RFC-227 Configuration Admin Updates

Draft

13 Pages

Abstract

10 point Arial Centered.

OSGi Configuration Admin is a slightly pedantic but highly effective flexible standardized model to configure applications. This RFC seeks a solution to carry configuration information in a bundle.

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 10.1.

`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		*Initial Version* *Carsten Ziegeler, Adobe <cziegele@adobe.com>*

1 Introduction

This RFC collects a numbers of requested enhancements to Configuration Admin Service that were suggested after Release 6 design work was completed. Some of the requirements are extracted from RFC 218 Configurator.

2 Application Domain

The Configuration Admin Specification was last change as part of Release 5. From that specification:

> The Configuration Admin service is an important aspect of the deployment of an OSGi framework. It allows an Operator to configure deployed bundles. Configuring is the process of defining the configuration data for bundles and assuring that those bundles receive that data when they are active in the OSGi framework.

3 Problem Description

3.1 RFC 218 Configurator

RFC 218 defines the Configurator, an extender that allows the storage of configuration data in a bundle. Some of the requirements from that RFC can best be realized with new features/requirements for the Configuration Admin Service.

3.2 Support ConfigurationPlugin-like behavior for non-MS/MSF users (Bug 2908)

The ConfigurationPlugin defined in the Configuration Admin Service Specification is invoked before a configuration is delivered to a ManagedService(Factory). The plugin is able to modify the configuration properties. There are several use cases like replacing configuration values with values provided through system properties (or similar mechanism), decode values, or provide additional properties.

While this works with registering a ManagedService(Factory), component containers like Declarative Services or Blueprint are not required to register ManagedServices on behalf of their components. Therefore whether the ConfigurationPlugin mechanism works with such containers is implementation dependent and not specified.

4 Requirements

4.1 Configuration Admin

- C0010 – It must be possible to specify the service.pid value when creating a factory configuration. This implies the need for a get_or_create factory configuration method in ConfigurationAdmin. (RFC 218)
- C0020 – It must be possible to prevent the updating of a configuration by the runtime even the developer forced it. (RFC 218)
- C0030 – It must be possible to avoid any action if a configuration is updated with the exact samef properties and values as it already has (RFC 218)
- C0040 - Support ConfigurationPlugin-like behavior for non-MS/MSF users

5 Technical Solution

5.1 PID Handling of Factory Configurations

The current Configuration Admin Service Specification provides no control over the PID of a factory configuration: a new factory configuration gets assigned a PID generated by the *Configuration Admin*. This makes it hard for any (provisioning) tool to manage such a configuration as it needs to store this generated PID in order to later on update or delete the factory configuration.

Configuration Admin specifies that a “PID should follow the symbolic-name syntax”, however in the examples in table 104.1 non symbolic-names are used. For targeted PIDs it's already defined that the pipe character '|' is used to separate the PID part from the target information which in fact means that a PID must not use this character. This should be more precisely specified in section 104.3.1. In the same way as the pipe character has been

introduced as a special character the character '#' is introduced as another special character for the PID handling of factory configurations.

By introducing two new methods on the *Configuration Admin* service, a client of the service can specify the PID of the factory configuration by providing the factory PID and a name:

```
public Configuration getFactoryConfiguration(String factoryPid, String name,
String location) throws IOException;

public Configuration getFactoryConfiguration(String factoryPid, String name)
throws IOException;
```

These methods require a factoryPID and a name argument. The method still generates a PID however the generated PID has the form: `factoryPid#name`. This ensures that the *Configuration Admin* can still guarantee uniqueness of the PID. If a configuration with the given combination of factoryPID and name already exists, this is returned, otherwise a new factory configuration is returned. The returned configuration has the factoryPID and a generated PID as mentioned above. Location handling, binding and permission checks works as defined for `getConfiguration`.

The name can be used to find a factory configuration using `listConfigurations`:

```
ConfigurationAdmin ca = …;
ca.listConfigurations("(service.pid=my.factory.pid#myname)");
```

5.2 Locking Configuration Records

The `Configuration` interface is enhanced with the following methods:

```
/**
 * Locks or unlocks the configuration.
 * @param flag If {@code true} the configuration is locked,
 *             if {@code false} the configuration is unlocked.
 * @throws IOException If the new lock state cannot be persisted.
 * @throws IllegalStateException If this configuration has been deleted.
 */
public void setLocked(boolean flag) throws IOException;

/**
 * Check if the configuration is currently locked.
 * @return {@code true} if the configuration is locked, {
 *         {@code false} otherwise.
 * @throws IllegalStateException If this configuration has been deleted.
 */
public boolean isLocked();
```

If the configuration is locked using `setLocked(true)` this state is persisted and the configuration remains locked until it is explicitly unlocked calling `setLocked(false)`. If the configuration is locked and Configuration.update(), Configuration.update(Dictionary) or `Configuration.delete()` is called, a `LockedConfigurationException` (subclass of IOException) is thrown.

5.2.1 Security impacts

A new action, LOCK, is added to the configuration permission. In order to lock or unlock a configuration, the caller needs the permission to do so. The verification of the permission is handled in the same way as for the CONFIGURE action as outline in section 104.11.1.

5.3 Improving Configuration Updates

Currently, any call of the update method on the Configuration object assumes that the configuration actually changed. The change count is increased, listeners and managed service (factories) are informed.

Configuration Admin should actually check whether the updated configuration is the same as the previous configuration, and if so ignore this operation. This allows all (provisioning) clients to simply update a configuration without reimplementing a complicated change check, Doing it once in Configuration Admin is more efficient and improves the handling for every client.

As configurations should only contain a limited set of types, equals can be called on the property values to find out whether the values are the same. For arrays, equals need to be called on each member of the array.

```
/**
 * Update the properties of this {@code Configuration} object if the
 * provided properties are different than the currently stored set
 *
 * If the properties are the same, no operation is performed, otherwise it
 * stores the properties in persistent storage after adding or overwriting
 * the following properties:
 * <ul>
 * <li>"service.pid" : is set to be the PID of this configuration.</li>
 * <li>"service.factoryPid" : if this is a factory configuration it is set
 * to the factory PID else it is not set.</li>
 * </ul>
 * These system properties are all of type {@code String}.
 *
 * <p>
 * If the corresponding Managed Service/Managed Service Factory is
 * registered, its updated method must be called asynchronously. Else, this
 * callback is delayed until aforementioned registration occurs.
 *
 * <p>
 * Also notifies all Configuration Listeners with a
 * {@link ConfigurationEvent#CM_UPDATED} event.
 *
 * @param properties the new set of properties for this configuration
 * @throws LockedConfigurationException If the configuration is locked
 * @throws IOException if update cannot be made persistent
 * @throws IllegalArgumentException if the {@code Dictionary} object
 *         contains invalid configuration types or contains case variants of
 *         the same key name.
 * @throws IllegalStateException If this configuration has been deleted.
 */
public void setProperties(Dictionary<String, ?> properties) throws IOException;
```

5.4 Capabilities

The Configuration Admin implementation bundle must provide the osgi.implementation capability with name osgi.cm. This capability can be used by provisioning tools and during resolution to ensure that a Configuration Admin implementation is present to manage configurations. The capability must also declare a uses constraint for the org.osgi.service.cm package and provide the version of this specification:

```
Provide-Capability: osgi.implementation;
                    osgi.implementation="osgi.cm";
                    uses:="org.osgi.service.cm";
                    version:Version="1.6"
```

This capability must follow the rules defined for the osgi.implementation Namespace.

The bundle providing the Configuration Admin service must provide a capability in the osgi.service namespace representing this service. This capability must also declare a uses constraint for the org.osgi.service.cm package:

```
Provide-Capability: osgi.service;
                    objectClass:List<String>="org.osgi.service.cm.ConfigurationAdmin";
                    uses:="org.osgi.service.cm"
```

This capability must follow the rules defined for the osgi.service Namespace.

5.5 Configuration Transformers

The Configuration Admin service allows third-party applications to participate in the configurationprocess. Bundles that register a service object under a ConfigurationTransformer interface can transform the configuration dictionary just before it reaches the configuration target service.

5.5.1 Configuration Transformation

Clients of Configuration Admin which consume configurations like the Declarative Service Implementation or the Blueprint Container implementation get the configuration either through registering a managed service or by getting the configuration from the Configuration Admin service.

Two alternatives: alternative A requires clients like DS to call a special method, alternative B requires clients to use a different interface to get “transparent” filtering:

ALTERNATIVE A:

In order to get the configuration transformed by the registered configuration transformer services, these clients call a new method on the Configuration Admin Service named “transform” passing in the bundle which will consume the configuration and a dictionary with the properties. Configuration admin calls the transformers as outlined below. For example for Declarative Services the consuming bundle is the bundle declaring the component which receives the configuration. Calling the method requires a new permission (TBD).

ALTERNATIVE B:

A new interface FOOConfigurationAdmin (couldn’t come up with a good name) is added which is an empty extension of the ConfigurationAdmin interface. ConfigurationAdmin registers a service as a service factory with this interface. This service acts in the same way as ConfigurationAdmin with the difference that every configuration that is handled out ist first transformed by calling all configuration transformers.

Clients of configuration manager like Declarative Services which get configurations from Configuration Admin on behalf of a target service and pass it to the target service, get the extended Configuration Admin service using the bundle context of bundle defining the service instead of the Configuration Admin service. Normal service permission for getting the special service is sufficient.

5.5.2 Configuration Transformer Service

The ConfigurationTransformer interface has two methods which inspects or modifies configuration data. The first one is used for transforming configuration properties before they hit the target service:

transform(Bundle, Dictionary).

All transformers in the service registry must be traversed and called. The changes made by a transformer must be visible to transformers that are called later. The transformers are sorted by ServiceReference (based on service ranking and service id) and are called in descending order.

The changes made by the transformer should normally not be validated, are dynamic and must not be stored. However, the Configuration Admin must ignore changes to the automatic properties as described in Automatic Properties.

TODO: bundleLocation is not passed on to ConfigurationPlugin – should we do the same here as well?

The second method is

prepare(Dictionary)

This method is called whenever configuration properties are updated right before they are persisted. This allows third-party plugins to filter out/mask properties. These values can be restored when the configuration is delivered to the target service using the transform() method.

All transformers in the service registry must be traversed and called. The changes made by a transformer must be visible to transformers that are called later. The transformers are sorted by ServiceReference (based on service ranking and service id) and are called in descending order.

5.5.3 Filtering

The transformer can register itself with the service property “cm.target” containing either a String with a PID or a String[] of PIDs. A Configuration Admin service must call a Configuration Transformer service only when this property is not set, or the target service's PID is listed in this property.

5.5.4 Permissions

Bundles registering ConfigurationTransformer objects must have ServicePermission[ConfigurationPlugin, REGISTER]. The Configuration Admin service must trust all services registered with the ConfigurationTransformer interface.

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

8.1

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

10.2 Author's Address

Name	Carsten Ziegeler
Company	Adobe Systems Incorporated
Address	Barfüsserplatz 6, 4055 Basel, Switzerland
Voice	+41 61 226 55 0
e-mail	cziegele@adobe.com

10.3 Acronyms and Abbreviations

10.4 End of Document