RFC 203 – Remote Service Admin 1.1

Draft

15 Pages

Abstract

The Remote Service Admin specification is lacking a mechanism to notify consumers of changes to an endpoint. The EndpointListener interface defines endpointAdded and endpointRemoved callbacks, but no mechanism to convey that an endpoint has been modified, for example because the service properties of the backing service have changed. This RFC addresses this issue.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

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

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

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

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

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

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable,

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

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

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 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.
```

0.6 Revision History

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

Revision	Date	Comments
Initial	August, 2013	David Bosschaert, initial version of separate RFC. Previous design has been taking place in RFC 183.
0.1	September, 2013	Post September F2F discussion - add the new "update" methods to the ExportRegistration and ImportRegistration
0.2	September, 2013	Clarify the uniqueness requirements for endpoint ids, include community feedback

1 Introduction

The OSGi Remote Services and Remote Service Admin specifications describe how OSGi services can be remoted and how to consume these remote services using the OSGi Services programming model.

The Remote Service Admin specification version 1.0 defines how listeners are notified of endpoints being added and removed. However the associated API does not support notifying listeners of changes to endpoints such as service property changes of the associated service. This RFC addresses this issue by proposing an extension to the Remote Service Admin specification.

2 Application Domain

This RFC relates to the domain of remote OSGi Services, specifically the Remote Service Admin specification.

3 Problem Description

The EndpointListener interface is used to implement a distributed discovery mechanism and it allows the registration of a listener for distributed endpoints to appear and disappear via the endpointAdded() and endpointRemoved() callback methods. However, an endpoint can also change. This is in particular the case when the service registration properties of the endpoint are modified. Such modifications are not supported by the EndpointListener today, it sends a sequence of endpointRemoved() and endpointAdded() callbacks in such a case which can cause unnecessary volatility in the system.

4 Requirements

RSA01 – The Solution MUST define a mechanism to provide Endpoint Listeners with a notification when an endpoint was modified.

RSA02 – The Solution SHOULD allow a Topology Manager to update the service properties an Exported or Imported Service without unregistering it. This may not be possible if the configuration type or access intent of the service changes.

5 Technical Solution

To receive modification events a new EndpointEventListener interface can be implemented by the listener. The EndpointEventListener follows a similar pattern as the ServiceListener in the core framework. The event holds a type attribute describing the type of event.

5.1 Backward compatibility

The existing EndpointListener interface sends a endpointRemoved() callback followed by an endpointAdded() callback in case an endpoint registration has changed (e.g. properties added or removed). The EndpointListener interface will continue to behave this way.

5.2 EndpointEventListener

The new EndpointEventListener will not send a sequence of REMOVED and ADDED events in such a case, but rather send a single MODIFIED or MODIFIED_ENDMATCH event, whichever is appropriate.

The EndpointEventListener is defined as follows:

```
public interface EndpointEventListener {
  void endpointChanged(EndpointEvent event, String matchedFilter);
}

public class EndpointEvent {
  public static final int ADDED = 0x00000001;
  public static final int REMOVED = 0x00000002;
  public static final int MODIFIED = 0x00000004;
  public static final int MODIFIED_ENDMATCH = 0x00000008;

  private final EndpointDescription endpoint;
  private final int type;

  public EndpointEvent(int type, EndpointDescription endpoint) {
    super(endpoint);
    this.endpoint = endpoint;
    this.type = type;
  }

  public EndpointDescription getEndpoint() {
    return endpoint;
  }

  public int getType() {
    return type;
  }
}
```

5.3 Updating Exported and Imported Services

In the Remote Service Admin specification the Topology manager is the component responsible for tracking and managing the services that should be exported from, or imported into, the service registry. Furthermore the Topology Manager is responsible for notifying RSA Discovery of endpoint changes (additions, deletions and modifications), and RSA Discovery is responsible for notifying Topology Managers of changes to the status of remote EndpointDescriptions.

In addition to its interactions with RSA Discovery, the Topology Manager is also responsible for interacting with RSA distribution providers. In RSA 1.0 this was broadly limited to creating and closing ImportRegistration and ExportRegistration objects using the Distribution Provider. To support Requirement RSA02 RSA 1.1 will need additional interaction mechanisms to indicate that an ImportRegistration or ExportRegistration should be updated. It is necessary for the Topology Manager to initiate these updates, because either:

- It is then responsible for notifying Discovery services of any changes to an exported endpoint

or

- The topology manager is the component that is notified of changes to a remote endpoint via Discovery announcements.

To support these cases it is necessary to add update methods to both ImportRegistration and ExportRegistration. As these are “provider types” that should only be implemented by RSA Distribution providers this represents a minor change to the RSA API.

5.3.1 ExportRegistration

```
/*
 * Copyright (c) OSGi Alliance (2009, 2013). All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.osgi.service.remoteserviceadmin;

import java.util.Map;
import org.osgi.framework.ServiceReference;

/**
 * An Export Registration associates a service to a local endpoint.
```

```
 *
 * The Export Registration can be used to delete the endpoint associated with an
 * this registration. It is created with the
 * {@link RemoteServiceAdmin#exportService(ServiceReference,Map)} method.
 *
 * When this Export Registration has been closed, all methods must return
 * {@code null}.
 *
 * @ThreadSafe
 * @noimplement
 * @author $Id: 5491675ba2ea4b7769f70040e8f48a22777028a3 $
 */
public interface ExportRegistration {
	/**
	 * Return the Export Reference for the exported service.
	 *
	 * @return The Export Reference for this registration.
	 * @throws IllegalStateException When this registration was not properly
	 *         initialized. See {@link #getException()}.
	 */
	ExportReference getExportReference();

	/**
	 * Update the endpoint represented by this {@link ExportRegistration} and
	 * return an updated {@link EndpointDescription}. If this method returns an
	 * updated {@link EndpointDescription}, then the object returned via
	 * {@link #getExportReference()} must also have been updated to return this
	 * new object. If this method does not return an updated
	 * {@link EndpointDescription} then the object returned via
	 * {@link #getExportReference()} should remain unchanged.
	 *
	 * When creating the updated {@link EndpointDescription} the
	 * {@link ServiceReference} originally passed to
	 * {@link RemoteServiceAdmin#exportService(ServiceReference, Map)} must be
	 * queried to pick up any changes to its service properties.
	 *
	 * If this argument is null then the original properties passed when
	 * creating this ExportRegistration should be used when constructing the
	 * updated {@link EndpointDescription}. Otherwise the new properties should
	 * be used, and replace the original properties for subsequent calls to the
	 * update method.
	 *
	 * @param properties properties to be merged with the current service
	 *        properties for the {@link ServiceReference} represented by this
	 *        {@link ExportRegistration}. If null is passed then the original
	 *        properties passed to
	 *        {@link RemoteServiceAdmin#exportService(ServiceReference, Map)}
	 *        will be used.
	 * @return The updated {@link EndpointDescription} for this registration.
	 * @throws IllegalStateException When this registration was not properly
	 *         initialized. See {@link #getException()}.
	 *
```

```
        */
        EndpointDescription update(Map<String, ?> properties);

        /**
         * Delete the local endpoint and disconnect any remote distribution
         * providers. After this method returns, all methods must return
         * {@code null}.
         *
         * This method has no effect when this registration has already been closed
         * or is being closed.
         */
        void close();

        /**
         * Return the exception for any error during the export process.
         *
         * If the Remote Service Admin for some reasons is unable to properly
         * initialize this registration, then it must return an exception from this
         * method. If no error occurred, this method must return {@code null}.
         *
         * The error must be set before this Export Registration is returned.
         * Asynchronously occurring errors must be reported to the log.
         *
         * @return The exception that occurred during the initialization of this
         *         registration or {@code null} if no exception occurred.
         */
        Throwable getException();
}
```

5.3.2 ImportRegistration

```
/*
 * Copyright (c) OSGi Alliance (2009, 2013). All Rights Reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.osgi.service.remoteserviceadmin;


/**
```

```
 * An Import Registration associates an active proxy service to a remote
 * endpoint.
 *
 * The Import Registration can be used to delete the proxy associated with an
 * endpoint. It is created with the
 * {@link RemoteServiceAdmin#importService(EndpointDescription)} method.
 *
 * When this Import Registration has been closed, all methods must return
 * {@code null}.
 *
 * @ThreadSafe
 * @noimplement
 * @author $Id: 45e6ba488e7eb4fbdbb34959f2056dd35fa28283 $
 */
public interface ImportRegistration {
        /**
         * Return the Import Reference for the imported service.
         *
         * @return The Import Reference for this registration.
         * @throws IllegalStateException When this registration was not properly
         *         initialized. See {@link #getException()}.
         */
        ImportReference getImportReference();

        /**
         * Update the local service represented by this {@link ImportRegistration}.
         * After this method returns the {@link EndpointDescription} returned via
         * {@link #getImportReference()} must have been updated.
         *
         * @param endpoint The updated endpoint
         *
         * @throws IllegalStateException When this registration was not properly
         *         initialized. See {@link #getException()}.
         * @throws IllegalArgumentException When the supplied
         *         {@link EndpointDescription} does not represent the same endpoint
         *         as this {@link ImportRegistration}.
         *
         */
        void update(EndpointDescription endpoint);

        /**
         * Close this Import Registration. This must close the connection to the
         * endpoint and unregister the proxy. After this method returns, all other
         * methods must return {@code null}.
         *
         * This method has no effect when this registration has already been closed
         * or is being closed.
         */
        void close();

        /**
         * Return the exception for any error during the import process.
```

```
        *
        * If the Remote Service Admin for some reasons is unable to properly
        * initialize this registration, then it must return an exception from this
        * method. If no error occurred, this method must return {@code null}.
        *
        * The error must be set before this Import Registration is returned.
        * Asynchronously occurring errors must be reported to the log.
        *
        * @return The exception that occurred during the initialization of this
        *         registration or {@code null} if no exception occurred.
        */
        Throwable getException();
}
```

5.4 Clarify the uniqueness of EndpointDescription Id Strings

The following is an extract from the OSGi R5 compendium:

> **122.4.3 Endpoint Id**
>
> An Endpoint Id is an opaque unique identifier for an Endpoint. There is no syntax defined for this string except that white space at the beginning and ending must be ignored. The actual syntax for this Endpoint Id must be defined by the actual configuration type.
>
> Two Endpoint Descriptions are deemed identical when their Endpoint Id is equal. The Endpoint Ids must be compared as string compares with leading and trailing spaces removed. The Endpoint Description class must use the String class' hash Code from the Endpoint Id as its own hashCode.

Furthermore the OSGi R5 compendium states that:

> Two Endpoint Descriptions are deemed equal when their Endpoint Id is equal. The Endpoint Id is a mandatory property of an Endpoint Description, it is further described at *Endpoint Id* on page 709. The hash code is therefore also based on the Endpoint Id.
>
> **122.4.1 Validity**
>
> A valid Endpoint Description must at least satisfy the following assertions:
>
> - It must have a non-null Id that uniquely identifies the Endpoint

The extracts above are sufficient to require that:

1. The Endpoint Id is opaque, and has no declared format or syntax
2. The Endpoint Id defines the identity of an EndpointDescription, regardless of its other properties
3. The Endpoint Id is expected to be "unique", although the scope of this uniqueness is not expressly defined.

The Remote Service Admin specification states that there may be multiple Topology Managers, Distribution Providers and Discovery Providers active concurrently within a single framework. It is clear that if these implementations are to coexist then Endpoint Id uniqueness must hold within the OSGi framework. This applies regardless of the number of installed Distribution Providers, more than one of which may be exporting a given service.

In the Remote Service Admin Specification EndpointDescriptions are created by distribution providers, but are passed on to Topology Managers. Depending upon the Topology Manager's implementation these EndpointDescription objects may then be passed to other actors, such as Discovery Providers, via EndpointListener or EndpointEventListener services. Discovery providers then advertise EndpointDescription objects over the network. These advertisements result in the EndpointDescription being serialized and reconstituted on a remote machine.

The above scenario adds some constraints on the uniqueness of EndpointDescription Ids.

1. Portable Discovery Providers can only use the EndpointDescription Id to determine which endpoint is being announced, updated or revoked. This applies both when being notified of local and remote events.

2. Portable Topology Managers can only use the EndpointDescription Id to determine which EndpointDescription they are receiving an event for.

As EndpointDescription objects are made available remotely, and therefore shared between frameworks, the required scope of Id uniqueness required is larger than a single framework.

If three frameworks are connected by a Discovery provider, and two produce an EndpointDescription with the same Id, then two “identical” notification events will arrive at the third framework. If one of the two frameworks then destroys the endpoint, and advertises the service removal to the third framework, then the third framework will reach the incorrect conclusion that there are now zero endpoints available.

We can therefore state that the minimum scope of Endpoint Id uniqueness is that no two distinct endpoints should have the same Id within a connected group. Distribution Providers must ensure that they do not produce Endpoint Id clashes within a connected group.

5.4.1 Ensuring Endpoint Ids are sufficiently Unique

It should be noted that Discovery Providers can be added to a framework at any time, increasing the size of a connected group. New Distribution Providers that can support additional configuration types can also be added dynamically, as can Topology Managers with the ability to source EndpointDescriptions from XML, representing external services. This means that although an Endpoint Id must only be unique with a single connected group, the group can expand in size or number of EndpointDescription objects representing a given service at any time.

The simplest way to ensure that a growth in the number of EndpointDescriptions and/or the size of the connected group does not violate the required uniqueness of Endpoint Ids is for implementations to make their Endpoint Ids globally unique. This protects against clashes regardless of changes to the connected group.

Whilst globally unique identifiers are a simple solution to the Endpoint Id uniqueness problem, they are not easy to implement in all environments. In some systems they can be prohibitively expensive to create, or of insufficient entropy to be genuinely unique. Some distribution providers may therefore choose not to use random globally unique ids.

In the case where no globally unique value is used the following actions are recommended (although not required)

1. Distribution Providers protect against intra-framework clashes using some known value unique to the service, for example the service id.

2. Distribution Providers protect against inter-provider collisions within a single framework by using some unique value, such as the distribution provider's bundle id. The distribution provider bundle's symbolic name is insufficient, as there can be multiple versions of the same distribution provider installed within a single framework.

3. Distribution Providers protect against inter-framework collisions using some value unique to the framework, such as the framework UUID.

These suggestions are not intended to be normative, and no implementation should rely on certain values being contained within the id. Distribution Providers are free to generate Endpoint Id in any way, as long as it meets the required level of uniqueness.

6 Data Transfer Objects

RFC 185 defines Data Transfer Objects as a generic means for management solutions to interact with runtime entities in an OSGi Framework. DTOs provides a common, easily serializable representation of the technology.

For all new functionality added to the OSGi Framework the question should be asked: would this feature benefit from a DTO? The expectation is that in most cases it would.

The DTOs for the design in this RFC should be described here and if there are no DTOs being defined an explanation should be given explaining why this is not applicable in this case.

This section is optional and could also be provided in a separate RFC.

7 Javadoc

Please include Javadoc of any new APIs here, once the design has matured. Instructions on how to export Javadoc for inclusion in the RFC can be found here: https://www.osgi.org/members/RFC/Javadoc

8 Considered Alternatives

For posterity, record the design alternatives that were considered but rejected along with the reason for rejection. This is especially important for external/earlier solutions that were deemed not applicable.

9 Security Considerations

Description of all known vulnerabilities this may either introduce or address as well as scenarios of how the weaknesses could be circumvented.

10 Document Support

10.1 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

Add references simply by adding new items. You can then cross-refer to them by chosing <Insert><Cross Reference><Numbered Item> and then selecting the paragraph. ***STATIC REFERENCES (I.E. BODGED) ARE NOT ACCEPTABLE, SOMEONE WILL HAVE TO UPDATE THEM LATER, SO DO IT PROPERLY NOW.***

10.2 Author's Address

Name	David Bosschaert
Company	Red Hat
Address	
Voice	
e-mail	david@redhat.com

Name	Richard Nicholson
Company	Paremus
Address	
Voice	
e-mail	

Name	Tim Ward
Company	Paremus
Address	
Voice	
e-mail	tim.ward@paremus.com

10.3 Acronyms and Abbreviations

10.4 End of Document