment property identifying the Framework host-computer's processor name.	88
String	**FRAMEWORK_SECURITY** Specifies the type of security manager the framework must use.	90
String	**FRAMEWORK_SECURITY_OSGI** Specifies that a security manager that supports all security aspects of the OSGi core specification including postponed conditions must be installed.	90
String	**FRAMEWORK_STORAGE** Specified the persistent storage area used by the framework.	90
String	**FRAMEWORK_STORAGE_CLEAN** Specifies if and when the persistent storage area for the framework should be cleaned.	91
String	**FRAMEWORK_STORAGE_CLEAN_ONFIRSTINIT** Specifies that the framework storage area must be cleaned before the framework is initialized for the first time.	91
String	**FRAMEWORK_SYSTEMPACKAGES** Framework environment property identifying packages which the system bundle must export.	88

String	**FRAMEWORK_SYSTEMPACKAGES_EXTRA** Framework environment property identifying extra packages which the system bundle must export from the current execution environment.	88
String	**FRAMEWORK_TRUST_REPOSITORIES** Specifies the trust repositories used by the framework.	92
String	**FRAMEWORK_UUID** Framework environment property identifying the Framework's universally unique identifier (UUID).	95
String	**FRAMEWORK_VENDOR** Framework environment property identifying the Framework implementation vendor.	87
String	**FRAMEWORK_VERSION** Framework environment property identifying the Framework version.	87
String	**FRAMEWORK_WINDOWSYSTEM** Specifies the current windowing system.	92
String	**IMPORT_PACKAGE** Manifest header identifying the packages on which the bundle depends.	75
String	**IMPORT_SERVICE** **Deprecated.** *As of 1.2.*	76
String	**INCLUDE_DIRECTIVE** Manifest header directive identifying a list of classes to include in the exported package.	83
String	**MANDATORY_DIRECTIVE** Manifest header directive identifying names of matching attributes which must be specified by matching Import-Package statements in the Export-Package manifest header.	84
String	**OBJECTCLASS** Service property identifying all of the class names under which a service was registered in the Framework.	93
String	**PACKAGE_SPECIFICATION_VERSION** **Deprecated.** *As of 1.3.*	77
String	**REMOTE_CONFIGS_SUPPORTED** Service property identifying the configuration types supported by a distribution provider.	95
String	**REMOTE_INTENTS_SUPPORTED** Service property identifying the intents supported by a distribution provider.	95
String	**REQUIRE_BUNDLE** Manifest header identifying the symbolic names of other bundles required by the bundle.	80
String	**RESOLUTION_DIRECTIVE** Manifest header directive identifying the resolution type in the Import-Package or Require-Bundle manifest header.	82
String	**RESOLUTION_MANDATORY** Manifest header directive value identifying a mandatory resolution type.	83
String	**RESOLUTION_OPTIONAL** Manifest header directive value identifying an optional resolution type.	83
String	**SELECTION_FILTER_ATTRIBUTE** Manifest header attribute is used for selection by filtering based upon system properties.	81
String	**SERVICE_DESCRIPTION** Service property identifying a service's description.	95
String	**SERVICE_EXPORTED_CONFIGS** Service property identifying the configuration types that should be used to export the service.	96
String	**SERVICE_EXPORTED_INTENTS** Service property identifying the intents that the distribution provider must implement to distribute the service.	96
String	**SERVICE_EXPORTED_INTENTS_EXTRA** Service property identifying the extra intents that the distribution provider must implement to distribute the service.	96
String	**SERVICE_EXPORTED_INTERFACES** Service property marking the service for export.	97

String	**SERVICE_ID** Service property identifying a service's registration number.	94
String	**SERVICE_IMPORTED** Service property identifying the service as imported.	97
String	**SERVICE_IMPORTED_CONFIGS** Service property identifying the configuration types used to import the service.	97
String	**SERVICE_INTENTS** Service property identifying the intents that this service implement.	97
String	**SERVICE_PID** Service property identifying a service's persistent identifier.	94
String	**SERVICE_RANKING** Service property identifying a service's ranking number.	94
String	**SERVICE_VENDOR** Service property identifying a service's vendor.	94
String	**SINGLETON_DIRECTIVE** Manifest header directive identifying whether a bundle is a singleton.	79
String	**SUPPORTS_BOOTCLASSPATH_EXTENSION** Framework environment property identifying whether the Framework supports bootclasspath extension bundles.	89
String	**SUPPORTS_FRAMEWORK_EXTENSION** Framework environment property identifying whether the Framework supports framework extension bundles.	89
String	**SUPPORTS_FRAMEWORK_FRAGMENT** Framework environment property identifying whether the Framework supports fragment bundles.	89
String	**SUPPORTS_FRAMEWORK_REQUIREBUNDLE** Framework environment property identifying whether the Framework supports the `Require-Bundle` manifest header.	90
String	**SYSTEM_BUNDLE_LOCATION** Location identifier of the OSGi *system bundle* , which is defined to be "System Bundle".	73
String	**SYSTEM_BUNDLE_SYMBOLICNAME** Alias for the symbolic name of the OSGi *system bundle* .	74
String	**USES_DIRECTIVE** Manifest header directive identifying a list of packages that an exported package uses.	83
String	**VERSION_ATTRIBUTE** Manifest header attribute identifying the version of a package specified in the Export-Package or Import-Package manifest header.	82
String	**VISIBILITY_DIRECTIVE** Manifest header directive identifying the visibility of a required bundle in the Require-Bundle manifest header.	85
String	**VISIBILITY_PRIVATE** Manifest header directive value identifying a private visibility type.	85
String	**VISIBILITY_REEXPORT** Manifest header directive value identifying a reexport visibility type.	85

Field Detail

SYSTEM_BUNDLE_LOCATION

```
public static final String SYSTEM_BUNDLE_LOCATION = "System Bundle"
```

Location identifier of the OSGi *system bundle* , which is defined to be "System Bundle".

SYSTEM_BUNDLE_SYMBOLICNAME

```
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

BUNDLE_CATEGORY

```
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.

BUNDLE_CLASSPATH

```
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.

BUNDLE_COPYRIGHT

```
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.

BUNDLE_DESCRIPTION

```
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.

BUNDLE_NAME

```
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

```
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

```
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

```
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

```
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

```
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

```
public static final String BUNDLE_UPDATELOCATION = "Bundle-UpdateLocation"
```

Manifest header identifying the location from which a new bundle version is obtained during a bundle update operation.

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

PACKAGE_SPECIFICATION_VERSION

```
public static final String PACKAGE_SPECIFICATION_VERSION = "specification-version"
```

Deprecated.

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

BUNDLE_NATIVECODE_PROCESSOR

```
public static final String BUNDLE_NATIVECODE_PROCESSOR = "processor"
```

Manifest header attribute identifying the processor required to run native bundle code specified in the Bundle-NativeCode manifest header).

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

```
Bundle-NativeCode: http.so ; processor=x86 ...
```

See Also:
BUNDLE_NATIVECODE

BUNDLE_NATIVECODE_OSNAME

```
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

BUNDLE_NATIVECODE_OSVERSION

```
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_DIRECTIVE

FRAGMENT_ATTACHMENT_RESOLVETIME

```
public static final String FRAGMENT_ATTACHMENT_RESOLVETIME = "resolve-time"
```

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

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

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

Since:
1.3
See Also:
FRAGMENT_ATTACHMENT_DIRECTIVE

FRAGMENT_ATTACHMENT_NEVER

```
public static final String FRAGMENT_ATTACHMENT_NEVER = "never"
```

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

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

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

Since:
1.3
See Also:
FRAGMENT_ATTACHMENT_DIRECTIVE

BUNDLE_LOCALIZATION

```
public static final String BUNDLE_LOCALIZATION = "Bundle-Localization"
```

Manifest header identifying the base name of the bundle's localization entries.

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

Since:
1.3
See Also:
BUNDLE_LOCALIZATION_DEFAULT_BASENAME

BUNDLE_LOCALIZATION_DEFAULT_BASENAME

```
public static final String BUNDLE_LOCALIZATION_DEFAULT_BASENAME = "OSGI-INF/l10n/bundle"
```

Default value for the `Bundle-Localization` manifest header.

Since:
1.3
See Also:
BUNDLE_LOCALIZATION

REQUIRE_BUNDLE

```
public static final String REQUIRE_BUNDLE = "Require-Bundle"
```

Manifest header identifying the symbolic names of other bundles required by the bundle.

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

Since:
1.3

BUNDLE_VERSION_ATTRIBUTE

```
public static final String BUNDLE_VERSION_ATTRIBUTE = "bundle-version"
```

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

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

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

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

Since:
1.3
See Also:
REQUIRE_BUNDLE

FRAGMENT_HOST

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

Manifest header identifying the symbolic name of another bundle for which that the bundle is a fragment.

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

Since:
1.3

SELECTION_FILTER_ATTRIBUTE

```
public static final String SELECTION_FILTER_ATTRIBUTE = "selection-filter"
```

Manifest header attribute is used for selection by filtering based upon system properties.

The attribute value is encoded in manifest headers like:

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

Since:
1.3
See Also:
BUNDLE_NATIVECODE

BUNDLE_MANIFESTVERSION

```
public static final String BUNDLE_MANIFESTVERSION = "Bundle-ManifestVersion"
```

Manifest header identifying the bundle manifest version. A bundle manifest may express the version of the syntax in which it is written by specifying a bundle manifest version. Bundles exploiting OSGi Release 4, or later, syntax must specify a bundle manifest version.

The bundle manifest version defined by OSGi Release 4 or, more specifically, by version 1.3 of the OSGi Core Specification is "2".

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

Since:
1.3

VERSION_ATTRIBUTE

```
public static final String VERSION_ATTRIBUTE = "version"
```

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

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

```
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

See Also:
IMPORT_PACKAGE, REQUIRE_BUNDLE, RESOLUTION_MANDATORY, RESOLUTION_OPTIONAL

RESOLUTION_MANDATORY

```
public static final String RESOLUTION_MANDATORY = "mandatory"
```

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

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

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

Since:
1.3
See Also:
RESOLUTION_DIRECTIVE

RESOLUTION_OPTIONAL

```
public static final String RESOLUTION_OPTIONAL = "optional"
```

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

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

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

Since:
1.3
See Also:
RESOLUTION_DIRECTIVE

USES_DIRECTIVE

```
public static final String USES_DIRECTIVE = "uses"
```

Manifest header directive identifying a list of packages that an exported package uses.

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

```
Export-Package: org.osgi.util.tracker; uses:="org.osgi.framework"
```

Since:
1.3
See Also:
EXPORT_PACKAGE

INCLUDE_DIRECTIVE

```
public static final String INCLUDE_DIRECTIVE = "include"
```

Manifest header directive identifying a list of classes to include in the exported package.

This directive is used by the Export-Package manifest header to identify a list of classes of the specified package which must be allowed to be exported. The directive value is encoded in the Export-Package manifest header like:

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

This directive is also used by the Bundle-ActivationPolicy manifest header to identify the packages from which class loads will trigger lazy activation. The directive value is encoded in the Bundle-ActivationPolicy manifest header like:

```
Bundle-ActivationPolicy: lazy; include:="org.osgi.framework"
```

Since:
1.3
See Also:
EXPORT_PACKAGE, BUNDLE_ACTIVATIONPOLICY

EXCLUDE_DIRECTIVE

```
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"
```

Since:
1.3
See Also:
FRAGMENT_HOST, EXTENSION_FRAMEWORK, EXTENSION_BOOTCLASSPATH

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

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

```
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

```
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

```
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

```
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

```
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

```
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

```
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

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

```
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

```
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

```
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

```
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

See Also:
FRAMEWORK_BUNDLE_PARENT_BOOT, FRAMEWORK_BUNDLE_PARENT_EXT, FRAMEWORK_BUNDLE_PARENT_APP, FRAMEWORK_BUNDLE_PARENT_FRAMEWORK

FRAMEWORK_BUNDLE_PARENT_BOOT

```
public static final String FRAMEWORK_BUNDLE_PARENT_BOOT = "boot"
```

Specifies to use of the boot class loader as the parent class loader for all bundle class loaders.

Since:
1.5
See Also:
FRAMEWORK_BUNDLE_PARENT

FRAMEWORK_BUNDLE_PARENT_EXT

```
public static final String FRAMEWORK_BUNDLE_PARENT_EXT = "ext"
```

Specifies to use the extension class loader as the parent class loader for all bundle class loaders.

Since:
1.5
See Also:
FRAMEWORK_BUNDLE_PARENT

FRAMEWORK_BUNDLE_PARENT_APP

```
public static final String FRAMEWORK_BUNDLE_PARENT_APP = "app"
```

Specifies to use the application class loader as the parent class loader for all bundle class loaders. Depending on how the framework is launched, this may refer to the same class loader as 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

```
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

```
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

```
public static final String SERVICE_RANKING = "service.ranking"
```

Service property identifying a service's ranking number.

This property may be supplied in the `properties Dictionary` object passed to the `BundleContext.registerService` method. The value of this property must be of type `Integer`.

The service ranking is used by the Framework to determine the *natural order* of services, see `ServiceReference.compareTo()`, and the *default* service to be returned from a call to the `BundleContext.getServiceReference()` method.

The default ranking is zero (0). A service with a ranking of `Integer.MAX_VALUE` is very likely to be returned as the default service, whereas a service with a ranking of `Integer.MIN_VALUE` is very unlikely to be returned.

If the supplied property value is not of type `Integer`, it is deemed to have a ranking value of zero.

SERVICE_VENDOR

```
public static final String SERVICE_VENDOR = "service.vendor"
```

Service property identifying a service's vendor.

This property may be supplied in the properties `Dictionary` object passed to the `BundleContext.registerService` method.

SERVICE_DESCRIPTION

```
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 an 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` of `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` of `String`.

Since:
1.6
See Also:
"Remote Services Specification"

SERVICE_EXPORTED_INTENTS

```
public static final String SERVICE_EXPORTED_INTENTS = "service.exported.intents"
```

Service property identifying the intents that the distribution provider must implement to distribute the service. Intents listed in this property are reserved for intents that are critical for the code to function correctly, for example, ordering of messages. These intents should not be configurable.

This property may be supplied in the `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_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` of `String`.

Since:
1.6

See Also:
"Remote Services Specification"

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) 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"

Interface Filter

org.osgi.framework

```
public interface Filter
```

An RFC 1960-based Filter.

`Filter`s can be created by calling `BundleContext.createFilter()` or `FrameworkUtil.createFilter()` with a filter string.

A `Filter` can be used numerous times to determine if the match argument matches the filter string that was used to create the `Filter`.

Some examples of LDAP filters are:

```
"(cn=Babs Jensen)"
"(!(cn=Tim Howes))"
"(&(" + Constants.OBJECTCLASS + "=Person)(|(sn=Jensen)(cn=Babs 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."
ThreadSafe

Method Summary		Page
boolean	**equals**(Object obj) Compares this `Filter` to another `Filter`.	100
int	**hashCode**() Returns the hashCode for this `Filter`.	100
boolean	**match**(Dictionary<String,?> dictionary) Filter using a `Dictionary` with case insensitive key lookup.	100
boolean	**match**(ServiceReference<?> reference) Filter using a service's properties.	99
boolean	**matchCase**(Dictionary<String,?> dictionary) Filter using a `Dictionary`.	101
boolean	**matches**(Map<String,?> map) Filter using a `Map`.	101
String	**toString**() Returns this `Filter`'s filter string.	100

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

```
java.lang.Object
  └ java.util.EventObject
      └ org.osgi.framework.FrameworkEvent
```

All Implemented Interfaces:
Serializable

```
public class FrameworkEvent
extends EventObject
```

A general event from the Framework.

`FrameworkEvent` objects are delivered to `FrameworkListener`s 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

Field Summary		Page
static int	**ERROR** An error has occurred.	103
static int	**INFO** An informational event has occurred.	104
static int	**PACKAGES_REFRESHED** A PackageAdmin.refreshPackage operation has completed.	103
static int	**STARTED** The Framework has started.	103
static int	**STARTLEVEL_CHANGED** A StartLevel.setStartLevel operation has completed.	103
static int	**STOPPED** The Framework has stopped.	104
static int	**STOPPED_BOOTCLASSPATH_MODIFIED** The Framework has stopped and the boot class path has changed.	104
static int	**STOPPED_UPDATE** The Framework has stopped during update.	104
static int	**WAIT_TIMEDOUT** The Framework did not stop before the wait timeout expired.	105
static int	**WARNING** A warning has occurred.	104

Constructor Summary	Page
FrameworkEvent(int type, Object source) **Deprecated.** *As of 1.2.*	105
FrameworkEvent(int type, Bundle bundle, Throwable throwable) Creates a Framework event regarding the specified bundle.	105

Method Summary		Page
Bundle	**getBundle**() Returns the bundle associated with the event.	106
Throwable	**getThrowable**() Returns the exception related to this event.	105
int	**getType**() Returns the type of framework event.	106

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.

Since:
1.2

See Also:
"The Start Level Service"

WARNING

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

A warning has occurred.

There was a warning associated with a bundle.

Since:
1.3

INFO

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

An informational event has occurred.

There was an informational event associated with a bundle.

Since:
1.3

STOPPED

```
public static final int STOPPED = 64
```

The Framework has stopped.

This event is fired when the Framework has been stopped because of a stop operation on the system bundle. The source of this event is the System Bundle.

Since:
1.5

STOPPED_UPDATE

```
public static final int STOPPED_UPDATE = 128
```

The Framework has stopped during update.

This event is fired when the Framework has been stopped because of an update operation on the system bundle. The Framework will be restarted after this event is fired. The source of this event is the System Bundle.

Since:
1.5

STOPPED_BOOTCLASSPATH_MODIFIED

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

The Framework has stopped and the boot class path has changed.

This event is fired when the Framework has been stopped because of a stop operation on the system bundle and a bootclasspath extension bundle has been installed or updated. The source of this event is the System Bundle.

Since:
1.5

WAIT_TIMEDOUT

```
public static final int WAIT_TIMEDOUT = 512
```

The Framework did not stop before the wait timeout expired.

This event is fired when the Framework did not stop before the wait timeout expired. The source of this event is the System Bundle.

Since:
1.5

Constructor Detail

FrameworkEvent

```
public FrameworkEvent(int type,
                      Object source)
```

Deprecated.

Creates a Framework event.

Parameters:
`type` - The event type.
`source` - The event source object. This may not be `null`.

FrameworkEvent

```
public FrameworkEvent(int type,
                      Bundle bundle,
                      Throwable throwable)
```

Creates a Framework event regarding the specified bundle.

Parameters:
`type` - The event type.
`bundle` - The event source.
`throwable` - The related exception. This argument may be `null` if there is no related exception.

Method Detail

getThrowable

```
public Throwable getThrowable()
```

Returns the exception related to this event.

Returns:
The related exception or `null` if none.

---

getBundle

```
public Bundle getBundle()
```

Returns the bundle associated with the event. This bundle is also the source of the event.

Returns:
The bundle associated with the event.

---

getType

```
public int getType()
```

Returns the type of framework event.

The type values are:

- STARTED
- ERROR
- WARNING
- INFO
- PACKAGES_REFRESHED
- STARTLEVEL_CHANGED
- STOPPED
- STOPPED_BOOTCLASSPATH_MODIFIED
- STOPPED_UPDATE
- WAIT_TIMEDOUT

Returns:
The type of state change.

Interface FrameworkListener

org.osgi.framework
All Superinterfaces:
EventListener

```
public interface FrameworkListener
extends EventListener
```

A `FrameworkEvent` listener. `FrameworkListener` is a listener interface that may be implemented by a bundle developer. When a `FrameworkEvent` is fired, it is asynchronously delivered to a `FrameworkListener`. The Framework delivers `FrameworkEvent` objects to a `FrameworkListener` in order and must not concurrently call a `FrameworkListener`.

A `FrameworkListener` object is registered with the Framework using the `BundleContext.addFrameworkListener()` method. `FrameworkListener` objects are called with a `FrameworkEvent` objects when the Framework starts and when asynchronous errors occur.

Version:
$Id: a32e7599ea09d3510759d77e824cb8d9eff67f9d $
See Also:
`FrameworkEvent`
NotThreadSafe

Method Summary		Page
void	**frameworkEvent**(FrameworkEvent event) Receives notification of a general `FrameworkEvent` object.	107

Method Detail

frameworkEvent

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

Receives notification of a general `FrameworkEvent` object.

Parameters:
`event` - The `FrameworkEvent` object.

Class FrameworkUtil

org.osgi.framework

```
java.lang.Object
  └ 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		Page
static Filter	**createFilter**(String filter) Creates a Filter object.	108
static Bundle	**getBundle**(Class<?> classFromBundle) Return a Bundle for the specified bundle class.	110
static boolean	**matchDistinguishedNameChain**(String matchPattern, List<String> dnChain) Match a Distinguished Name (DN) chain against a pattern.	109

Method Detail

createFilter

```
public static Filter createFilter(String filter)
                           throws InvalidSyntaxException
```

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

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

This method returns a Filter implementation which may not perform as well as the framework implementation-specific Filter implementation returned by BundleContext.createFilter(String).

Parameters:
filter - The filter string.
Returns:
A Filter object encapsulating the filter string.
Throws:
InvalidSyntaxException - If filter contains an invalid filter string that cannot be parsed.
NullPointerException - If filter is null.
See Also:
Filter

---

matchDistinguishedNameChain

```
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. Dr√©z√©ry, o=ACME, c=US
dc=www, dc=acme, dc=com, o=ACME, c=US
o=ACME, c=US
```

The following DNs would not match:

```
street = 9C\, Avenue St. Dr√©z√©ry, o=ACME, c=FR
dc=www, dc=acme, dc=com, c=US
```

If a wildcard is used for a value of an RDN, the value must be exactly *. The wildcard must match any value, and no substring matching must be done. For example:

```
cn=*,o=ACME,c=*
```

This match pattern with wildcard must match the following DNs:

```
cn=Bugs Bunny,o=ACME,c=US
cn = Daffy Duck , o = ACME , c = US
cn=Road Runner, o=ACME, c=NL
```

But not:

```
o=ACME, c=NL
dc=acme.com, cn=Bugs Bunny, o=ACME, c=US
```

A match pattern may contain a chain of DN match patterns. The semicolon(';' \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	Page
InvalidSyntaxException(String msg, String filter) Creates an exception of type `InvalidSyntaxException`.	111
InvalidSyntaxException(String msg, String filter, Throwable cause) Creates an exception of type `InvalidSyntaxException`.	112

Method Summary		Page
Throwable	**getCause**() Returns the cause of this exception or `null` if no cause was set.	112
String	**getFilter**() Returns the filter string that generated the `InvalidSyntaxException` object.	112
Throwable	**initCause**(Throwable cause) Initializes the cause of this exception to the specified value.	112

Constructor Detail

InvalidSyntaxException

```
public InvalidSyntaxException(String msg,
                              String filter)
```

Creates an exception of type `InvalidSyntaxException`.

This method creates an `InvalidSyntaxException` object with the specified message and the filter string which generated the exception.

Parameters:
`msg` - The message.
`filter` - The invalid filter string.

InvalidSyntaxException

```
public InvalidSyntaxException(String msg,
                              String filter,
                              Throwable cause)
```

Creates an exception of type `InvalidSyntaxException`.

This method creates an `InvalidSyntaxException` object with the specified message and the filter string which generated the exception.

Parameters:
`msg` - The message.
`filter` - The invalid filter string.
`cause` - The cause of this exception.
Since:
1.3

Method Detail

getFilter

```
public String getFilter()
```

Returns the filter string that generated the `InvalidSyntaxException` object.

Returns:
The invalid filter string.
See Also:
`BundleContext.getServiceReferences()`, `BundleContext.addServiceListener(ServiceListener,String)`

---

getCause

```
public Throwable getCause()
```

Returns the cause of this exception or `null` if no cause was set.

Overrides:
`getCause` in class `Throwable`
Returns:
The cause of this exception or `null` if no cause was set.
Since:
1.3

---

initCause

```
public Throwable initCause(Throwable cause)
```

Initializes the cause of this exception to the specified value.

Overrides:
`initCause` in class `Throwable`
Parameters:
`cause` - The cause of this exception.
Returns:
This exception.
Throws:
`IllegalArgumentException` - If the specified cause is this exception.
`IllegalStateException` - If the cause of this exception has already been set.

Since:

1.3

Class PackagePermission

org.osgi.framework

```
java.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		Page
static String	**EXPORT** **Deprecated.** *Since 1.5.*	115
static String	**EXPORTONLY** The action string `exportonly`.	115
static String	**IMPORT** The action string `import`.	115

Constructor Summary	Page
PackagePermission(String name, String actions) Creates a new `PackagePermission` object.	115
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.	116

Method Summary		Page
boolean	**equals**(Object obj) Determines the equality of two `PackagePermission` objects.	117
String	**getActions**() Returns the canonical string representation of the `PackagePermission` actions.	117
int	**hashCode**() Returns the hash code value for this object.	117
boolean	**implies**(Permission p) Determines if the specified permission is implied by this object.	116

Permission Collection	**newPermissionCollection**() Returns a new PermissionCollection object suitable for storing PackagePermission objects.	117

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
javax.servlet.*
*
```

For the `import` action, the name can also be a filter expression. The filter gives access to the following attributes:

- signer - A Distinguished Name chain used to sign the exporting bundle. Wildcards in a DN are not matched according to the filter string rules, but according to the rules defined for a DN chain.
- 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 `PackagePermssion` 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

```
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

```
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 `PackagePermission` object; `false` otherwise.

---

hashCode

```
public int hashCode()
```

Returns the hash code value for this object.

Overrides:
`hashCode` in class `BasicPermission`
Returns:
A hash code value for this object.

Class ServiceEvent

org.osgi.framework

```
java.lang.Object
  └ java.util.EventObject
      └ org.osgi.framework.ServiceEvent
```

All Implemented Interfaces:
Serializable

```
public class ServiceEvent
extends EventObject
```

An event from the Framework describing a service lifecycle change.

`ServiceEvent` objects are delivered to `ServiceListener`s and `AllServiceListener`s 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

Field Summary	Page
static int **MODIFIED** The properties of a registered service have been modified.	119
static int **MODIFIED_ENDMATCH** The properties of a registered service have been modified and the new properties no longer match the listener's filter.	119
static int **REGISTERED** This service has been registered.	118
static int **UNREGISTERING** This service is in the process of being unregistered.	119

Constructor Summary	Page
ServiceEvent(int type, ServiceReference<?> reference) Creates a new service event object.	119

Method Summary	Page
ServiceReference<?> **getServiceReference**() Returns a reference to the service that had a change occur in its lifecycle.	120
int **getType**() Returns the type of event.	120

Field Detail

REGISTERED

```
public static final int REGISTERED = 1
```

This service has been registered.

This event is synchronously delivered **after** the service has been registered with the Framework.

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

MODIFIED

```
public static final int MODIFIED = 2
```

The properties of a registered service have been modified.

This event is synchronously delivered **after** the service properties have been modified.

See Also:
ServiceRegistration.setProperties()

UNREGISTERING

```
public static final int UNREGISTERING = 4
```

This service is in the process of being unregistered.

This event is synchronously delivered **before** the service has completed unregistering.

If a bundle is using a service that is `UNREGISTERING`, the bundle should release its use of the service when it receives this event. If the bundle does not release its use of the service when it receives this event, the Framework will automatically release the bundle's use of the service while completing the service unregistration operation.

See Also:
ServiceRegistration.unregister(), BundleContext.ungetService()

MODIFIED_ENDMATCH

```
public static final int MODIFIED_ENDMATCH = 8
```

The properties of a registered service have been modified and the new properties no longer match the listener's filter.

This event is synchronously delivered **after** the service properties have been modified. This event is only delivered to listeners which were added with a non-`null` filter where the filter matched the service properties prior to the modification but the filter does not match the modified service properties.

Since:
1.5
See Also:
ServiceRegistration.setProperties()

Constructor Detail

ServiceEvent

```
public ServiceEvent(int type,
                    ServiceReference<?> reference)
```

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

```
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

```
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 `ServiceException` object is created by the Framework or service implementation to denote an exception condition in the service. A type code is used to identify the exception type for future extendability. Service implementations may also create subclasses of `ServiceException`. When subclassing, the subclass should set the type to `SUBCLASSED` to indicate that `ServiceException` has been subclassed.

This exception conforms to the general purpose exception chaining mechanism.

Since:
1.5
Version:
$Id: 7eb3f12f99fa32b5a28ea318cea9faece24df0b5 $

Field Summary		Page
static int	**FACTORY_ERROR** The service factory produced an invalid service object.	122
static int	**FACTORY_EXCEPTION** The service factory threw an exception.	122
static int	**REMOTE** An error occurred invoking a remote service.	122
static int	**SUBCLASSED** The exception is a subclass of ServiceException.	122
static int	**UNREGISTERED** The service has been unregistered.	122
static int	**UNSPECIFIED** No exception type is unspecified.	122

Constructor Summary	Page
ServiceException(String msg) Creates a `ServiceException` with the specified message.	123
ServiceException(String msg, int type) Creates a `ServiceException` with the specified message and type.	123
ServiceException(String msg, int type, Throwable cause) Creates a `ServiceException` with the specified message, type and exception cause.	123
ServiceException(String msg, Throwable cause) Creates a `ServiceException` with the specified message and exception cause.	122

Method Summary		Page
int	**getType**() Returns the type for this exception or UNSPECIFIED if the type was unspecified or unknown.	123

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 getType()
```

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:
$Id: 6dad978a4354eedf8a4317b4aac37f2f2315d093 $
See Also:
`BundleContext.getService()`
ThreadSafe

Method Summary		Page
S	**getService**(Bundle bundle, ServiceRegistration<S> registration) Creates a new service object.	124
void	**ungetService**(Bundle bundle, ServiceRegistration<S> registration, S service) Releases a service object.	125

Method Detail

getService

```
S getService(Bundle bundle,
             ServiceRegistration<S> registration)
```

Creates a new service object.

The Framework invokes this method the first time the specified `bundle` requests a service object using the `BundleContext.getService(ServiceReference)` method. The service factory can then return a specific service object for each bundle.

The Framework caches the value returned (unless it is `null`), and will return the same service object on any future call to `BundleContext.getService` for the same bundle. This means the Framework must not allow this method to be concurrently called for the same bundle.

The Framework will check if the returned service object is an instance of all the classes named when the service was registered. If not, then `null` is returned to the bundle.

Parameters:
`bundle` - The bundle using the service.

`registration` - The `ServiceRegistration` object for the service.

Returns:

A service object that **must** be an instance of all the classes named when the service was registered.

See Also:

`BundleContext.getService()`

---

ungetService

```
void ungetService(Bundle bundle,
                  ServiceRegistration<S> registration,
                  S service)
```

Releases a service object.

The Framework invokes this method when a service has been released by a bundle. The service object may then be destroyed.

Parameters:

`bundle` - The bundle releasing the service.

`registration` - The `ServiceRegistration` object for the service.

`service` - The service object returned by a previous call to the `ServiceFactory.getService` method.

See Also:

`BundleContext.ungetService()`

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		Page
void	**serviceChanged**(ServiceEvent event) Receives notification that a service has had a lifecycle change.	126

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: 1f94e06913cdf0eaa24f90925290df1215f3d5ff $
ThreadSafe

Field Summary		Page
static String	**GET** The action string `get`.	128
static String	**REGISTER** The action string `register`.	128

Constructor Summary	Page
ServicePermission(String name, String actions) Create a new ServicePermission.	128
ServicePermission(ServiceReference<?> reference, String actions) Creates a new requested `ServicePermission` object to be used by code that must perform `checkPermission` for the `get` action.	129

Method Summary		Page
boolean	**equals**(Object obj) Determines the equality of two ServicePermission objects.	129
String	**getActions**() Returns the canonical string representation of the actions.	129
int	**hashCode**() Returns the hash code value for this object.	130
boolean	**implies**(Permission p) Determines if a `ServicePermission` object "implies" the specified permission.	129
PermissionCollection	**newPermissionCollection**() Returns a new `PermissionCollection` object for storing `ServicePermission` objects.	129

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.

ServicePermission

```
public ServicePermission(ServiceReference<?> reference,
                         String actions)
```

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

Parameters:
`reference` - The requested service.
`actions` - The action `get`.
Throws:
`IllegalArgumentException` - If the specified action is not `get` or reference is `null`.
Since:
1.5

Method Detail

implies

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

Determines if a `ServicePermission` object "implies" the specified permission.

Overrides:
`implies` in class `BasicPermission`
Parameters:
`p` - The target permission to check.
Returns:
`true` if the specified permission is implied by this object; `false` otherwise.

---

getActions

```
public String getActions()
```

Returns the canonical string representation of the actions. Always returns present actions in the following order: `get`, `register`.

Overrides:
`getActions` in class `BasicPermission`
Returns:
The canonical string representation of the actions.

---

newPermissionCollection

```
public PermissionCollection newPermissionCollection()
```

Returns a new `PermissionCollection` object for storing `ServicePermission` objects.

Overrides:
`newPermissionCollection` in class `BasicPermission`
Returns:
A new `PermissionCollection` object suitable for storing `ServicePermission` objects.

---

equals

```
public boolean equals(Object obj)
```

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

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

---

hashCode

```
public int hashCode()
```

Returns the hash code value for this object.

Overrides:
`hashCode` in class `BasicPermission`
Returns:
Hash code value for this object.

Interface ServiceReference

org.osgi.framework

Type Parameters:
S - Type of Service.
All Superinterfaces:
Comparable<Object>

```
public interface ServiceReference
extends Comparable<Object>
```

A reference to a service.

The Framework returns `ServiceReference` objects from the `BundleContext.getServiceReference` and `BundleContext.getServiceReferences` methods.

A `ServiceReference` object may be shared between bundles and can be used to examine the properties of the service and to get the service object.

Every service registered in the Framework has a unique `ServiceRegistration` object and may have multiple, distinct `ServiceReference` objects referring to it. `ServiceReference` objects associated with a `ServiceRegistration` object have the same `hashCode` and are considered equal (more specifically, their `equals()` method will return `true` when compared).

If the same service object is registered multiple times, `ServiceReference` objects associated with different `ServiceRegistration` objects are not equal.

Version:
$Id: 10e6b437fdf20dea9d8327f8135634a48a9dfd88 $
See Also:
BundleContext.getServiceReference(), BundleContext.getServiceReferences(), BundleContext.getService()
ThreadSafe

Method Summary		Page
int	**compareTo**(Object reference) Compares this `ServiceReference` with the specified `ServiceReference` for order.	133
Bundle	**getBundle**() Returns the bundle that registered the service referenced by this `ServiceReference` object.	132
Object	**getProperty**(String key) Returns the property value to which the specified property key is mapped in the properties `Dictionary` object of the service referenced by this `ServiceReference` object.	132
String[]	**getPropertyKeys**() Returns an array of the keys in the properties `Dictionary` object of the service referenced by this `ServiceReference` object.	132
Bundle[]	**getUsingBundles**() Returns the bundles that are using the service referenced by this `ServiceReference` object.	132
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.	133

Method Detail

getProperty

```
Object getProperty(String key)
```

Returns the property value to which the specified property key is mapped in the properties `Dictionary` object of the service referenced by this `ServiceReference` object.

Property keys are case-insensitive.

This method must continue to return property values after the service has been unregistered. This is so references to unregistered services (for example, `ServiceReference` objects stored in the log) can still be interrogated.

Parameters:
`key` - The property key.
Returns:
The property value to which the key is mapped; `null` if there is no property named after the key.

---

getPropertyKeys

```
String[] getPropertyKeys()
```

Returns an array of the keys in the properties `Dictionary` object of the service referenced by this `ServiceReference` object.

This method will continue to return the keys after the service has been unregistered. This is so references to unregistered services (for example, `ServiceReference` objects stored in the log) can still be interrogated.

This method is *case-preserving* ; this means that every key in the returned array must have the same case as the corresponding key in the properties `Dictionary` that was passed to the `BundleContext.registerService(String[],Object,Dictionary)` or `ServiceRegistration.setProperties()` methods.

Returns:
An array of property keys.

---

getBundle

```
Bundle getBundle()
```

Returns the bundle that registered the service referenced by this `ServiceReference` object.

This method must return `null` when the service has been unregistered. This can be used to determine if the service has been unregistered.

Returns:
The bundle that registered the service referenced by this `ServiceReference` object; `null` if that service has already been unregistered.
See Also:
`BundleContext.registerService(String[],Object,Dictionary)`

---

getUsingBundles

```
Bundle[] getUsingBundles()
```

Returns the bundles that are using the service referenced by this `ServiceReference` object. Specifically, this method returns the bundles whose usage count for that service is greater than zero.

Returns:
An array of bundles whose usage count for the service referenced by this `ServiceReference` object is greater than zero; `null` if no bundles are currently using that service.

Since:
1.1

---

isAssignableTo

```
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

```
int compareTo(Object reference)
```

Compares this `ServiceReference` with the specified `ServiceReference` for order.

If this `ServiceReference` and the specified `ServiceReference` have the same `service.id` they are equal. This `ServiceReference` is less than the specified `ServiceReference` if it has a lower `service.ranking` and greater if it has a higher service ranking. Otherwise, if this `ServiceReference` and the specified `ServiceReference` have the same `service.ranking`, this `ServiceReference` is less than the specified `ServiceReference` if it has a higher `service.id` and greater if it has a lower service id.

Specified by:
`compareTo` in interface `Comparable`

Parameters:
`reference` - The `ServiceReference` to be compared.

Returns:
Returns a negative integer, zero, or a positive integer if this `ServiceReference` is less than, equal to, or greater than the specified `ServiceReference`.

Throws:
`IllegalArgumentException` - If the specified `ServiceReference` was not created by the same framework instance as this `ServiceReference`.

Since:

1.4

Interface ServiceRegistration

org.osgi.framework

Type Parameters:
S - Type of Service.

---

public interface **ServiceRegistration**

A registered service.

The Framework returns a ServiceRegistration object when a BundleContext.registerService method invocation is successful. The ServiceRegistration object is for the private use of the registering bundle and should not be shared with other bundles.

The ServiceRegistration object may be used to update the properties of the service or to unregister the service.

Version:
$Id: 6487c568cab6629edb05feaf86d83caadf9acf4e $
See Also:
BundleContext.registerService(String[],Object,Dictionary)
ThreadSafe

Method Summary		Page
ServiceReference<S>	**getReference**() Returns a ServiceReference object for a service being registered.	135
void	**setProperties**(Dictionary<String,?> properties) Updates the properties associated with a service.	135
void	**unregister**() Unregisters a service.	136

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

`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

org.osgi.framework
All Superinterfaces:
BundleListener, EventListener

```
public interface SynchronousBundleListener
extends BundleListener
```

A synchronous `BundleEvent` listener. `SynchronousBundleListener` is a listener interface that may be implemented by a bundle developer. When a `BundleEvent` is fired, it is synchronously delivered to a `SynchronousBundleListener`. The Framework may deliver `BundleEvent` objects to a `SynchronousBundleListener` out of order and may concurrently call and/or reenter a `SynchronousBundleListener`.

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, `SynchronousBundleListener`s are synchronously called during bundle lifecycle processing. The bundle lifecycle processing will not proceed until all `SynchronousBundleListener`s 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

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.

- Major version. A non-negative integer.
- Minor version. A non-negative integer.
- Micro version. A non-negative integer.
- Qualifier. A text string. See `Version(String)` for the format of the qualifier string.

`Version` objects are immutable.

Since:
1.3
Version:
$Id: 6b36c5c1ac6ff508fca81eedc6a25c20dfbf00ce $
Immutable

Field Summary		Page
static Version	**emptyVersion** The empty version "0.0.0".	139

Constructor Summary	Page
Version(int major, int minor, int micro) Creates a version identifier from the specified numerical components.	139
Version(int major, int minor, int micro, String qualifier) Creates a version identifier from the specified components.	139
Version(String version) Created a version identifier from the specified string.	140

Method Summary		Page
int	**compareTo**(Version other) Compares this `Version` object to another object.	142
boolean	**equals**(Object object) Compares this `Version` object to another object.	141
int	**getMajor**() Returns the major component of this version identifier.	140
int	**getMicro**() Returns the micro component of this version identifier.	141
int	**getMinor**() Returns the minor component of this version identifier.	140

String	**getQualifier**() Returns the qualifier component of this version identifier.	*141*
int	**hashCode**() Returns a hash code value for the object.	*141*
static Version	**parseVersion**(String version) Parses a version identifier from the specified string.	*140*
String	**toString**() Returns the string representation of this version identifier.	*141*

Field Detail

emptyVersion

```
public static final Version emptyVersion
```

The empty version "0.0.0".

Constructor Detail

Version

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

Creates a version identifier from the specified numerical components.

The qualifier is set to the empty string.

Parameters:
`major` - Major component of the version identifier.
`minor` - Minor component of the version identifier.
`micro` - Micro component of the version identifier.
Throws:
`IllegalArgumentException` - If the numerical components are negative.

Version

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

Creates a version identifier from the specified components.

Parameters:
`major` - Major component of the version identifier.
`minor` - Minor component of the version identifier.
`micro` - Micro component of the version identifier.
`qualifier` - Qualifier component of the version identifier. If `null` is specified, then the qualifier will be set to the empty string.
Throws:
`IllegalArgumentException` - If the numerical components are negative or the qualifier string is invalid.

Version

```
public Version(String version)
```

Created a version identifier from the specified string.

Here is the grammar for version strings.

```
version ::= major('.'minor('.'micro('.'qualifier)?)?)?
major ::= digit+
minor ::= digit+
micro ::= digit+
qualifier ::= (alpha|digit|'_'|'-')+
digit ::= [0..9]
alpha ::= [a..zA..Z]
```

There must be no whitespace in version.

Parameters:
`version` - String representation of the version identifier.
Throws:
`IllegalArgumentException` - If `version` is improperly formatted.

Method Detail

parseVersion

```
public static Version parseVersion(String version)
```

Parses a version identifier from the specified string.

See `Version(String)` for the format of the version string.

Parameters:
`version` - String representation of the version identifier. Leading and trailing whitespace will be ignored.
Returns:
A `Version` object representing the version identifier. If `version` is `null` or the empty string then `emptyVersion` will be returned.
Throws:
`IllegalArgumentException` - If `version` is improperly formatted.

---

getMajor

```
public int getMajor()
```

Returns the major component of this version identifier.

Returns:
The major component.

---

getMinor

```
public int getMinor()
```

Returns the minor component of this version identifier.

Returns:
The minor component.

getMicro

```
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](#)

Interface Summary		Page
BundleStartLevel	Query and modify the start level information for a bundle.	*144*
FrameworkStartLevel	Query and modify the start level information for the framework.	*146*

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)"
```

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 `bundle.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		Page
int	**getStartLevel**() Return the assigned start level value for the bundle.	144
boolean	**isActivationPolicyUsed**() Returns whether the bundle's autostart setting indicates that the activation policy declared in the bundle manifest must be used.	145
boolean	**isPersistentlyStarted**() Returns whether the bundle's autostart setting indicates it must be started.	145
void	**setStartLevel**(int startlevel) Assign a start level value to the bundle.	144

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

Method Detail

getStartLevel

```
int getStartLevel()
```

Return the assigned start level value for the bundle.

Returns:
The start level value of the bundle.
Throws:
`IllegalStateException` - If the bundle has been uninstalled.
See Also:
`setStartLevel(int)`

---

setStartLevel

```
void setStartLevel(int startlevel)
```

Assign a start level value to the bundle.

The bundle will be assigned the specified start level. The start level value assigned to the bundle will be persistently recorded by the Framework.

If the new start level for the bundle is lower than or equal to the active start level of the Framework and the bundle's autostart setting indicates this bundle must be started, the Framework will start the bundle as described in the `Bundle.start(int)` method using the `Bundle.START_TRANSIENT` option. The `Bundle.START_ACTIVATION_POLICY` option must also be used if `isActivationPolicyUsed()` returns `true`. The actual starting of the bundle must occur asynchronously.

If the new start level for the bundle is higher than the active start level of the Framework, the Framework will stop the bundle as described in the `Bundle.stop(int)` method using the `Bundle.STOP_TRANSIENT` option. The actual stopping of the bundle must occur asynchronously.

Parameters:
`startlevel` - The new start level for the bundle.
Throws:
`IllegalArgumentException` - If the specified start level is less than or equal to zero, or if the bundle is the system bundle.
`IllegalStateException` - If the bundle has been uninstalled.
`SecurityException` - If the caller does not have `AdminPermission[bundle,EXECUTE]` and the Java runtime environment supports permissions.

---

isPersistentlyStarted

`boolean` **`isPersistentlyStarted`**`()`

Returns whether the bundle's autostart setting indicates it must be started.

The autostart setting of a bundle indicates whether the bundle is to be started when its start level is reached.

Returns:
`true` if the autostart setting of the bundle indicates it is to be started. `false` otherwise.
Throws:
`IllegalStateException` - If this bundle has been uninstalled.
See Also:
`Bundle.START_TRANSIENT`

---

isActivationPolicyUsed

`boolean` **`isActivationPolicyUsed`**`()`

Returns whether the bundle's autostart setting indicates that the activation policy declared in the bundle manifest must be used.

The autostart setting of a bundle indicates whether the bundle's declared activation policy is to be used when the bundle is started.

Returns:
`true` if the bundle's autostart setting indicates the activation policy declared in the manifest must be used. `false` if the bundle must be eagerly activated.
Throws:
`IllegalStateException` - If the bundle has been uninstalled.
See Also:
`Bundle.START_ACTIVATION_POLICY`

Interface FrameworkStartLevel

org.osgi.framework.startlevel
All Superinterfaces:
BundleReference

```
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		Page
int	**getInitialBundleStartLevel**() Return the initial start level value that is assigned to a Bundle when it is first installed.	147
int	**getStartLevel**() Return the active start level value of the Framework.	146
void	**setInitialBundleStartLevel**(int startlevel) Set the initial start level value that is assigned to a Bundle when it is first installed.	147
void	**setStartLevel**(int startlevel, FrameworkListener... listeners) Modify the active start level of the Framework and notify when complete.	146

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[SystemBundle,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

Interface Summary		Page
BundleRevision	Bundle Revision.	150
BundleWiring	A wiring for a bundle.	152
BundleWirings	The in use bundle wirings for a bundle.	157
Capability	A capability that has been provided from a bundle wiring .	158
FrameworkWiring	Query and modify wiring information for the framework.	161

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:
BundleReference

```
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).

Version:
$Id: 85e0fa85db8c90aa6e44daa836299d154ef9b6b6 $
ThreadSafe

Field Summary		Page
int	**TYPE_FRAGMENT** Bundle revision type indicating the bundle revision is a fragment.	150

Method Summary		Page
String	**getSymbolicName**() Returns the symbolic name for this bundle revision.	150
int	**getTypes**() Returns the special types of this bundle revision.	151
Version	**getVersion**() Returns the version for this bundle revision.	151

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()
```

Returns the symbolic name for this bundle revision.

Returns:
The symbolic name for this bundle revision.

See Also:
Bundle.getSymbolicName()

---

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()

---

getTypes

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.

Interface BundleWiring

org.osgi.framework.wiring
All Superinterfaces:
BundleReference

```
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		Page
int	**FINDENTIRES_RECURSE** The find entries operation must recurse into subdirectories.	153
int	**LISTRESOURCES_LOCAL** 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`.	153
int	**LISTRESOURCES_RECURSE** The list resource names operation must recurse into subdirectories.	153

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

boolean	**isInUse**() Returns `true` if this bundle wiring is in use.	154
List<String>	**listResources**(String path, String filePattern, int options) Returns the names of resources visible to this bundle wiring's class loader.	156

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

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 `Capability`s, 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 `Capability`s 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 `FINDENTIRES_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.

Interface BundleWirings

org.osgi.framework.wiring
All Superinterfaces:
BundleReference

```
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().

Version:
$Id: dc8656f9ab4562e4eda1c461c1b3414e5743515e $
ThreadSafe

Method Summary	Page
List<BundleWiring> **getWirings**() Return the in use wirings for the referenced bundle.	157

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.

Interface Capability

org.osgi.framework.wiring

```
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		Page
String	**BUNDLE_CAPABILITY** Capability name space for bundle capabilities.	159
String	**PACKAGE_CAPABILITY** Capability name space for package capabilities.	158

Method Summary		Page
Map<String,Object>	**getAttributes**() Returns the attributes of this capability.	159
Map<String,String>	**getDirectives**() Returns the directives of this capability.	159
String	**getNamespace**() Returns the name space of this capability.	159
BundleWiring	**getProviderWiring**() Returns the bundle wiring providing this capability.	159
Collection<BundleWiring>	**getRequirerWirings**() Returns the bundle wirings that require this capability.	160

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 cabability'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.

BUNDLE_CAPABILITY

```
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 cabability'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

org.osgi.framework.wiring
All Superinterfaces:
BundleReference

```
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		Page
Collection<Bundle>	**getDependencyClosure**(Collection<Bundle> bundles) Returns the dependency closure for the specified bundles.	163
Collection<Bundle>	**getRemovalPendingBundles**() Returns the bundles that have non-current, in use bundle wirings.	162
void	**refreshBundles**(Collection<Bundle> bundles, FrameworkListener... listeners) Refreshes the specified bundles.	161
boolean	**resolveBundles**(Collection<Bundle> bundles) Resolves the specified bundles.	162

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

Method Detail

refreshBundles

```
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 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<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

```
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 Bundles which have non-current, in use BundleWirings, 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 `Bundle`s were not created by the same framework instance associated with this FrameworkWiring.

Package org.osgi.util.tracker

Tracker Package Version 1.5.

See:
[Description](#)

Interface Summary		Page
BundleTracker Customizer	The `BundleTrackerCustomizer` interface allows a `BundleTracker` to customize the `Bundle`s that are tracked.	*170*
ServiceTracker Customizer	The `ServiceTrackerCustomizer` interface allows a `ServiceTracker` to customize the service objects that are tracked.	*180*

Class Summary		Page
BundleTracker	The `BundleTracker` class simplifies tracking bundles much like the `ServiceTracker` simplifies tracking services.	*165*
ServiceTracker	The `ServiceTracker` class simplifies using services from the Framework's service registry.	*172*

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

org.osgi.util.tracker

```
java.lang.Object
  └ org.osgi.util.tracker.BundleTracker
```

Type Parameters:
`T` - The type of the tracked object.
All Implemented Interfaces:
BundleTrackerCustomizer<T>

```
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 $
ThreadSafe

Field Summary	Page
protected BundleContext	**context** The Bundle Context used by this `BundleTracker`.	166

Constructor Summary	Page
BundleTracker(BundleContext context, int stateMask, BundleTrackerCustomizer<T> customizer) Create a `BundleTracker` for bundles whose state is present in the specified state mask.	166

Method Summary	Page
T	**addingBundle**(Bundle bundle, BundleEvent event) Default implementation of the `BundleTrackerCustomizer.addingBundle` method.	167
void	**close**() Close this `BundleTracker`.	167
Bundle[]	**getBundles**() Return an array of `Bundle`s for all bundles being tracked by this `BundleTracker`.	168
T	**getObject**(Bundle bundle) Returns the customized object for the specified `Bundle` if the specified bundle is being tracked by this `BundleTracker`.	168

Map<Bundle,T>	**getTracked**() Return a Map with the Bundles and customized objects for all bundles being tracked by this BundleTracker.	*169*
int	**getTrackingCount**() Returns the tracking count for this BundleTracker.	*169*
void	**modifiedBundle**(Bundle bundle, BundleEvent event, T object) Default implementation of the BundleTrackerCustomizer.modifiedBundle method.	*167*
void	**open**() Open this BundleTracker and begin tracking bundles.	*166*
void	**remove**(Bundle bundle) Remove a bundle from this BundleTracker.	*168*
void	**removedBundle**(Bundle bundle, BundleEvent event, T object) Default implementation of the BundleTrackerCustomizer.removedBundle method.	*168*
int	**size**() Return the number of bundles being tracked by this BundleTracker.	*169*

Field Detail

context

```
protected final BundleContext context
```

The Bundle Context used by this BundleTracker.

Constructor Detail

BundleTracker

```
public BundleTracker(BundleContext context,
                     int stateMask,
                     BundleTrackerCustomizer<T> customizer)
```

Create a BundleTracker for bundles whose state is present in the specified state mask.

Bundles whose state is present on the specified state mask will be tracked by this BundleTracker.

Parameters:

context - The BundleContext against which the tracking is done.
stateMask - The bit mask of the ORing of the bundle states to be tracked.
customizer - The customizer object to call when bundles are added, modified, or removed in this BundleTracker. If customizer is null, then this BundleTracker will be used as the BundleTrackerCustomizer and this BundleTracker will call the BundleTrackerCustomizer methods on itself.

See Also:

Bundle.getState()

Method Detail

open

```
public void open()
```

Open this BundleTracker and begin tracking bundles.

Bundle which match the state criteria specified when this BundleTracker was created are now tracked by this BundleTracker.

Throws:
`IllegalStateException` - If the `BundleContext` with which this `BundleTracker` was created is no longer valid.
`SecurityException` - If the caller and this class do not have the appropriate `AdminPermission[context bundle,LISTENER]`, and the Java Runtime Environment supports permissions.

---

close

```
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.

---

getObject

```
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

```
public int size()
```

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

Returns:
The number of bundles being tracked.

---

getTrackingCount

```
public int getTrackingCount()
```

Returns the tracking count for this `BundleTracker`. The tracking count is initialized to 0 when this `BundleTracker` is opened. Every time a bundle is added, modified or removed from this `BundleTracker` the tracking count is incremented.

The tracking count can be used to determine if this `BundleTracker` has added, modified or removed a bundle by comparing a tracking count value previously collected with the current tracking count value. If the value has not changed, then no bundle has been added, modified or removed from this `BundleTracker` since the previous tracking count was collected.

Returns:
The tracking count for this `BundleTracker` or -1 if this `BundleTracker` is not open.

---

getTracked

```
public Map<Bundle,T> getTracked()
```

Return a `Map` with the `Bundle`s and customized objects for all bundles being tracked by this `BundleTracker`.

Returns:
A `Map` with the `Bundle`s and customized objects for all services being tracked by this `BundleTracker`. If no bundles are being tracked, then the returned map is empty.

Since:
1.5

Interface BundleTrackerCustomizer

org.osgi.util.tracker

Type Parameters:
`T` - The type of the tracked object.
All Known Implementing Classes:
BundleTracker

`public interface` **`BundleTrackerCustomizer`**

The `BundleTrackerCustomizer` interface allows a `BundleTracker` to customize the `Bundle`s 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 `BundleEvent`s 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		Page
`T`	**addingBundle**(Bundle bundle, BundleEvent event) A bundle is being added to the `BundleTracker`.	170
`void`	**modifiedBundle**(Bundle bundle, BundleEvent event, T object) A bundle tracked by the `BundleTracker` has been modified.	171
`void`	**removedBundle**(Bundle bundle, BundleEvent event, T object) A bundle tracked by the `BundleTracker` has been removed.	171

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

org.osgi.util.tracker

```
java.lang.Object
  └ org.osgi.util.tracker.ServiceTracker
```

Type Parameters:
S - The type of the service being tracked.
T - The type of the tracked object.
All Implemented Interfaces:
ServiceTrackerCustomizer<S,T>

```
public class ServiceTracker
extends Object
implements ServiceTrackerCustomizer<S,T>
```

The ServiceTracker class simplifies using services from the Framework's service registry.

A ServiceTracker object is constructed with search criteria and a ServiceTrackerCustomizer object. A ServiceTracker can use a ServiceTrackerCustomizer to customize the service objects to be tracked. The ServiceTracker can then be opened to begin tracking all services in the Framework's service registry that match the specified search criteria. The ServiceTracker correctly handles all of the details of listening to 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: b375e1b7075486696b42fc55546375f0546b0d58 $
ThreadSafe

Field Summary		Page
protected BundleContext	**context** The Bundle Context used by this ServiceTracker.	173
protected Filter	**filter** The Filter used by this ServiceTracker which specifies the search criteria for the services to track.	173

Constructor Summary	Page
ServiceTracker(BundleContext context, Class<S> clazz, ServiceTrackerCustomizer<S,T> customizer) Create a ServiceTracker on the specified class.	175
ServiceTracker(BundleContext context, String clazz, ServiceTrackerCustomizer<S,T> customizer) Create a ServiceTracker on the specified class name.	174
ServiceTracker(BundleContext context, Filter filter, ServiceTrackerCustomizer<S,T> customizer) Create a ServiceTracker on the specified Filter object.	174
ServiceTracker(BundleContext context, ServiceReference<S> reference, ServiceTrackerCustomizer<S,T> customizer) Create a ServiceTracker on the specified ServiceReference.	174

Method Summary		Page
T	**addingService**(ServiceReference<S> reference) Default implementation of the ServiceTrackerCustomizer.addingService method.	176
void	**close**() Close this ServiceTracker.	175
T	**getService**() Returns a service object for one of the services being tracked by this ServiceTracker.	178
T	**getService**(ServiceReference<S> reference) Returns the service object for the specified ServiceReference if the specified referenced service is being tracked by this ServiceTracker.	178
ServiceReference<S>	**getServiceReference**() Returns a ServiceReference for one of the services being tracked by this ServiceTracker.	177
ServiceReference<S>[]	**getServiceReferences**() Return an array of ServiceReferences for all services being tracked by this ServiceTracker.	177
Object[]	**getServices**() Return an array of service objects for all services being tracked by this ServiceTracker.	178
SortedMap<ServiceReference<S>, T>	**getTracked**() Return a SortedMap of the ServiceReferences and service objects for all services being tracked by this ServiceTracker.	179
int	**getTrackingCount**() Returns the tracking count for this ServiceTracker.	179
void	**modifiedService**(ServiceReference<S> reference, T service) Default implementation of the ServiceTrackerCustomizer.modifiedService method.	176
void	**open**() Open this ServiceTracker and begin tracking services.	175
void	**open**(boolean trackAllServices) Open this ServiceTracker and begin tracking services.	175
void	**remove**(ServiceReference<S> reference) Remove a service from this ServiceTracker.	178
void	**removedService**(ServiceReference<S> reference, T service) Default implementation of the ServiceTrackerCustomizer.removedService method.	176
int	**size**() Return the number of services being tracked by this ServiceTracker.	178
T	**waitForService**(long timeout) Wait for at least one service to be tracked by this ServiceTracker.	177

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.

Since:
1.1

Constructor Detail

ServiceTracker

```
public ServiceTracker(BundleContext context,
                      ServiceReference<S> reference,
                      ServiceTrackerCustomizer<S,T> customizer)
```

Create a `ServiceTracker` on the specified `ServiceReference`.

The service referenced by the specified `ServiceReference` will be tracked by this `ServiceTracker`.

Parameters:
`context` - The `BundleContext` against which the tracking is done.
`reference` - The `ServiceReference` for the service to be tracked.
`customizer` - The customizer object to call when services are added, modified, or removed in this `ServiceTracker`. If customizer is `null`, then this `ServiceTracker` will be used as the `ServiceTrackerCustomizer` and this `ServiceTracker` will call the `ServiceTrackerCustomizer` methods on itself.

ServiceTracker

```
public ServiceTracker(BundleContext context,
                      String clazz,
                      ServiceTrackerCustomizer<S,T> customizer)
```

Create a `ServiceTracker` on the specified class name.

Services registered under the specified class name will be tracked by this `ServiceTracker`.

Parameters:
`context` - The `BundleContext` against which the tracking is done.
`clazz` - The class name of the services to be tracked.
`customizer` - The customizer object to call when services are added, modified, or removed in this `ServiceTracker`. If customizer is `null`, then this `ServiceTracker` will be used as the `ServiceTrackerCustomizer` and this `ServiceTracker` will call the `ServiceTrackerCustomizer` methods on itself.

ServiceTracker

```
public ServiceTracker(BundleContext context,
                      Filter filter,
                      ServiceTrackerCustomizer<S,T> customizer)
```

Create a `ServiceTracker` on the specified `Filter` object.

Services which match the specified `Filter` object will be tracked by this `ServiceTracker`.

Parameters:
`context` - The `BundleContext` against which the tracking is done.
`filter` - The `Filter` to select the services to be tracked.
`customizer` - The customizer object to call when services are added, modified, or removed in this `ServiceTracker`. If customizer is null, then this `ServiceTracker` will be used as the `ServiceTrackerCustomizer` and this `ServiceTracker` will call the `ServiceTrackerCustomizer` methods on itself.

Since:
1.1

ServiceTracker

```
public ServiceTracker(BundleContext context,
                      Class<S> clazz,
                      ServiceTrackerCustomizer<S,T> customizer)
```

Create a `ServiceTracker` on the specified class.

Services registered under the name of the specified class will be tracked by this `ServiceTracker`.

Parameters:
`context` - The `BundleContext` against which the tracking is done.
`clazz` - The class of the services to be tracked.
`customizer` - The customizer object to call when services are added, modified, or removed in this `ServiceTracker`. If customizer is `null`, then this `ServiceTracker` will be used as the `ServiceTrackerCustomizer` and this `ServiceTracker` will call the `ServiceTrackerCustomizer` methods on itself.
Since:
1.5

Method Detail

open

```
public void open()
```

Open this `ServiceTracker` and begin tracking services.

This implementation calls `open(false)`.

Throws:
`IllegalStateException` - If the `BundleContext` with which this `ServiceTracker` was created is no longer valid.
See Also:
`open(boolean)`

---

open

```
public void open(boolean trackAllServices)
```

Open this `ServiceTracker` and begin tracking services.

Services which match the search criteria specified when this `ServiceTracker` was created are now tracked by this `ServiceTracker`.

Parameters:
`trackAllServices` - If `true`, then this `ServiceTracker` will track all matching services regardless of class loader accessibility. If `false`, then this `ServiceTracker` will only track matching services which are class loader accessible to the bundle whose `BundleContext` is used by this `ServiceTracker`.
Throws:
`IllegalStateException` - If the `BundleContext` with which this `ServiceTracker` was created is no longer valid.
Since:
1.3

---

close

```
public void close()
```

Close this ServiceTracker.

This method should be called when this ServiceTracker should end the tracking of services.

This implementation calls getServiceReferences() to get the list of tracked services to remove.

---

addingService

```
public T addingService(ServiceReference<S> reference)
```

Default implementation of the ServiceTrackerCustomizer.addingService method.

This method is only called when this ServiceTracker has been constructed with a null ServiceTrackerCustomizer argument.

This implementation returns the result of calling getService on the BundleContext with which this ServiceTracker was created passing the specified ServiceReference.

This method can be overridden in a subclass to customize the service object to be tracked for the service being added. In that case, take care not to rely on the default implementation of removedService to unget the service.

Specified by:
addingService in interface ServiceTrackerCustomizer
Parameters:
reference - The reference to the service being added to this ServiceTracker.
Returns:
The service object to be tracked for the service added to this ServiceTracker.
See Also:
ServiceTrackerCustomizer.addingService(ServiceReference)

---

modifiedService

```
public void modifiedService(ServiceReference<S> reference,
                            T service)
```

Default implementation of the ServiceTrackerCustomizer.modifiedService method.

This method is only called when this ServiceTracker has been constructed with a null ServiceTrackerCustomizer argument.

This implementation does nothing.

Specified by:
modifiedService in interface ServiceTrackerCustomizer
Parameters:
reference - The reference to modified service.
service - The service object for the modified service.
See Also:
ServiceTrackerCustomizer.modifiedService(ServiceReference, Object)

---

removedService

```
public void removedService(ServiceReference<S> reference,
                           T service)
```

Default implementation of the ServiceTrackerCustomizer.removedService method.

This method is only called when this ServiceTracker has been constructed with a null ServiceTrackerCustomizer argument.

This implementation calls `ungetService`, on the `BundleContext` with which this `ServiceTracker` was created, passing the specified `ServiceReference`.

This method can be overridden in a subclass. If the default implementation of `addingService` method was used, this method must unget the service.

Specified by:
`removedService` in interface `ServiceTrackerCustomizer`
Parameters:
`reference` - The reference to removed service.
`service` - The service object for the removed service.
See Also:
`ServiceTrackerCustomizer.removedService(ServiceReference, Object)`

---

waitForService

```
public T waitForService(long timeout)
                throws InterruptedException
```

Wait for at least one service to be tracked by this `ServiceTracker`. This method will also return when this `ServiceTracker` is closed.

It is strongly recommended that `waitForService` is not used during the calling of the `BundleActivator` methods. `BundleActivator` methods are expected to complete in a short period of time.

This implementation calls `getService()` to determine if a service is being tracked.

Parameters:
`timeout` - The time interval in milliseconds to wait. If zero, the method will wait indefinitely.
Returns:
Returns the result of `getService()`.
Throws:
`InterruptedException` - If another thread has interrupted the current thread.
`IllegalArgumentException` - If the value of timeout is negative.

---

getServiceReferences

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

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

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

---

getServiceReference

```
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.

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 `ServiceReference`s 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 `ServiceReference`s 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

---

public interface **ServiceTrackerCustomizer**

The ServiceTrackerCustomizer interface allows a ServiceTracker to customize the service objects that are tracked. A ServiceTrackerCustomizer is called when a service is being added to a ServiceTracker. The ServiceTrackerCustomizer can then return an object for the tracked service. A ServiceTrackerCustomizer is also called when a tracked service is modified or has been removed from a ServiceTracker.

The methods in this interface may be called as the result of a ServiceEvent being received by a ServiceTracker. Since ServiceEvents are synchronously delivered by the Framework, it is highly recommended that implementations of these methods do not register (BundleContext.registerService), modify (ServiceRegistration.setProperties) or unregister (ServiceRegistration.unregister) a service while being synchronized on any object.

The ServiceTracker class is thread-safe. It does not call a ServiceTrackerCustomizer while holding any locks. ServiceTrackerCustomizer implementations must also be thread-safe.

Version:
$Id: c654a963336cee74762b8f54c8cef8d5774f8b4d $
ThreadSafe

Method Summary		Page
T	**addingService**(ServiceReference<S> reference) A service is being added to the ServiceTracker.	180
void	**modifiedService**(ServiceReference<S> reference, T service) A service tracked by the ServiceTracker has been modified.	181
void	**removedService**(ServiceReference<S> reference, T service) A service tracked by the ServiceTracker has been removed.	181

Method Detail

addingService

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

```
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 it properties modified.

Parameters:
`reference` - The reference to the service that has been modified.
`service` - The service object for the specified referenced service.

---

removedService

```
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 exploting 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 BundleRevsion

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

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

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

[3] Java: Understanding JSR14, http://twit88.com/blog/2008/08/26/java-understanding-jsr14/

[4] Java theory and practice: Using Java 5 language features in earlier JDKs, http://www.ibm.com/developerworks/java/library/j-jtp02277.html

9.2 Author's Address

Name	BJ Hargrave
Company	IBM Corporation
Address	
Voice	
e-mail	hargrave@us.ibm.com

9.3 Acronyms and Abbreviations

9.4 End of Document