RFC 0074 Manifest Localization

Confidential, Final Review Draft

9 Pages

Abstract

A solution to enable localization of bundle manifest information.

0 Document Information

0.1 Table of Contents

0.2 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in [1].

`Source code is shown in this typeface.`

All Page Within This Box

0.3 Revision History

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

Revision	Date	Comments
Initial	Jan 23 2003	First Draft Thomas Watson, IBM, tjwatson@us.ibm.com Jennifer Fogell, IBM, jfogell@us.ibm.com BJ Hargrave, IBM, hargrave@us.ibm.com
2nd draft	4 Feb 2004	Address comments received at Jan 2004 EG meeting in Austin, TX. Jennifer Fogell, IBM, jfogell@us.ibm.com BJ Hargrave, IBM, hargrave@us.ibm.com
3rd draft	24 Feb 2004	Address comments received at Jan 11th 2004 EG conference call -Removed sentence about having permission to access localization file -Added constants for “Bundle-Localization” and “META-INF/bundle” Jennifer Fogell, IBM, jfogell@us.ibm.com BJ Hargrave, IBM, hargrave@us.ibm.com
Final Review Draft	14 April 2004	No content changes since 3rd draft. Changed to Final Review Draft.
Final Review 2nd Draft	30 April 2004	Addressed localization file search order in fragments. Jennifer Fogell, IBM, jfogell@us.ibm.com
Final Review 3rd Draft	8 May 2004	Corrected constant names to match RFC 71. BJ Hargrave, IBM, hargrave@us.ibm.com
Final Review 4th Draft	8 June 2004	Corrected the localization key lookup order to match that of java.util.ResourceBundle. This issue was raised because the specification did not properly allow a less specific localization to “back” up a more specific localization. BJ Hargrave, IBM, hargrave@us.ibm.com

All Page Within This Box

1 Introduction

The bundle manifest contains human readable information. The OSGi framework needs to be enhanced to support localization of bundle manifest information. This RFC describes the additional requirements, enhancements and design of the localization of the bundle manifest information.

2 Application Domain

Localization is an important attribute of any product that must support a global audience. The OSGi framework specification is important in the worldwide marketplace. Thus the framework should have the necessary support to allow localization of human readable information stored in framework defined area such as the bundle manifest headers.

3 Problem Description

The current bundle manifest contains information about the bundle. Some of this information is human readable. However, there is no provision for this information to be localized. There is no means for this information to be localized when retrieved such as via Bundle.getHeaders.

4 Requirements

4.1 Localization of manifest information

The framework should provide a mechanism for information in the OSGi defined manifest headers to be localized such that the Dictionary returned by Bundle.getHeaders returns localized information.

4.1.1 Use Cases

The Bundle-Description manifest header contains human readable information that could be displayed to a user when requesting user interaction regarding the install or update of a bundle. This information needs to be localizable so that the appropriate localized version can be displayed for the user.

5 Technical Solution

To put the proposed changes in context, this RFC has copied sections from the OSGi R3 spec and modified them to reflect the proposed specification changes. The respective sections of the R3 spec are referenced in the titles of the sections below.

5.1 Manifest Headers[3]

A bundle can carry descriptive information about itself in the manifest file that is contained in its JAR file under the name of META-INF/MANIFEST.MF.

The Framework defines OSGi manifest headers such as Export-Package and Bundle-Activator, which bundle developers can use to supply descriptive information about a bundle. Manifest headers must strictly follow the rules for manifest headers as defined in Manifest Format.

All manifest headers are optional and any standard manifest headers not specified have no value by default (except for Bundle-Classpath that has dot ('.', \u002E) as default when no value is specified).

All manifest headers that may be declared in a bundle's manifest file are listed in Table 2, "Manifest Headers," on page ?.

A Framework implementation must:

• Process the main section of the manifest. Individual sections of the manifest may be ignored.

• Ignore unrecognized manifest headers. Additional manifest headers may be defined by the bundle developer as needed.

All Page Within This Box

• Ignore unknown attributes on OSGi-defined manifest headers.

5.1.1 Retrieving Manifest Headers[4]

The Bundle interface defines the following methods to retrieve manifest header information:

- getHeaders() - Returns a Dictionary object that contains the bundle's manifest headers and values as key/value pairs. The values returned are localized according to the locale returned by java.util.Locale.getDefault.

- getHeaders(String locale). - Returns a Dictionary object that contains the bundle's manifest headers and values as key/value pairs. The values returned are localized using the specified locale. If the specified locale is `null` then the locale returned by java.util.Locale.getDefault is used. If the specified locale is the empty string, this method will return the raw (unlocalized) manifest headers including any leading '%'.

The getHeaders() methods must continue to provide the manifest header information after the bundle enters the UNINSTALLED state. After the bundle has been uninstalled, this method will only return manifest headers localized for the default locale at the time the bundle was uninstalled.

A framework implementation must use only the raw (unlocalized) manifest headers when processing manifest headers that have semantic meaning.

5.1.2 Bundle Localization Files

Localized values of the headers are stored in properties files within the bundle. The default base name of the bundle localization property files is "META-INF/bundle". The bundle manifest header "Bundle-Localization" may optionally define another base name for the localization files. This location is relative to the root of the bundle.

The localization files must be loaded only from the bundle itself or one of it's fragments irrespective of classloaders. For example, the bundle will still be searched for localization files even if dot ('.') is not in the classpath. A localization file contains key/value entries for localized information. Each header in a bundle's manifest may be localized with the exception of any bundle manifest header which has a specified semantic meaning as defined in the OSGi specification. A localization key can be specified as the value of a bundle's manifest header using the following syntax:

```
header-value ::= localization-key | <any valid manifest value that does not start
with '%'>
localization-key ::=  '%' key
key := <a valid key in a properties file, see java.util.Properties>
```

For example, consider the following bundle manifest entries:

```
Bundle-Name: %acmebundle
Bundle-Vendor: %acmecorporation
Bundle-Description: %acmedescription
Bundle-Activator: com.acme.bundle.Activator
Bundle-Localization: localization/bundlemanifest
Acme-Defined-Header: %acme special header
```

User defined headers can also be localized. Spaces in the localization keys are allowed.

The above manifest entries could be localized by the following entries in the manifest localization file localization/bundlemanifest.properties.

```
acmebundle=The ACME Bundle
acmecorporation=The ACME Corporation
acmedescription=The ACME Bundle provides all of the ACME services
```

All Page Within This Box

```
acme special header=User Defined Acme Data
```

The above manifest entries could also have French localizations in the manifest localization file localization/bundlemanifest_fr_FR.properties.

If the Bundle-Localization header is not present, the base name of the manifest localization file is: META-INF/bundle.

Notice that the Bundle-Activator does not have an entry in the localization file. Only the headers that do not have semantic meaning to the Framework can be localized using a bundle localization file.

The following list contains some of the bundle manifest headers that have semantic meaning to the Framework and cannot be localized:

```
Bundle-Activator
Bundle-ClassPath
Bundle-Localization
Bundle-NativeCode
Bundle-RequiredExecutionEnvironment
Bundle-UpdateLocation
Bundle-Version
DynamicImport-Package
Export-Package
Import-Package
```

Other headers that have semantic meaning to the Framework cannot be localized.

5.1.2.1 Localization File Families

Bundle localization files belong to a family that shares a common localization file base name with additional suffixes which are derived from a locale name. The Framework searches for bundle localization files by appending suffixes to the localization file base name according to the specified locale or the default locale as returned by java.util.Locale.getDefault. When locating a key within the family of localization files, the files must be examined from most specific suffix to least specific suffix for the given locale. This allows localization files for more specific locales to contain different localizations or to not contain a localization so that it may be provided by the localization file for a less specific locale. The order for locating keys in a family of bundle localization files is the order as specified by the java.util.ResourceBundle.getBundle method.

5.1.2.2 Finding Localization Files

When searching for a localization file of a host bundle, the Framework must first look in the bundle and then look in the currently attached fragment bundles. When searching for a localization file of a fragment bundle, the Framework must first look in the fragment's host bundle and then look in the host's currently attached fragment bundles. If a fragment is attached to more than one host, the search will only include the first host (that is the host bundle with the lowest bundle id) and its' attached fragments. If the bundle (either a host or a fragment) is not resolved, then only the bundle itself should be searched. If no localization file is found, the Dictionary returned by Bundle.getHeaders will return the raw values as specified in the manifest header values without the leading '%. If an appropriate localization file is found but no value is found for a specific localization key, the raw header value without the leading '%' will be returned in the Dictionary.

5.2 New framework constants

The constant “public static final String BUNDLE_LOCALIZATION = “Bundle-Localization”” will be added to the framework constants. It will be used as a Manifest header attribute to optionally specify the base name of the bundle manifest localization files.

The constant “public static final String BUNDLE_LOCALIZATION_DEFAULT_BASENAME = “META-INF/bundle”” will be added to the framework constants. It will be used as the default localization base name. It is highly recommended that fragment bundles specify a unique localization base name. Since the search for a fragment’s localization file will start at the host bundle, if the fragment uses the default localization base name, the host’s bundles localization file could be used instead of intended fragment localization file.

6 Considered Alternatives

Considered allowing the Framework to use the values defined in localization files for headers that have semantic meaning. For example, this could allow for a bundle to have a different BundleActivator depending on what the current default locale is. This opened up more questions like when would changes to the default locale take effect. Would the change occur on bundle restart/refresh. What would happen if the Bundle-Classpath changed. What if the import/export packages changed? In the end this seemed to add too much complexity and actually solved different requirements than what we are trying to address in this RFC. This RFC only seeks to localize non-semantic manifest headers.

Considered having the keys in the localization file be the bundle manifest headers instead of allowing user defined keys. This imposed limitations on the usage of localization files: duplicate values in the manifest would be required to have different keys in the localization file even when their values are the same; fragment bundles would be forced to specify a different localization file then their host to prevent key collision.

Considered a more complex definition of the key syntax to allow for variable substitution:

localization-key::=% key (‘{‘ var ‘}’)*

This would allow variables to be inserted into the string value present in the localization file. This approach would add complexity and would not gain much useful functionality since header values typically simple strings and they are constant so variable substitution is not really needed since there is no way the variables could change.

7 Security Considerations

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

[3]. OSGI Service Platform Release 3 Section 4.3 Manifest Headers

[4]. OSGI Service Platform Release 3 Section 4.3.1 Retrieving Manifest Headers

8.2 Author's Address

Name	Jennifer Fogell
Company	IBM
Address	11501 Burnet Rd, Austin, TX 78758 USA
Voice	+1 512 838 9246
e-mail	jfogell@us.ibm.com

8.3 Acronyms and Abbreviations

8.4 End of Document

All Page Within This Box