RFC 203 – Remote Service Admin 1.1

Draft

16 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 9.1.

```
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
0.3	October, 2013	Update based on comments from 23/10 EEG call. Restructure sections to place background information in section 3. No changes to content.
0.4	November, 2013	Remove unused sections
0.5	December, 2013	Update Namespaces to separate out the extender section. Use three separate namespaces for the RSA actor capabilities
0.6	February, 2014	Update JavaDoc post Austin F2F. Minor changes to address bugs raised by RI and CT teams

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

In Remote Service Admin 1.0 it was possible for Topology Managers to dynamically register and unregister remote services, however there was no way to update a service without unregistering it. This causes significant disruption when remote clients are dependent upon the availability of the remote service for their operation.

It has also been found that part of the Remote Service Admin 1.0 specification wording relating to Endpoint Id uniqueness is insufficiently clear, in that it does not specify the scope of the uniqueness. This needs to be clarified so that different Topology Managers, Discovery Providers and Distribution Providers can reliably interoperate.

3.1 Notifying the Discovery Provider of updates

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.

3.2 Notifying the Distribution Provider of updates

In the Remote Service Admin specification the Topology Manager is 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 updates 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.

3.3 Clarifying Endpoint Id Uniqueness

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.

3.3.1 Required Scope of Endpoint Id Uniqueness

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.

3.3.2 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.

3.4 Make Event property names consistent

In section 122.7.1 of RSA 1.0 the specification defined property mappings for mapping RemoteServiceAdminEvent objects to OSGi Event Admin events. This includes the following:

- c a u s e – The exception, if present.

The constant “cause” does not match the normal pattern used by Event Admin, which would normally use the String “exception” as a key. There is also no constant defined by Remote Service Admin for “cause”.

3.5 Define capability namespaces for RSA components

The Remote Services Admin specification defines multiple actors. Discovery providers, Distribution Providers and Topology Managers. There is also an extender defined for consuming “local” XML endpoint descriptions from inside bundles.

These actors should provide well-defined capabilities so that they can be easily located in a repository, and to enable simpler provisioning of implementations. The local discovery extender should also define a capability in the osgi.extender namespace.

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

5.1 Discovering Remote Service Updates

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.

5.1.1 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.

5.1.2 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 matchedFfilter);
}
```

```
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;
  }
}
```

5.2 Updating Exported and Imported Services

To support requirement RSA02 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.

5.2.1 ExportRegistration

The ExportRegistration represents an OSGi service that has been exported by a Distribution Provider. It is the Topology Manager's role to track and manage which services are exported, therefore an update method should be added to this interface. The Topology Manager should also have an opportunity to change the set of properties that were passed when originally creating the ExportRegistration

```
/**
 * 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 or
		 *         null if there was a failure updating the endpoint. If a failure
		 *         occurs then it can be accessed using {@link #getException()}.
		 * @throws IllegalStateException If this registration is closed, or when
		 *         this registration was not properly initialized. See
		 *         {@link #getException()}.
		 *
		 * @since 1.1
		 */
		EndpointDescription update(Map<String, ?> properties);
```

5.2.2 ImportRegistration

The ExportRegistration represents an OSGi service that has been imported into the framework by a Distribution Provider. It is the Topology Manager's role to track and manage which services are imported, but not to actually register the imported service, therefore an update method should be added to this interface. The Topology Manager must be able to pass the new EndpointDescription for this imported service.

```
		/**
		 * 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
		 *
		 * @return <code>true</code> if the endpoint was successfully updated,
		 *         <code>false</code> otherwise. If the update fails then the
		 *         failure can be retrieved from {@link #getException()}.
		 *
		 * @throws IllegalStateException When this registration is closed, or if it
		 *         was not properly initialized. See {@link #getException()}.
		 * @throws IllegalArgumentException When the supplied
		 *         {@link EndpointDescription} does not represent the same endpoint
		 *         as this {@link ImportRegistration}.
		 *
		 * @since 1.1
		 */
```

~~void~~boolean update(EndpointDescription endpoint);

5.2.3 Update events for RemoteServiceAdmin

The Remote Service Admin specification defines RemoteServiceAdminEvent and RemoteServiceAdminEventListener types which are used to provide notifications when Endpoints are added and removed. This RFC defines two new event types, one for ExportRegistration update, and one for ImportRegistration update. These events should be issued when the relevant update succeeds. If an update fails either with an error or a warning, then the relavant error or warning event should be issued.

5.3 Clarify the uniqueness of EndpointDescription Id Strings

The RSA 1.1 specification should clarify the uniqueness of EndpointDescription Id Strings to state that they are required to be unique within the connected set of frameworks. This means that no two distribution providers connected by a discovery service should produce the same Id String unless the EndpointDescription is for the same endpoint.

5.3.1 Producing a suitably Unique identifier

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 may 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 may be included in the specification, but 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.

5.4 Event Admin property mapping

The event mappings defined in Remote Service Admin 1.0 were missing constant definitions, and also inconsistent with normal mappings. To remedy this this RFC proposes that:

- Implementations continue to map failure exceptions to the “cause” property. This ensures backward compatibility between versions of the specification. There will continue to be no constant defined for this key.

- RSA 1.1 Implementations must also map failure exceptions to the “exception” property defined by Event Admin. This is expected to be the normal location that event receivers will use to find exception values.

5.5 Generic Capability namespaces

This RFC suggests that implementations make use of Generic Capabilities to describe their interaction within the Remote Services Admin specification.

5.5.1 The Local Discovery Extender

The RSA specification defines an extender mechanism for local discovery of EndpointDescription XML files. Currently there is no defined mechanism for EndpointDescription bundles to declare that they need this extender to be present. RSA 1.1 therefore defines the value 'osgi.remoteserviceadmin' for the 'osgi.extender' capability. This capability should be provided by Local Discovery extenders, and required by bundles that provide EndpointXML descriptions to the extender. The version of the provided capability is defined to be 1.1, to match the version of the specification that defined the capability.

Example capability for an implementation of the extender:

```
Provide-Capability:      osgi.extender;      osgi.extender=osgi.remoteserviceadmin;
version:Version=1.1
```

Example requirement from a bundle that contains Endpoint XML files:

```
Require-Capability: osgi.extender; filter:=(&(osgi.extender=osgi.remoteserviceadmin)
(version>=1.1)); effective:=active
```

5.5.2 The Discovery Provider Capability

Discovery Providers use the 'osgi.remoteserviceadmin.discovery' namespace to declare themselves as a discovery provider. The version defined for this namespace is version 1.1, and indicates that the discovery provider meets the RSA 1.1 specification requrirements for discovery providers.

For Discovery providers there is an additional defined attribute, '`protocols`'. The protocols attribute is of type List<String> and contains a list of the discovery protocols supported by the discovery provider. Local discovery providers (using the Endpoint XML extender), should use the value '`local`' to indicate that they support local discovery. Other values for the protocols attribute are implementation specific.

The primary motivation for the 'osgi.remoteserviceadmin.discovery' capability is to allow for provisioning from OSGi Bundle Repositories using the Repository API. Bundles that expose remote services should not state a requirement for a Topology Manager capability.

Example: A discovery provider that provides local and SLP discovery:

```
Provide-
Capability:osgi.remoteserviceadmin.discovery;protocols:List<String>=”SLP,local”;
version:Version=1.1
```

5.5.3 The Distribution Provider capability

Remote Service Admin distribution providers advertise their supported distribution mechanisms at runtime using configurations. These are selected at runtime using the `service.exported.configs` service property. The Generic Capability for distribution providers therefore defines a namespace 'osgi.remoteserviceadmin.distribution'

with an attribute “configs”, of type List<String>, that a distribution provider can use to advertise the configs that it supports. The version defined for this namespace is version 1.1, and indicates that the distribution provider meets the RSA 1.1 specification requrirements for distribution providers.

The primary motivation for the 'osgi.remoteserviceadmin.distribution' capability is to allow for provisioning from OSGi Bundle Repositories using the Repository API. Bundles that expose remote services should not state a requirement for a Distribution Provider capability.

Example: A Distribution provider that provides JAX-WS and JAX-RS as distribution mechanisms:

```
Provide-Capability: osgi.remoteserviceadmin.distribution;
configs:List<String>=”jax-ws,jax-rs”; version:Version=1.1
```

5.5.4 The Topology Manager Capability

Remote Service Admin topology managers may use different policies when determining which services to import. The Generic capabilitiy namespace for topology managers is 'osgi.remoteserviceadmin.topology' and therefore includes the `policy` attribute. This attribute has a String value that may be one of the specification suggested policies, such as 'promiscuous' or 'fail-over', or a custom mechanism. The version defined for this namespace is version 1.1, and indicates that the distribution provider meets the RSA 1.1 specification requrirements for topology managers.

The primary motivation for the 'osgi.remoteserviceadmin.topology' capability is to allow for provisioning from OSGi Bundle Repositories using the Repository API. Bundles that expose remote services should not state a requirement for a Topology Manager capability.

Example: A Topology Manager that uses the promiscuous policy

```
Provide-Capability: osgi.remoteserviceadmin.topology; policy=promiscuous;
version:Version=1.1
```

6 Data Transfer Objects

The updates defined in this RFC do not require any additional DTOs

7 Considered Alternatives

7.1.1 The osgi.remoteserviceadmin namespace

The osgi.remoteserviceadmin namespace uses the `osgi.remoteserviceadmin` attribute to indicate the type of actor that the bundle is. There are three defined values for the `osgi.remoteserviceadmin` attribute: '`discovery`', '`distribution`' and '`topology`'. The initial version of the osgi.remoteserviceadmin namespace is 1.1, to match the version of the RSA specification that introduced it, and is declared using the `version` attribute, of type Version.

Discovery Providers

For Discovery providers there is an additional defined attribute, '`protocols`'. The protocols attribute is of type List<String> and contains a list of the discovery protocols supported by the discovery provider. Local discovery providers (using the Endpoint XML extender), should use the value '`local`' to indicate that they support local discovery. Other values for the protocols attribute are implementation specific.

For consistency, Local Discovery extenders should also declare their extender capabiliity using the osgi.extender namespace and the attribute value 'osgi.remoteserviceadmin'. The extender capability should be declared at version 1.1

Example: A discovery provider that provides local and SLP discovery:

```
Provide-Capability: osgi.extender; osgi.extender=osgi.remoteserviceadmin;
version:Version=1.1, osgi.remoteserviceadmin; osgi.remoteserviceadmin=discovery;
protocols:List<String>=”SLP,local”; version:Version=1.1
```

Distribution Providers

Remote Service Admin distribution providers advertise their supported distribution mechanisms using configurations. These are selected at runtime using the `service.exported.configs` service property. The Generic Capability for distribution providers therefore defines an attribute “`configs`”, of type List<String>, that a distribution provider can use to advertise the configs that it supports.

Example: A Distribution provider that provides JAX-WS and JAX-RS as distribution mechanisms:

```
Provide-Capability: osgi.remoteserviceadmin; osgi.remoteserviceadmin=distribution;
configs:List<String>=”jax-ws,jax-rs”; version:Version=1.1
```

Topology Managers

Remote Service Admin topology managers may use different policies when determining which services to import. The Generic capabilitiy for topology managers should therefore include the `policy` attribute. This attribute has a String value that may be one of the specification suggested policies, such as promiscuous or fail-over, or a custom mechanism.

Example: A Topology Manager that uses the promiscuous policy

```
Provide-Capability: osgi.remoteserviceadmin; osgi.remoteserviceadmin=topology;
policy=promiscuous; version:Version=1.1
```

8 Security Considerations

The proposed changes in the RFC do not require any security changes

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	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

9.3 Acronyms and Abbreviations

9.4 End of Document