RFC 126 Service Hooks

Final

15 Pages

Abstract

This RFC describes the means for a bundle to hook into the service registry operations.

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	7 January 2008	Initial Draft of RFC BJ Hargrave, IBM
2nd draft	18 May 2008	Second draft based upon CPEG feedback. Mostly small changes. Some of the methods where changed to use Collections instead of array since it is expected that data is mutated as control passes along the chain. Security documentation was added to indicate that hook require permission to manipulate the services of a bundle. BJ Hargrave, IBM

Revision	Date	Comments
3rd draft	10 June 2008	Third draft based upon CPEG feedback. Removed the overloaded addServiceListener method and added a new ServiceEvent type. Without the overloaded addServiceListener method, the ListenerHook class has been simplified. Added Exposure to Hook Generated Objects. Additional input from the EGs is needed here. BJ Hargrave, IBM
4th draft	10 July 2008	Fourth draft based upon decisions at the CPEG f2f meeting. CPEG agreed, with approval of the EEG RFC 119 team, to scale this design back. The changes now do not allow the hooks to inject objects into the service registry which avoids the issue of creating new dependencies between the bundles creating and using services and the hook bundles. BJ Hargrave, IBM
5th draft	20 Oct 2008	Updated based upon implementation experience. The PubishHook and ListenerHook designs were changed. BJ Hargrave, IBM
Final	2 December 2008	No changed. Final for CPEG voting. BJHargrave, IBM

1 Introduction

This RFC details how a bundle can hook into service layer operations and influence the operation or observe the operation.

2 Application Domain

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

3 Problem Description

The service layer operations provide no means for a bundle (external to the framework) to observe or manipulate the operations as they occur. Certain specialized bundles need to be able to alter output results of service layer find and event delivery operations to affect their purpose. Such purposes may include things like distributed service model, etc.

4 Requirements

The solution must work with the current service layer model and allow certain bundles to observe the find and listen operations and to potentially reduce the result to affect their desired goals.

The solution must be secured when java permissions are in effect.

5 Technical Solution

5.1 Modifications to current API

In addition to the API in following section, some changes to current API are needed

5.1.1 New ServiceEvent type

A new type, `MODIFIED_ENDMATCH`, is added to ServiceEvent to allow a listener to detect when a service property modification results in the service no longer matching the filter with which the ServiceListener was added. Existing ServiceListener implementation should properly ignore the new event type since the ServiceEvent class has long been documented that new types may be added. (We have added new types to the similarly designed BundleEvent and FrameworkEvent classes in prior releases.) New ServiceListener implementations (such as an updated ServiceTracker) can use the new ServiceEvent type to detect when a service property modification results in an end to the filter match. The new ServiceEvent type 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.

Here is some pseudo code which demonstrates how the framework should process delivery of this new event to a listener in response to the modification of service properties.

```
If (listenerFilter == null)
      /* if no filter, deliver MODIFIED event */
      deliverEvent(listener, ServiceEvent.MODIFIED);
else if (listenerFilter.match(newProperties))
      /* if filter matches new properties, deliver MODIFIED event */
      deliverEvent(listener, ServiceEvent.MODIFIED);
else if (listenerFilter.match(oldProperties))
      /* if filter does not match new properties but does
         match old properties, deliver MODIFIED_ENDMATCH event */
      deliverEvent(listener, ServiceEvent.MODIFIED_ENDMATCH);
```

ServiceTracker must be changed to use the new ServiceEvent type and always register a listener with a user supplied filter string. ServiceTracker can use MODIFIED_ENDMATCH to untrack a service and avoid having to evaluate filters in the ServiceListener implementation.

5.2 New Hook Classes

The javadoc in the next section describes the proposed new hook classes. In order to reduce the number of doPrivileged calls by bundle programmers, the Framework must perform a doPrivileged call around any calls to the framework hooks. See Privileged Callbacks (Core Specification section 4.8.3).

The following hook types are defined.

5.2.1 PublishHook

Bundles registering this service will be called during framework service publish (register, modify, and unregister service) operations. This method is called prior to service event delivery when a publishing bundle registers, modifies or unregisters a service and can filter the bundles which receive the event.

5.2.2 FindHook

Bundles registering this service will be called during framework service find (get service references) operations. This method is called during the service find operation by the finding bundle and can filter the result of the find operation.

5.2.3 ListenerHook

Bundles registering this service will be called during service listener addition and removal. The hook is notified of the collection of service listeners and what they may be listening for and well as future changes to that collection.

5.3 Backwards Compatibility Requirements

Hooks are very specialized services which are tied closely to the operations of the service layer of the framework. While every attempt to maintain the backwards compatibility of the hook api will be made, it is possible that changes or additions to the service layer API in future versions of the OSGi Core Specification make require changes to the hook API which breaks backwards compatibility.

6 Javadoc

This section contains the javadoc for the new hook classes.

6.1 org.osgi.framework.hooks.service
Interface FindHook

```
public interface FindHook
```

OSGi Framework Service Find Hook Service.

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

ThreadSafe

Method Summary

void	**find**(BundleContext context, java.lang.String name, java.lang.String filter, boolean allServices, java.util.Collection references) Find hook method.

Method Detail

6.1.1 find

```
void find(BundleContextcontext,
          java.lang.Stringname,
          java.lang.Stringfilter,
          booleanallServices,
          java.util.Collectionreferences)
```

Find hook method. This method is called during the service find operation (for example, BundleContext.getServiceReferences(String, String)). This method can filter the result of the find operation.

Parameters:

context - The bundle context of the bundle performing the find operation.
name - The class name of the services to find or null to find all services.
filter - The filter criteria of the services to find or null for no filter criteria.
allServices - true if the find operation is the result of a call to BundleContext.getAllServiceReferences(String, String)
references - A Collection of Service References to be returned as a result of the find operation. The method implementation may remove service references from the collection to prevent the references from being returned to the bundle performing the find operation. The collection supports all the optional Collection operations except add and addAll. Attempting to add to the collection will result in an UnsupportedOperationException. The collection is not synchronized.

6.2 org.osgi.framework.hooks.service
Interface ListenerHook

```
public interface ListenerHook
```

OSGi Framework Service Listener Hook Service.

Bundles registering this service will be called during service listener addition and removal.

ThreadSafe

Nested Class Summary

static interface	**ListenerHook.ListenerInfo** Information about a Service Listener.

Method Summary

void	**added**(java.util.Collection listeners)

	Added listeners hook method.
void	**removed**(java.util.Collection listeners) Removed listeners hook method.

Method Detail

6.2.1 added

```
void added(java.util.Collectionlisteners)
```

Added listeners hook method. This method is called to provide the hook implementation with information on newly added service listeners. This method will be called as service listeners are added while this hook is registered. Also, immediately after registration of this hook, this method will be called to provide the current collection of service listeners which had been added prior to the hook being registered.

Parameters:
listeners - A collection of ListenerHook.ListenerInfos for newly added service listeners which are now listening to service events. Attempting to add to or remove from the collection will result in an UnsupportedOperationException. The collection is not synchronized.

6.2.2 removed

```
void removed(java.util.Collectionlisteners)
```

Removed listeners hook method. This method is called to provide the hook implementation with information on newly removed service listeners. This method will be called as service listeners are removed while this hook is registered.

Parameters:
listeners - A collection of ListenerHook.ListenerInfos for newly removed service listeners which are no longer listening to service events. Attempting to add to or remove from the collection will result in an UnsupportedOperationException. The collection is not synchronized.

6.3 org.osgi.framework.hooks.service
Interface ListenerHook.ListenerInfo

Enclosing interface:
ListenerHook

```
public static interface ListenerHook.ListenerInfo
```

Information about a Service Listener. This interface describes the bundle which added the Service Listener and the filter with which it was added.

ThreadSafe

Method Summary	
BundleContext	**getBundleContext**() Return the context of the bundle which added the listener.
java.lang.String	**getFilter**() Return the filter string with which the listener was added.

Method Detail

6.3.1 getBundleContext

BundleContext **getBundleContext**()

Return the context of the bundle which added the listener.

Returns:
The context of the bundle which added the listener.

6.3.2 getFilter

java.lang.String **getFilter**()

Return the filter string with which the listener was added.

Returns:
The filter string with which the listener was added. This may be null if the listener was added without a filter.

6.4 org.osgi.framework.hooks.service
Interface PublishHook

public interface **PublishHook**

OSGi Framework Service Publish Hook Service.

Bundles registering this service will be called during framework service publish (register, modify, and unregister service) operations.

ThreadSafe

Method Summary	
void	**event**(ServiceEvent event, java.util.Collection contexts) Event hook method.

Method Detail

6.4.1 event

```
void event(ServiceEventevent,
           java.util.Collectioncontexts)
```

Event hook method. This method is called prior to service event delivery when a publishing bundle registers, modifies or unregisters a service. This method can filter the bundles which receive the event.

Parameters:

event - The service event to be delivered.
contexts - A Collection of Bundle Contexts for bundles which have listeners to which the specified event will be delivered. The method implementation may remove bundle contexts from the collection to prevent the event from being delivered to the associated bundles. The collection supports all the optional Collection operations except add and addAll. Attempting to add to the collection will result in an UnsupportedOperationException. The collection is not synchronized.

---

7 Considered Alternatives

7.1 Pre and post hooks

The original proposal was based upon pre and post hook methods being called before and after each service operation. The issue with that proposal was matching the pre and post hook calls to a specific service operation (since service operations can be nested). The current proposal is a stack-based mechanism as each hook is responsible for calling to the next step of the operation and in general provides a simpler model.

7.2 Listening to specific service names

Adding a new addServiceListener method was rejected in favor of adding a new ServiceEvent type to notify ServiceListeners added with a filter string that a property modification occurred which causes the service to no longer match the filter.

The BundleContext.addServiceListener method must overloaded with a new method which takes an array of service names:

```
addServiceListener(ServiceListener listener, String[] names)
```

This new method is an alternate to the filter variant of addServiceListener which takes a specific list of service names to watch. The strings in the names parameter allow the use of the "*" wildcard like filters. Using this new method is the equivalent of calling addServiceListener with the filter string:

```
"(|(objectClass="+name[0]")" ... "(objectClass="+name[n]+")"
```

This new any new function, but provide a means of registering interest in specific service names which can allow the framework to make certain optimizations, rather than trying to look inside a potentially complicated filter string.

7.3 Framework Proxying of Hook Generated Objects

An alternative to Exposure to Hook Generated Objects was rejected. This alternative would require the framework to proxy wrap all hook generated objects such that no other bundle (including other hooks) would be exposed to hook generated objects. Then, when a hook service is unregistered, for all object generated by the hook, the framework generated proxy wrapper would dereference the hook generated objects and route around the dereferenced object.

While this is simpler and less perturbing than the requirements in Exposure to Hook Generated Objects, this alternative raised many correctness issues. For example, a hook may have modified the properties of a service during registration. Now that the hook is removed, the service property on the currently registered service should be removed. Doing this is harder for the framework to detect and "unwind".

Due to these correctness issues, this alternative was rejected.

7.4 Full Manipulation Capabilities

The goals of this RFC have been scaled back to avoid the issues with exposure of hook generated objects to bundle and the associated dependencies which are created and their cleanup.

7.4.1.1 Chained Hook Classes

Service hooks are not called for service operations on other service hooks. Each hook type (except for ListenerHook which is not chained) is chained and called in a sequence during the processing of a service operation. Each hook in the chain must call the chain object to continue processing the chain. The "pre" phase of a hook are the operations that occur prior to calling the chain object to pass control to the next hook in the chain. This typically includes modifying parameters before calling the chain object. The "post" phase of the hook are the operations that occur after the chain object returns. This may include modifying the return value of the chain object.

In general, the hooks are very powerful mechanism that can be used to perform many interesting functions. But because the hooks are powerful, care must be avoid misusing or abusing them.

The following hook types are defined.

7.4.1.2 PublishHook

Bundles registering this service will be called during framework service publish (register service) operations. This hook allows one to influence the service publish (registration) operation.

Since a PublishHook implementation may create and return a ServiceRegistration object (which wraps the original, framework created ServiceRegistration object), care must be taken by the framework during automatic

service unregistration at bundle stop. Since the hook created ServiceRegistration object may manage resources that should be freed when the service is unregistered, during automatic service unregistration at bundle stop, the framework MUST call the unregister method on the ServiceRegistration object returned to the bundle which registered the service to allow any hook defined processing to occur.

7.4.1.3 FindHook

Bundles registering this service will be called during framework service find (get service references) operations. This hooks allows one to influence the service find operation. This hook supports things such as just-in-time service publication by examining the find parameters during the pre phase processing of the hook.

7.4.1.4 BindHook

Bundles registering this service will be called during framework service bind (get service object) operations. This hook allows one to influence the service bind operation. This hook supports thing such as proxy wrapping of the service object during the post phase processing of the hook.

7.4.1.5 EventHook

Bundles registering this service will be called during framework service event delivery. This hook allows one to influence the set of bundles which receive a service event.

7.4.2 Exposure to Hook Generated Objects

During the course of service operations, a bundle may become exposed to hook generated objects. For example, a hook could create and return a ServiceRegistration wrapper or create and return a service object wrapper to allow the hook to insert behavior onto the object's operations.

But hooks are services registered from ordinary bundles. Those bundles can be stopped and their hook services unregistered or the hook bundle may decide for some reason to unregister the hook service. This can leave the hook generated objects dangling in the system. In order to properly manage these dependencies from the client bundle to the hook service, the framework must track these dependencies and properly resolve them.

When a hook service returns a hook generated object, if the service operation completes normally resulting in the client bundle becoming dependent on the hook generated object, the framework must mark the client bundle dependent upon the hook service. If the client bundle is stopped, then all dependencies on the hook service are removed. However, if the hook service is unregistered, the client bundles that are dependent on that hook service must all be stopped and restarted so that they will re-execute the service operations without the presence of the removed hook service.

Given potential perturbation issues with multiple bundles being restarted multiple times of a set of hook bundles having a set of hooks service are updated, some care must be taken to avoid this. However, there are several scenarios to consider:

●Simple unregistration of hook service – The hook bundle may decide to unregister the hook service for some reason. The unregistration is not as a result of the hook bundle stopping and it is unknown when or if the hook service will be reregistered. In this case, there are no other event to wait for to decide to handle the client bundle dependencies on the hook service.

●Hook bundle is stopped – The hook bundle is stopped for some reason (that is not part of an update operation). As a result, all hook services from the hook bundle are unregistered. A client bundle may have dependencies on multiple hook services from the hook bundle. We don't want to restart the client bundle multiple times (once for each unregistered hook service that the client bundle is dependent upon). Also, it is unknown when or if the hook

bundle will be restarted and its hook service reregistered. In this case, there are no other event to wait for to decide to handle the client bundle dependencies on the hook services.

●One or more hook bundles are updated – A set of hook bundles are updated (hopefully all stopped, all updated and all restarted). It is also possible a refreshPackages call may be made on these hook bundles prior to restarting them. If we know the hook services are being unregistered as part of a larger update operation, we will want to coordinate restarting the dependent client bundles. In fact, we would want to stop the dependent client bundles before we stop the hook bundles and restart them after we restart the hook bundles. But this would require the framework to know things that is does not know. It has been suggested to delay stopping and restarting the dependent client bundles until refreshPackages is called. This does leave the dependent client bundles operating while the hook bundles have been stopped and updated.

From a correctness point of view, restarting the dependent client bundles must be done when the hook service is unregistered. However, there is the potential for multiple restarts. Additional input from CPEG and EEG is requested here.

7.4.3 AdminPermission

The proposed new action to be added to AdminPermission is rejected. Since the capability of the hook is reduced, such fine grained control is not necessary.

`AdminPermission.SERVICE_HOOK`

This new type allows control over whether the hook can manipulate the service operations of the target bundle.

8 Security Considerations

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

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

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

9.2 Author's Address

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

9.3 Acronyms and Abbreviations

9.4 End of Document