RFC 67: Exception Handling in Initial Provisioning

Confidential, Draft

7 Pages

Abstract

Most of the exceptions that occur during Initial Provisioning usually happen after the call that initiated the action has returned. This makes it impossible to throw the exception back to the caller using the normal exception mechanisms. This RFC proposes a standard way of using the LogService to report exceptions that occur during Initial Provisioning in such a way that bundles using Initial Provisioning can react to exceptions that occur.

Copyright © OSGi 2003.

This contribution is made to the Open Services Gateway Initiative (OSGI) as MEMBER LICENSED MATERIALS pursuant to the terms of the OSGI membership agreement and specifically the license rights and warranty disclaimers as set forth in Sections 3.2 and 12.1, respectively.

All company, brand and product names contained within this document may be trademarks that are the sole property of the respective owners.

The above notice must be included on all copies of this document that are made.

0 Document Information

0.1 Table of Contents

0.2 Status

This document specifies implementation requirements for the ProvisioningService of the Open Services Gateway Initiative, and requests discussion and suggestions for improvements. Distribution of this document is unlimited within OSGi.

0.3 Acknowledgement

We would like to thank Peter Kriens for his dogged insistence that this problem be addressed.

0.4 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 8.1.

```
Source code is shown in this typeface.
```

0.5 Revision History

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

Revision	Date	Comments
Initial	06/12/03	Initial Revision
0.2	08/01/03	Added types to the ProvisioningException. breed@almaden.ibm.com
0.3	09/09/03	Changed from using exceptions to using an entry in the provisioning information. breed@almaden.ibm.com
0.4	12/01/03	Added Unknown error code.

1 Introduction

Initial Provisioning provides a way to incrementally add provisioning information to a service platform and install new bundles. It is designed to allow the provisioning to proceed in various stages by allowing each stage to trigger the next. Each stage is begun when new provisioning information is loaded into the platform by the provisioning bundle. It is important to note that when provisioning information is changed to trigger the retrieval of the provisioning information for the next stage, the actual retrieval happens asynchronously with respect to the change of the information. This means that there is no receiver for any exceptions that may occur during the retrieval or processing of the new provisioning information.

Rather than add new classes to implement a listener pattern for receiving provisioning exceptions, this RFC proposes to add an entry into provisioning information to indicate an error.

2 Application Domain

Platform provisioning takes a variety of forms depending on the deployment. One simple case is a box that an end user buys, takes home, turns on, and is presented a web page with a form that takes a URL and some sort of identification, which is provided to her by a service provider. In another scenario the user will have a smart card that will be inserted into a device and the provisioning information will be retrieved from the smart card. Other scenarios may have the provisioning done by the service provider using a proprietary interface to put in at least some of the provisioning information.

One thing these scenarios all have in common is the need to report problems. In the first scenario the URL or identification information may be invalid. In the second the smart card may be bad or the provider unreachable. Server errors or data corruption errors will affect even the proprietary interface.

How the errors are reported may vary greatly. For the first scenario, the error would be reported via a web page to the user. The second scenario may employ a LCD readout on the device, and the third may use an event mechanism in the proprietary interface.

3 Problem Description

A standard way to publish provisioning exceptions is needed.

The Initial Provisioning reference implementation of the R3 OSGi specification has the following cases for throwing exceptions:

1. Couldn't load or save provisioning information.

2. Couldn't get PermissionAdmin.

3. A bundle to be started does not exist.

4. A bundle couldn't be started or installed.

5. Missing or invalid MIME type.

6. MalformedURLException.

7. IOException when retrieving document of a URL.

8. Corrupted ZipInputStream.

These eight cases can be further consolidated by type of exception thrown:

1. IOException when reading or writing provisioning information.

2. Exceptionless problems: “Couldn't get PermissionAdmin” and “Bundle to be started is not installed” and missing extra field.

3. BundleException when starting or installing a bundle.

4. IOException when retrieving or processing a provisioning zip file.

Note further that only types 1 and 4 are exceptions that halt the provisioning process. These two types of exceptions are the ones that we need to address in this RFC. The other types should be logged using the LogService if available.

4 Requirements

A properly designed exception handling mechanism will have the following properties:

1. Enough information should be present and presented in such a way that the problem can be programically diagnosed.

2. The handling mechanism should not assume a single receiver of exceptions.

3. Because initial provisioning is implemented on an unprovisioned platform it should minimize its use of optional services.

5 Technical Solution

When a fatal exception occurs in the provisioning process we add an entry to the provisioning information with the name of **provisioning.error** and the value of a string that has a numeric error code followed by a human readable string describing the error. The code and human readable string must be separated by a space.

These are the possible error codes:

0 = Unknown error

1 = Couldn't load or save provisioning information

2 = MalformedURLException

3 = IOException when retrieving document of a URL

4 = Corrupted ZipInputStream

When a new **provisioning.error** entry is added to the provisioning information, the **provisioning.update.count** will be incremented as normal.

6 Considered Alternatives

We considered using the LogService. We would have to add a ProvisioningException class in order to log the exceptions and errors properly. A dependency one LogService would also have been introduced. This simplier scheme was adopted for the fatal errors. Other errors may be logged to the LogService, if available, in an implementation specific way.

The alternative to using the LogService would be to implement a ProvisioningListener class. This proposal was rejected to avoid proliferation of services and related classes.

7 Security Considerations

There are no additional security exposures in this proposal. Any bundle using the mechanisms described here must already have access to the ProvisioningService.

8 Document Support

8.1 References

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

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

8.2 Author's Address

Name	Benjamin Reed
Company	IBM
Address	650 Harry Rd, San Jose, CA 95120
Voice	408 927-1811
e-mail	breed@almaden.ibm.com

8.3 Acronyms and Abbreviations

8.4 End of Document