RFC 121 Bundle Tracker

Final

18 Pages

Abstract

The BundleTracker class simplifies tracking bundles much like the ServiceTracker simplified tracking services.

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 [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	7 August 2007	Initial Draft.
Draft 2	13 September 2007	Based upon CPEG discussion, I remove support for async BundleListener use by the tracker.
Draft 3	3 October 2007	Based upon CPEG discussion, I modified the customizer signature to pass the BundleEvent, if any, which triggered the action.
Final	2 December 2008	No Changes. Going Final for CPEG voting.

1 Introduction

Service Tracker[4] has long been around (since Release 2) and has long been a very useful tool providing a simple and correct way to track a set of services in the face of dynamism. It is very useful for implementing whiteboard pattern approaches. With the advent of the extender model, it is now important to have a simple and correct way to track a set of bundles in the face of dynamism.

2 Application Domain

Any bundle, such as an extender bundle, that needs to track a set of bundles in a given range of states will need code to enable this tracking. Currently this is custom code for each such bundle.

3 Problem Description

Tracking bundles and services in the OSGi environment is challenging to do simply and correctly. Bundles may change state at any time and the bundle which needs to do the tracking will be started after a set of bundles already are present. The same challenges present for tracking services are also present for tracking bundles. Like the Service Tracker class introduced in Release 2, a Bundle Tracker class is needed[3] to define a standard, correct and easy-to-use way to track bundles.

4 Requirements

The following requirements are met by the proposed solution:

1. The Bundle Tracker class must be modeled along the Service Tracker class to provide a familiar pattern to developers.

2. The Bundle Tracker class must be in the org.osgi.util.tracker package to share code with Service Tracker reducing size, errors and maintenance.

3. The Bundle Tracker class must be correct with respect to the dynamic nature of OSGi.

4. The Bundle Tracker class must track all existing bundles which match the specified criteria as well as bundle whose state change to match the specified criteria after the tracker is opened.

5. The Bundle Tracker must be thread safe.

6. The Bundle Tracker must be able to support early versions of the framework (SynchronousBundleListener support is the minimal requirement.)

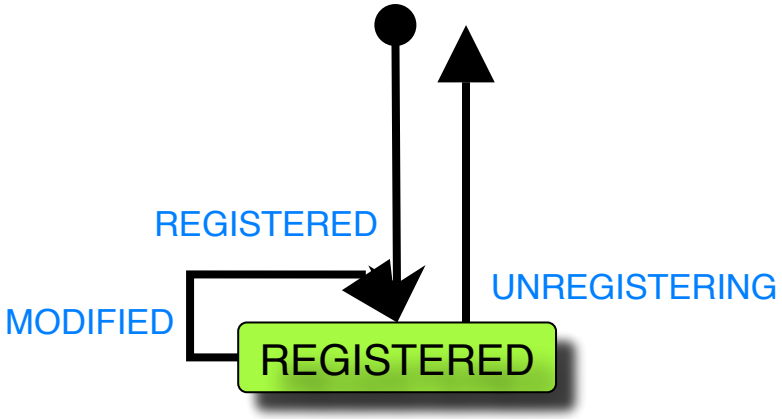
5 Technical Solution

5.1 Design Discussion

5.1.1 Overview

The Service Tracker design is based upon tracking service which are registered and match some specified criteria. Service events are used to indicate a state change in the service and are (only) synchronously delivered. During event processing, the event type along with criteria match against the service metadata is used to decide whether the service is to be added or removed from the tracker, or has just been modified.

The state diagram of a service is very simple.



The state diagram of a bundle is much more complex. The Bundle Tracker design is based upon the bundle's state. Bundles are tracked if they are in a set of states and not tracked otherwise. In this design, the state of the bundle is of primary importance and not the type of the bundle event received by the tracker.

5.1.2 org.osgi.util.tracker package

This design places the new Bundle Tracker into the existing org.osgi.util.tracker package. This is valuable for 2 reasons. First, it enables code sharing with the Service Tracker class. The tracking logic has been refactored from the Service Tracker class into an abstract base class which is then used by the Bundle Tracker. This reduces footprint and (hopefully) errors due to maintenance of duplicated code.

Given the name of the package, which does not include the word "service", there is no package naming issue. However section 701 of the spec will need to be renamed from Service Tracker Specification to simply Tracker Specification.

Furthermore, with this addition to the package, the version of the package is incremented to 1.4.

5.1.3 Tracking Criteria

The tracking criteria for the Bundle Tracker are supplied as a bit mask in the constructor. This mask is an ORing of a set of bundle states. If a bundle is in one of those states, the Bundle Tracker will track it. Since the tracker must track bundles whose state matches the criteria at the time the tracker is opened as well as bundles whose state changes to match the criteria after the tracker is opened, a consistent test is needed.

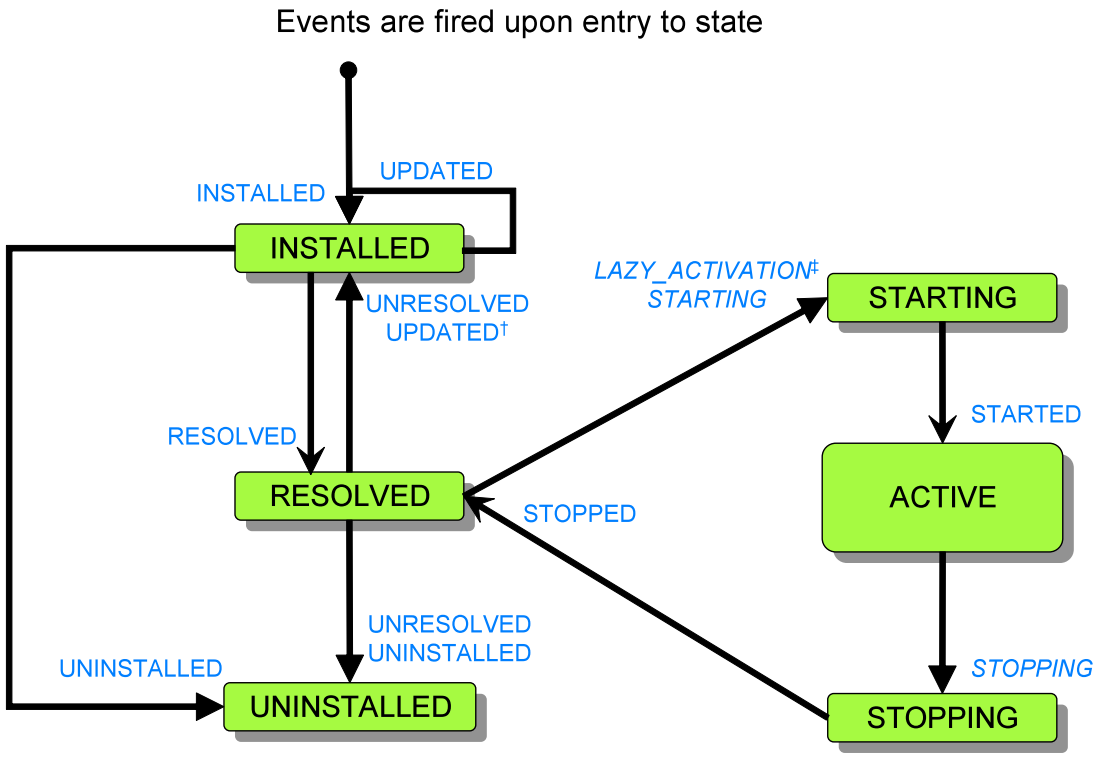
During tracker open, the only state to test is the bundle's state as visible via Bundle.getState. However, while processing bundle events, both the bundle's state and the event type are available for examination. Bundle event types are not sufficient to describe the resulting state of the bundle. The event type UNRESOLVED can be fired for entry to the INSTALLED and UNINSTALLED state.

In order to provide a simple and consistent test, the Bundle Tracker always examines the bundle's state during open and bundle event processing. The bundle event type is not use. The delivery of the bundle event is used as a trigger to test the bundle against the tracking criteria to decide of the bundle should be tracked or untracked.

5.1.4 Synchronous Listener

The Bundle Tracker uses a Synchronous Bundle Listener. With synchronous bundle event processing, the bundle's state is set before the event is synchronously fired and the event is delivered before the state can change again.

The following diagram depicts the bundle states and the event fired upon entry to those states. Note that entry to STARTING and STOPPING states is only signaled to synchronous bundle listeners and would not be reliably observable to asynchronous bundle listeners.



† only if updated

‡ only if lazy activation; STARTING is later fired when activation commences

Events in *italics* are only delivered to synchronous bundle listeners

5.1.5 Customized object

Like Service Tracker, the Bundle Tracker also allows the tracking of a customized object (object returned from BundleTrackerCustomized.addingBundle) along with the tracked bundle. For Service Tracker, this customized object is typically the service object. For Bundle Tracker, the default implementation of addingBundle simply returns the bundle.

5.2 org.osgi.util.tracker.BundleTracker

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

All Implemented Interfaces:

BundleTrackerCustomizer

```
public class BundleTracker
```

```
extends java.lang.Object
implements BundleTrackerCustomizer
```

The `BundleTracker` class simplifies tracking bundles much like the `ServiceTracker` simplifies tracking services.

A `BundleTracker` object is constructed with state criteria and a `BundleTrackerCustomizer` object. A `BundleTracker` object can use the `BundleTrackerCustomizer` object to select which bundles are tracked and to create a customized object to be tracked with the bundle. The `BundleTracker` object 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 `getCustomizedObject` 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` object while holding any locks. `BundleTrackerCustomizer` implementations must also be thread-safe.

Since:

1.4

Field Summary

protected org.osgi.framework.BundleContext	**context** Bundle context this `BundleTracker` object is tracking against.

Constructor Summary

BundleTracker(org.osgi.framework.BundleContext context, int stateMask, BundleTrackerCustomizer customizer)
Create a `BundleTracker` object for bundles whose state is present in the specified state mask.

Method Summary

java.lang.Object	**addingBundle**(org.osgi.framework.Bundle bundle, org.osgi.framework.BundleEvent event) Default implementation of the `BundleTrackerCustomizer.addingBundle` method.
void	**close**() Close this `BundleTracker` object.
org.osgi.framework.Bundle[]	**getBundles**() Return an array of `Bundle` objects for all bundles being tracked by this `BundleTracker` object.
java.lang.Object	**getObject**(org.osgi.framework.Bundle bundle) Returns the customized object for the specified `Bundle` object if the

	bundle is being tracked by this `BundleTracker` object.
`int`	**`getTrackingCount`**`()` Returns the tracking count for this `BundleTracker` object.
`void`	**`modifiedBundle`**`(org.osgi.framework.Bundle bundle, org.osgi.framework.BundleEvent event, java.lang.Object object)` Default implementation of the `BundleTrackerCustomizer.modifiedBundle` method.
`void`	**`open`**`()` Open this `BundleTracker` object and begin tracking bundles.
`void`	**`remove`**`(org.osgi.framework.Bundle bundle)` Remove a bundle from this `BundleTracker` object.
`void`	**`removedBundle`**`(org.osgi.framework.Bundle bundle, org.osgi.framework.BundleEvent event, java.lang.Object object)` Default implementation of the `BundleTrackerCustomizer.removedBundle` method.
`int`	**`size`**`()` Return the number of bundles being tracked by this `BundleTracker` object.

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

Field Detail

5.2.1 context

```
protected final org.osgi.framework.BundleContext context
```

Bundle context this `BundleTracker` object is tracking against.

Constructor Detail

5.2.2 BundleTracker

```
public BundleTracker(org.osgi.framework.BundleContext context,
                     int stateMask,
                     BundleTrackerCustomizer customizer)
```

Create a `BundleTracker` object 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` object.

Parameters:

`context` - `BundleContext` object against which the tracking is done.
`stateMask` - A bit mask of the `OR`ing of the bundle states to be tracked.
`customizer` - The customizer object to call when bundles are added, modified, or removed in this `BundleTracker` object. If customizer is `null`, then this `BundleTracker` object will be used as the `BundleTrackerCustomizer` object and the `BundleTracker` object will call the `BundleTrackerCustomizer` methods on itself.
See Also:
`Bundle.getState()`

Method Detail

5.2.3 addingBundle

```
public java.lang.Object addingBundle(org.osgi.framework.Bundle bundle,
                                     org.osgi.framework.BundleEvent event)
```

Default implementation of the `BundleTrackerCustomizer.addingBundle` method.

This method is only called when this `BundleTracker` object has been constructed with a `null` `BundleTrackerCustomizer` argument. The default implementation returns the specified `Bundle` object.

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` - 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 customized object to be tracked for the bundle added to this `BundleTracker` object.
See Also:
`BundleTrackerCustomizer`

5.2.4 close

```
public void close()
```

Close this `BundleTracker` object.

This method should be called when this `BundleTracker` object should end the tracking of bundles.

5.2.5 getBundles

```
public org.osgi.framework.Bundle[] getBundles()
```

Return an array of `Bundle` objects for all bundles being tracked by this `BundleTracker` object.
Returns:
Array of `Bundle` objects or `null` if no bundles are being tracked.

5.2.6 getObject

`public java.lang.Object` **`getObject`**`(org.osgi.framework.Bundle bundle)`

Returns the customized object for the specified `Bundle` object if the bundle is being tracked by this `BundleTracker` object.

Parameters:
`bundle` - Bundle being tracked.
Returns:
Customized object or `null` if the specified `Bundle` object is not being tracked.

5.2.7 getTrackingCount

`public int` **`getTrackingCount`**`()`

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

The tracking count can be used to determine if this `BundleTracker` object 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` object since the previous tracking count was collected.

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

5.2.8 modifiedBundle

```
public void modifiedBundle(org.osgi.framework.Bundle bundle,
                           org.osgi.framework.BundleEvent event,
                           java.lang.Object object)
```

Default implementation of the `BundleTrackerCustomizer.modifiedBundle` method.

This method is only called when this `BundleTracker` object has been constructed with a `null` `BundleTrackerCustomizer` argument. The default implementation does nothing.

Specified by:
`modifiedBundle` in interface `BundleTrackerCustomizer`
Parameters:
`bundle` - 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 bundle.
See Also:
`BundleTrackerCustomizer`

5.2.9 open

`public void` **`open`**`()`

Open this `BundleTracker` object and begin tracking bundles.

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

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

5.2.10 remove

```
public void remove(org.osgi.framework.Bundle bundle)
```

Remove a bundle from this `BundleTracker` object. The specified bundle will be removed from this `BundleTracker` object. If the specified bundle was being tracked then the `BundleTrackerCustomizer.removedBundle` method will be called for that bundle.

Parameters:
`bundle` - Bundle to be removed.

5.2.11 removedBundle

```
public void removedBundle(org.osgi.framework.Bundle bundle,
                          org.osgi.framework.BundleEvent event,
                          java.lang.Object object)
```

Default implementation of the `BundleTrackerCustomizer.removedBundle` method.

This method is only called when this `BundleTracker` object has been constructed with a `null` `BundleTrackerCustomizer` argument. The default implementation does nothing.

Specified by:
`removedBundle` in interface `BundleTrackerCustomizer`
Parameters:
`bundle` - 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 bundle.
See Also:
`BundleTrackerCustomizer`

5.2.12 size

```
public int size()
```

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

Returns:
Number of bundles being tracked.

5.3 org.osgi.util.tracker.BundleTrackerCustomizer

All Known Implementing Classes:

BundleTracker

```
public interface BundleTrackerCustomizer
```

The `BundleTrackerCustomizer` interface allows a `BundleTracker` object to customize the bundle objects that are tracked. The `BundleTrackerCustomizer` object is called when a bundle is being added to the `BundleTracker` object. The `BundleTrackerCustomizer` can then return an object for the tracked bundle. The `BundleTrackerCustomizer` object is also called when a tracked bundle has been removed from the `BundleTracker` object.

The methods in this interface may be called as the result of a `BundleEvent` being received by a `BundleTracker` object. 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` object while holding any locks. `BundleTrackerCustomizer` implementations must also be thread-safe.

Since:

1.4

Method Summary

java.lang.Object	**addingBundle**(org.osgi.framework.Bundle bundle, org.osgi.framework.BundleEvent event) A bundle is being added to the `BundleTracker` object.
void	**modifiedBundle**(org.osgi.framework.Bundle bundle, org.osgi.framework.BundleEvent event, java.lang.Object object) A bundle tracked by the `BundleTracker` object has been modified.
void	**removedBundle**(org.osgi.framework.Bundle bundle, org.osgi.framework.BundleEvent event, java.lang.Object object) A bundle tracked by the `BundleTracker` object has been removed.

Method Detail

5.3.1 addingBundle

```
java.lang.Object addingBundle(org.osgi.framework.Bundle bundle,
                              org.osgi.framework.BundleEvent event)
```

A bundle is being added to the `BundleTracker` object.

This method is called before a bundle which matched the search parameters of the `BundleTracker` object is added to it. This method should return the object to be tracked for this

`Bundle` object. The returned object is stored in the `BundleTracker` object and is available from the `getBundles` method.

Parameters:
`bundle` - Bundle being added to the `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 object to be tracked for the `Bundle` object or `null` if the `Bundle` object should not be tracked.

5.3.2 modifiedBundle

```
void modifiedBundle(org.osgi.framework.Bundle bundle,
                    org.osgi.framework.BundleEvent event,
                    java.lang.Object object)
```

A bundle tracked by the `BundleTracker` object has been modified.

This method is called when a bundle being tracked by the `BundleTracker` object has had its state modified.

Parameters:
`bundle` - 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 modified bundle.

5.3.3 removedBundle

```
void removedBundle(org.osgi.framework.Bundle bundle,
                   org.osgi.framework.BundleEvent event,
                   java.lang.Object object)
```

A bundle tracked by the `BundleTracker` object has been removed.

This method is called after a bundle is no longer being tracked by the `BundleTracker` object.

Parameters:
`bundle` - 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 removed bundle.

---

6 Considered Alternatives

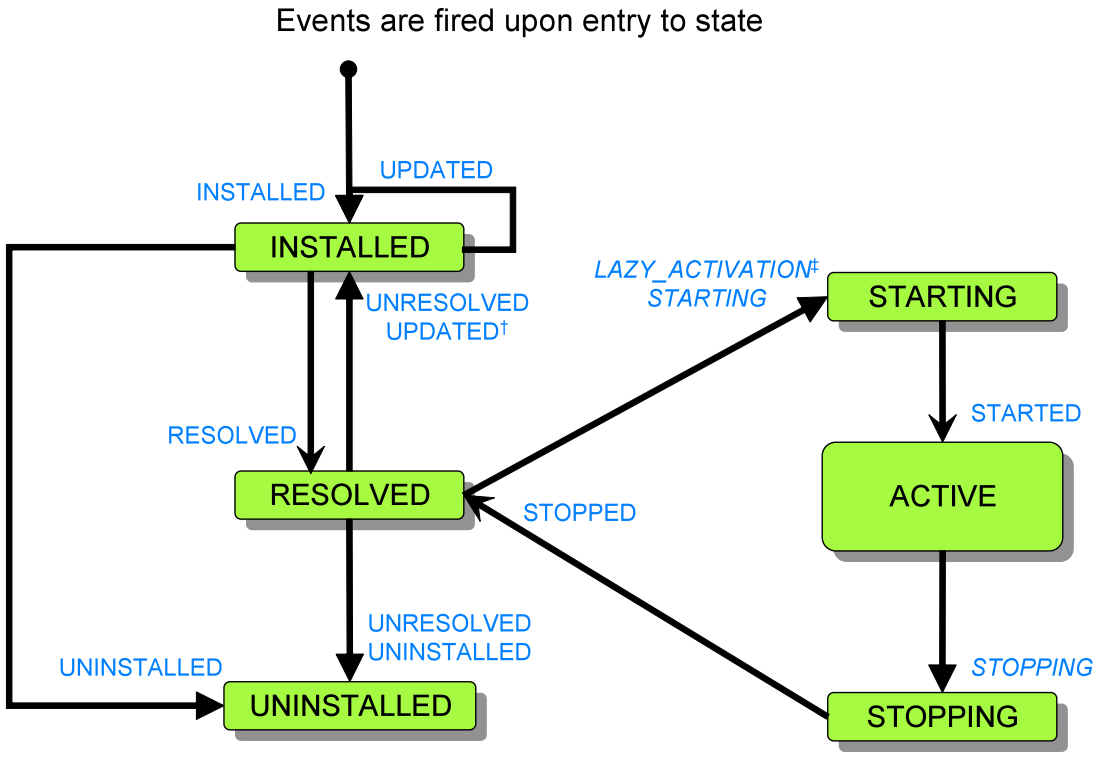
6.1 Using Services to model Bundles

See Member Bug 501[3] for some discussion of this design alternative.

6.2 Using asynchronous Bundle Listener

The first draft of this RFC suggested allowing BundleTracker to provide the option of using either BundleListener (asynchronous) and SynchronousBundleListener. This was changed to only support SynchronousBundleListsner due to issues with using asynchronous BundleListener. Following is the removed text:

> The Bundle Tracker supports user configuration to use either a Synchronous Bundle Listener or the asynchronous Bundle Listener. For synchronous bundle event processing, the bundle's state is set before the event is synchronously fired and the event is delivered before the state can change again. However, if asynchronous bundle event processing is used, then the behavior of the tracker will be different since entry to some states are not visible to asynchronous bundle listeners and the time between the event firing and event delivery may prevent some bundle state transitions from being observed.
>
> The following diagram depicts the bundle states and the event fired upon entry to those states. Note that entry to STARTING and STOPPING states is only signaled to synchronous bundle listeners and are thus not reliably observable to asynchronous bundle listeners.



† only if updated

‡ only if lazy activation; STARTING is later fired when activation commences

Events in *italics* are only delivered to synchronous bundle listeners

There are also cases when several bundle events can be fired before the first event is asynchronously delivered. During processing of the first event by the BundleTracker, the state of the bundle at the time the final event was fired is observed. The can result in the bundle becoming tracked while processing the first event. As the remaining events are then delivered, the bundle's state does not actually change, but the Bundle Tracker will call the modifiedBundle method of the customizer for each additional event. There are also cases where the bundle should be removed and added back the tracker (e.g. bundle update), but delay in delivery of the STOPPED event, which might result in the bundle being removed from the tracker, until after the bundle has been restarted will result in the tracker never seeing the bundle leave the ACTIVE state and thus never being removed and then added back to the tracker.

Thus the value of supporting asynchronous bundle listeners in Bundle Tracker is dubious and we may want to consider removing it.

To deal with the above issues, two approaches are possible but still may deliver suboptimal results.

1. Process the event types in addition to the bundle state to "simulate" the bundle being removed and added if necessary. This logic could be fairly complex as it will have to map event type onto the state map. This would only be necessary for the asynchronous listener.

2. Have Bundle Tracker always use a synchronous bundle listener and wrap the addBundleListener call in a doPrivileged method to not require the caller to have the necessary permission. This would make every bundle able to synchronously be notified of bundle events which will provide a form of privilege elevation in secured systems.

7 Security Considerations

Bundle Tracker runs in the security context of the bundle using it. It doesn't provide or remove any of the security checks that are already in place for bundles.

In order to support tracking bundles synchronously, a SynchronousBundleListener must be used. In order to prevent elevation of privilege, the Bundle Tracker implementation must not use doPrivilege when registering the SynchronousBundleListener object. This means that the code calling the open method (which makes the addBundleListener call) and the Bundle Tracker class itself must both have the AdminPermission[context bundle, LISTENER] permission. In particular, the bundle containing the org.osgi.util.tracker package must have this permission. If the org.osgi.util.tracker package is delivered as part of the framework implementation, then it likely has AllPermission and this requirement is then met.

8 Document Support

8.1 References

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

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

[3]. Member Bug 501, https://www2.osgi.org/members/bugzilla/show_bug.cgi?id=501

[4]. Service Tracker Specification, OSGi Core Specification, R4 V4.1, Section 701

8.2 Author's Address

Name	BJ Hargrave
Company	IBM Corporation
Address	800 N Magnolia Av, Orlando, FL, USA
Voice	+1 386 848 1781
e-mail	hargrave@us.ibm.com

8.3 Acronyms and Abbreviations

8.4 End of Document