



Red Hat Fuse 7.13

迁移指南

迁移到 Red Hat Fuse 7.13

Red Hat Fuse 7.13 迁移指南

迁移到 Red Hat Fuse 7.13

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

将 Fuse 安装升级到最新版本的 Red Hat Fuse 时，请使用本指南帮助您。

目录

前言 3

使开源包含更多 4

第 1 章 在 OPENSIFT 上升级 FUSE 5

第 2 章 升级 FUSE ONLINE 6

第 3 章 升级到 SPRING BOOT 2 7

 3.1. 开始前 7

 3.2. 从 SPRING BOOT 1 升级到 SPRING BOOT 2 7

第 4 章 从 FABRIC8 MAVEN 插件迁移到 OPENSIFT MAVEN 插件 9

第 5 章 在 SPRING BOOT 独立升级 FUSE 应用程序 10

 5.1. CAMEL 迁移注意事项 10

 5.2. 关于 MAVEN 依赖项 12

 5.3. 更新 FUSE 项目的 MAVEN 依赖项 12

第 6 章 在 JBOSS EAP 独立上升级 FUSE 应用程序 15

 6.1. CAMEL 迁移注意事项 15

 6.2. 关于 MAVEN 依赖项 17

 6.3. 更新 FUSE 项目的 MAVEN 依赖项 18

 6.4. 升级 JAVA EE 依赖项 18

 6.5. 在 JBOSS EAP 安装中升级现有 FUSE 19

 6.6. 同时升级 FUSE 和 JBOSS EAP 19

第 7 章 在 KARAF 独立升级 FUSE 应用程序 21

 7.1. CAMEL 迁移注意事项 21

 7.2. 关于 MAVEN 依赖项 23

 7.3. 更新 FUSE 项目的 MAVEN 依赖项 23

第 8 章 在 KARAF 上升级 FUSE STANDALONE 25

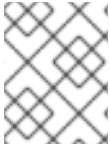
 8.1. 将 FUSE 升级到 KARAF 的影响 25

 8.2. 在 KARAF 上升级 FUSE STANDALONE 25

 8.3. 在 KARAF 上回滚 FUSE 的升级 27

前言

本指南提供有关更新 Red Hat Fuse 和 Fuse 应用程序的信息：



注意

如果您要从 Fuse 6 迁移到最新的 Fuse 7 版本，请在本指南中的说明操作，请按照 [Red Hat Fuse 7.0 迁移](#) 指南中的说明进行操作。

[第 2 章 升级 Fuse Online](#)

[第 5 章 在 Spring Boot 独立升级 Fuse 应用程序](#)

[第 4 章 从 Fabric8 Maven 插件迁移到 Openshift Maven 插件](#)

[第 6 章 在 JBoss EAP 独立升级 Fuse 应用程序](#)

[第 7 章 在 Karaf 独立升级 Fuse 应用程序](#)

[第 8 章 在 Karaf 上升级 Fuse Standalone](#)

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看我们的 [CTO Chris Wright 信息](#)。

第1章 在 OPENSIFT 上升级 FUSE

Fuse 7.12 包含相应的更新，使其能够与 OpenShift Container Platform (OCP) 4.9 或更高版本搭配使用。如果您计划升级到 OCP 4.10，则必须将 Fuse 升级到 7.11 版本，**然后才能将 OCP 升级到 4.10**。早期版本的 Fuse (prior 到 7.10)不支持 OCP 4.9 或更高版本。

如需更多信息，请参阅 [OpenShift 上的 Fuse 指南](#)。

第 2 章 升级 FUSE ONLINE

全新的应用程序镜像（包括补丁和安全修复）会针对 Fuse 发布。您会收到通过红帽的勘误更新频道通知这些更新。然后您可以升级 Fuse 镜像。

对于 OCP 4.x，请按照使用 OperatorHub (OCP 4.x)升级 Fuse 中的步骤，使用 OpenShift OperatorHub 从 Fuse 升级到 7.11。

您应该确定升级到 Fuse 7.11 是否需要现有集成进行更改。即使不需要更改，在升级 Fuse 时都必须重新发布任何正在运行的集成。

使用 OperatorHub (OCP 4.x)升级 Fuse.

使用 OpenShift OperatorHub 从 Fuse Online 7.10 升级到 7.11。

1. 如果要从 Fuse 7.9.x 升级到 Fuse Online 7.10.1，您必须首先手动升级到 Fuse 7.10.0，如从 **Fuse Online 7.9.x 升级到 7.10.1** 所述，需要手动升级步骤。
2. Fuse 7.11 需要 OpenShift Container Platform (OCP) 4.6 或更高版本。如果使用 OCP 4.5 或更早版本，则必须升级到 OCP 4.6 或更高版本，如果要升级到 Fuse 7.11。
3. 在 OCP 4.9 上，当您升级到 7.11 时，Fuse Operator 升级过程中会显示以下警告：
W1219 18:38:58.064578 1 warning.go:70] extensions/v1beta1 Ingress 在 v1.14+ 中弃用，在 v1.22+ 中不可用；使用 networking.k8s.io/v1 Ingress

此时会出现这个警告，因为客户端(Fuse 用于 Kubernetes/OpenShift API 初始化代码)访问已弃用的 Ingress 版本。这个警告不是使用已弃用 API 的指示器，且没有升级到 Fuse 7.11 的问题。

从 Fuse Online 7.10 或更早的版本升级到较新的 Fuse Online 7.11 版本的升级过程取决于您安装 Fuse Online 7.11 时选择的批准策略：

- 对于自动更新，当有新版本的 Fuse 操作器可用时，OpenShift Operator Lifecycle Manager (OLM)会自动升级 Fuse Online 的运行实例，而无需人为干预。
- 对于手动更新，当有新版 Operator 可用时，OLM 会创建更新请求。作为集群管理员，您必须手动批准该更新请求，将 Fuse Online operator 更新至新版本，如手动批准 OpenShift 文档的待处理 Operator 升级 部分所述。

在基础架构升级过程中和之后，现有集成将继续使用旧版本的 Fuse 库和依赖项运行。

要让现有集成与更新的 Fuse Online 版本运行，您必须重新发布集成。

第 3 章 升级到 SPRING BOOT 2

本章介绍了如何将应用程序从 Spring Boot 1 升级到 Spring Boot 2.0。

3.1. 开始前

在开始迁移到 Spring Boot 2 之前，您必须检查系统要求和依赖项。

- 升级到最新的 1.5.x 版本
 - 开始之前，请先升级到最新的 1.5.x 可用版本。这是为了确保您正在根据该行的最新依赖项进行构建。
- 查看依赖项
 - 迁移到 Spring Boot 2 将导致升级多个依赖项。查看带有 2.0.x 的依赖项管理 1.5.x 的依赖项管理，以评估您的项目会受到影响。
 - 识别不是由 Spring Boot 管理的依赖项的兼容版本，然后为这些定义显式版本。
- 查看自定义配置
 - 可能需要在升级过程中检查项目定义的任何自定义配置。如果它可以被使用标准自动配置替换，请在升级前执行此操作。
- 查看系统要求
 - Spring Boot 2.0 需要 Java 8 或更高版本。
 - 它还需要 Spring Framework 5.0。
 - Java 6 和 7 不再被支持。

3.2. 从 SPRING BOOT 1 升级到 SPRING BOOT 2

查看项目及其依赖项后，请升级到 Spring Boot 2.x 的最新维护版本。建议您在阶段升级。例如，首先从 Spring Boot 1.5 升级到 Spring Boot 2.0，然后升级到 2.1，然后升级到 Spring Boot 2 的最新维护版本。

迁移配置属性

使用 Spring Boot 2.0 时，许多配置属性被重命名或删除。因此，您需要相应地更新 `application.properties/application.yml`。您可以使用新的 `spring-boot-properties-migrator` 模块的帮助来实现这一点。将作为依赖项添加到项目后，这不仅可以分析应用程序的环境和在启动时打印诊断，而且还会在运行时暂时迁移属性。

流程

1. 将 `spring-boot-properties-migrator` 模块添加到项目的 `pom.xml` 的 `dependencies` 部分。

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-properties-migrator</artifactId>
<scope>runtime</scope>
</dependency>
```

```
runtime("org.springframework.boot:spring-boot-properties-migrator")
```



注意

完成迁移后，请确保从项目的依赖项中删除此模块。

第 4 章 从 FABRIC8 MAVEN 插件迁移到 OPENSIFT MAVEN 插件

`fabric8-maven-plugin` 已从 Fuse 7.11 中完全删除。我们建议您使用 `openshift-maven-plugin` 在 OpenShift 上的 Fuse 中构建和部署 Maven 项目。

流程

使用以下说明来更新应用程序，使其可以使用 `openshift-maven` 插件。

1. 将应用程序的 `src/main/fabric8` 目录重命名为 `src/main/jkube`。
2. 在项目的 `pom.xml` 中找到 `org.jboss.redhat-fuse:fabric8-maven-plugin` 依赖项，并将它更改为 `org.jboss.redhat-fuse:openshift-maven-plugin`。请参阅 [Sample pom.xml](#)。
3. 检查依赖项。例如，`org.arquillian.cube:arquillian-cube-openshift`, `org.jboss.arquillian.junit:arquillian-junit-container`, `io.fabric8:kubernetes-assertions` 不再用于我们的示例，可能不再需要。
4. 您可以创建一些示例测试，可用于反映迁移后的 API 更改。如需更多信息，请参阅 [Spring Boot Camel 快速启动](#) 中的示例测试。

其他资源

- [OpenShift Maven 插件](#).

第 5 章 在 SPRING BOOT 独立升级 FUSE 应用程序

在 Spring Boot 上升级 Fuse 应用程序：

- 您应该考虑 Apache Camel 更新，如 [第 5.1 节 “Camel 迁移注意事项”](#) 所述。
- 您必须更新 Fuse 项目的 Maven 依赖项，以确保您使用正确的 Fuse 版本。

通常，您可以使用 Maven 构建 Fuse 应用程序。Maven 是 Apache 的免费开源构建工具。Maven 配置在 Fuse 应用的 `pom.xml` 文件中定义。在构建 Fuse 项目时，默认行为是 Maven 搜索外部存储库并下载所需的工件。您可以将 Fuse Bill of Materials (BOM) 的依赖项添加到 `pom.xml` 文件中，以便 Maven 构建过程获取正确的 Fuse 支持的工件集合。

以下小节提供了有关 Maven 依赖项的信息，以及如何在 Fuse 项目中更新它们。

- [第 5.2 节 “关于 Maven 依赖项”](#)
- [第 5.3 节 “更新 Fuse 项目的 Maven 依赖项”](#)

5.1. CAMEL 迁移注意事项

使用 `MongoClients` 工厂创建到 MongoDB 的连接

在 Fuse 7.12 中，使用 `com.mongodb.client.MongoClient` 而不是 `com.mongodb.MongoClient` 创建与 MongoDB 的连接（请注意完整路径中的额外 `.client` 子软件包）。

如果任何现有 Fuse 应用程序使用 `camel-mongodb` 组件，您必须：

- 更新您的应用，以创建连接 bean 作为 `com.mongodb.client.MongoClient` 实例。例如，创建一个到 MongoDB 的连接，如下所示：

```
import com.mongodb.client.MongoClient;
```

然后您可以创建 `MongoClient` bean，如下例所示：

```
return MongoClient.create("mongodb://admin:password@192.168.99.102:32553");
```

- 如果需要，请重构与 `MongoClient` 类公开的方法相关的任何代码。

Camel 2.23

红帽 Fuse 使用 Apache Camel 2.23。升级到 Fuse 7.8 时，您应该考虑对 Camel 2.22 和 2.23 的以下更新。

Camel 2.22 更新

- Camel 从 Spring Boot v1 升级到 v2，因此不再支持 v1。
- 升级到 Spring Framework 5。Camel 应该与 Spring 4.3.x 一起工作，但在未来的 Spring 5.x 将会是最低的 Spring 版本。
- 升级到 Karaf 4.2。您可以在 Karaf 4.1 上运行 Camel，但我们在此版本中仅正式支持 Karaf 4.2。
- 使用 toD DSL 优化，以将端点和制作者重新用于组件。例如，基于 HTTP 的组件现在将利用动态 URI 发送到同一主机的生成者 (HTTP 客户端)。

- 现在，可以配置具有 read-lock idempotent/idempotent-changed 的 File2 使用者，以延迟发行任务，以便在文件被视为进程中时扩展窗口，这可用于具有共享的幂等存储库的 active/active 集群设置，以确保其他节点没有太快地将已处理的文件作为文件（如果您有 readLockRemoveOnCommit=true 时才需要）。
- 允许以 request/reply 模式在 Netty4 producer 上插件自定义请求/回复 ID 管理器实现。Twitter 组件现在默认使用扩展模式来支持大于 140 个字符的 tweets
- REST DSL 生成者现在支持使用 endpointProperties 在 REST 配置中配置。
- Kafka 组件现在支持 HeaderFilterStrategy 来插件自定义实现，以控制 Camel 和 Kafka 信息之间的标头映射。
- REST DSL 现在支持客户端请求验证 REST 服务是否可以验证 Content-Type/Accept 标头。
- Camel 现在有一个 Service Registry SPI，它允许您使用 Camel 实现或 Spring Cloud 将路由注册到服务 registry（如 consul、etcd 或 zookeeper）。
- SEDA 组件现在的默认队列大小为 1000，而不是无限。
- 以下值得注意的问题已被修复：
 - 修复了带有 camel-cxf 消费者的 CXF 持续超时问题，可能会导致消费者返回带有数据的响应，而不是触发调用 SOAP 客户端的超时。
 - 修复了 camel-cxf 消费者在使用强大的单向操作时不会发布 UoW。
 - 修复了使用 AdviceWith 并使用 weave 方法 onException 等问题。
 - 修复了并行处理和流模式中的 Splitter 可能会阻止，当迭代消息正文时，第一次调用 next() 方法调用时，迭代消息正文。
 - 修复了 Kafka 消费者，如果 autoCommitEnable=false，则不会自动提交。
 - 修复了文件消费者默认使用 markerFile 作为 read-lock，它应该没有。
 - 修复了使用带有 Kafka 的手动提交的问题，以提供当前的记录偏移，而不是之前（第一个为 -1）。
 - 修复了 Java DSL 中基于内容的路由路由器，可能无法解析 predicates 中的属性占位符。

Camel 2.23 更新

- 升级到 Spring Boot 2.1。
- 现在，可以使用 spring-boot 自动配置来配置额外的组件级别选项。这些选项包含在 spring-boot 组件元数据 JSON 文件描述符中，以获得工具协助。
- 添加了一个文档部分，其中包含所有组件、数据格式和语言的所有 Spring Boot auto 配置选项。
- 所有 Camel Spring Boot starter JAR 现在在其 JAR 中包含 META-INF/spring-autoconfigure-metadata.properties 文件，以优化 Spring Boot 自动配置。
- Throttler 现在支持基于动态表达式的关联组，以便您可以将消息分组到不同的节流集合中。
- Hystrix EIP 现在允许继承 Camel 的错误处理程序，以便在启用了带有 redeliveries 的错误处理时再次重试整个 Hystrix EIP 块。

- SQL 和 EISql 用户现在支持路由表单中的动态查询参数。请注意，这个功能仅限于使用简单表达式调用 Bean。
- swagger-restdsl maven 插件现在支持从 Swagger 规格文件生成 DTO 模型类。
- 以下值得注意的问题已被修复：
 - 修正了 Aggregator2，不会传播控制标头以强制完成所有组，因此如果其他聚合器 EIP 在路由过程中正在使用另一个聚合器 EIP，则不会再次发生。
 - 修复了在错误处理程序中重新传送被激活时的 Tracer 无法正常工作。
 - XML 文档的内置类型转换器可能会将错误输出到 stdout，现在它已被修复为使用日志记录 API 的输出。
 - 修复了当消息正文基于流处理时，使用 charset 选项编写文件的 SFTP 无法正常工作。
 - 修复了当在多个路由路由时不会重复使用的 Zipkin root id，将它们分组到一个父 span 中。
 - 修复了在主机名包含带有数字的 IP 地址时，使用 HTTP 端点进行优化的问题。
 - 修复了在临时队列中带有请求/回复的 RabbitMQ 的问题，并使用手动确认模式。它不会确认临时队列（需要进行请求/回复）。
 - 修复了各种 HTTP 消费者组件，它们可能无法在 Allow header for OPTIONS 请求（比如使用 rest-dsl 时）返回所有允许的 HTTP 动词。
 - 修复了 FluentProducerTemplate 的 thread-safety 问题。

5.2. 关于 MAVEN 依赖项

Maven Bill of Materials (BOM) 文件的目的是提供一组策展的 Maven 依赖项版本，这些版本可以很好地协同工作，从而为您为每个 Maven 工件单独定义版本。

每个容器都有一个专用的 BOM 文件，在其中运行 Fuse。

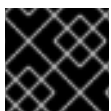


注意

您可以在此处找到这些 BOM 文件：<https://github.com/jboss-fuse/redhat-fuse>。或者，访问有关 BOM 文件更新的信息 [的最新发行注记](#)。

Fuse BOM 提供以下优点：

- 定义 Maven 依赖项的版本，以便在将依赖项添加到 pom.xml 文件时不需要指定版本。
- 定义一组对特定版本的 Fuse 经过全面测试并支持的策展依赖关系。
- 简化 Fuse 的升级。



重要

红帽仅支持由 Fuse BOM 定义的一组依赖项。

5.3. 更新 FUSE 项目的 MAVEN 依赖项

要升级 Spring Boot 的 Fuse 应用程序，请更新项目的 Maven 依赖项。

流程

1. 打开项目的 `pom.xml` 文件。
2. 在项目的 `pom.xml` 文件中添加 `dependenciesManagement` 元素（或者，可能在父 `pom.xml` 文件中），如下例所示：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.13.0.fuse-7_13_0-00012-redhat-00001</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-springboot-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```



注意

确保也更新您的 Spring Boot 版本。这通常位于 `pom.xml` 文件中的 Fuse 版本下：

```
<properties>
  <!-- configure the versions you want to use here -->
  <fuse.version>7.13.0.fuse-7_13_0-00012-redhat-00001</fuse.version>
  <spring-boot.version>2.7.18</spring-boot.version>
</properties>
```

3. 保存 `pom.xml` 文件。

在将 BOM 指定为 `pom.xml` 文件中的依赖项后，可以在没有指定工件版本的情况下将 Maven 依赖项添加到 `pom.xml` 文件中。例如，要为 `camel-velocity` 组件添加依赖项，您可以在 `pom.xml` 文件中的 `dependencies` 元素中添加以下 XML 片段：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
```

```
<scope>provided</scope>  
</dependency>
```

注意如何在这个依赖项定义中省略 **version** 元素。

第 6 章 在 JBOSS EAP 独立上升级 FUSE 应用程序

在 JBoss EAP 上升级 Fuse 应用程序：

- 您应该考虑 Apache Camel 更新，如 [第 6.1 节 “Camel 迁移注意事项”](#) 所述。
- 您必须更新 Fuse 项目的 Maven 依赖项，以确保您使用正确的 Fuse 版本。
通常，您可以使用 Maven 构建 Fuse 应用程序。Maven 是 Apache 的免费开源构建工具。Maven 配置在 Fuse 应用的 `pom.xml` 文件中定义。在构建 Fuse 项目时，默认行为是 Maven 搜索外部存储库并下载所需的工件。您可以将 Fuse Bill of Materials (BOM) 的依赖项添加到 `pom.xml` 文件中，以便 Maven 构建过程获取正确的 Fuse 支持的工件集合。

以下小节提供了有关 Maven 依赖项的信息，以及如何在 Fuse 项目中更新它们。

- [第 6.2 节 “关于 Maven 依赖项”](#)
- [第 6.3 节 “更新 Fuse 项目的 Maven 依赖项”](#)
- 您必须更新 Fuse 项目的 Maven 依赖项，以确保您使用 Java EE 依赖项的升级版本，如 [第 6.4 节 “升级 Java EE 依赖项”](#) 所述。

6.1. CAMEL 迁移注意事项

使用 `MongoClients` 工厂创建到 MongoDB 的连接

在 Fuse 7.12 中，使用 `com.mongodb.client.MongoClient` 而不是 `com.mongodb.MongoClient` 创建与 MongoDB 的连接（请注意完整路径中的额外 `.client` 子软件包）。

如果任何现有 Fuse 应用程序使用 `camel-mongodb` 组件，您必须：

- 更新您的应用，以创建连接 bean 作为 `com.mongodb.client.MongoClient` 实例。
例如，创建一个到 MongoDB 的连接，如下所示：

```
import com.mongodb.client.MongoClient;
```

然后您可以创建 `MongoClient` bean，如下例所示：

```
return MongoClient.create("mongodb://admin:password@192.168.99.102:32553");
```

- 如果需要，请重构与 `MongoClient` 类公开的方法相关的任何代码。

Camel 2.23

红帽 Fuse 使用 Apache Camel 2.23。升级到 Fuse 7.8 时，您应该考虑对 Camel 2.22 和 2.23 的以下更新。

Camel 2.22 更新

- Camel 从 Spring Boot v1 升级到 v2，因此不再支持 v1。
- 升级到 Spring Framework 5。Camel 应该与 Spring 4.3.x 一起工作，但在未来的 Spring 5.x 将会是最低的 Spring 版本。
- 升级到 Karaf 4.2。您可以在 Karaf 4.1 上运行 Camel，但我们在此版本中仅正式支持 Karaf 4.2。

- 使用 toD DSL 优化，以将端点和制作者重新用于组件。例如，基于 HTTP 的组件现在将利用动态 URI 发送到同一主机的生成者(HTTP 客户端)。
- 现在，可以配置具有 read-lock idempotent/idempotent-changed 的 File2 使用者，以延迟发行任务，以便在文件被视为进程中时扩展窗口，这可用于具有共享的幂等存储库的 active/active 集群设置，以确保其他节点没有太快地将已处理的文件作为文件（如果您有 readLockRemoveOnCommit=true 时才需要）。
- 允许以 request/reply 模式在 Netty4 producer 上插件自定义请求/回复 ID 管理器实现。Twitter 组件现在默认使用扩展模式来支持大于 140 个字符的 tweets
- REST DSL 生成者现在支持使用 endpointProperties 在 REST 配置中配置。
- Kafka 组件现在支持 HeaderFilterStrategy 来插件自定义实现，以控制 Camel 和 Kafka 信息之间的标头映射。
- REST DSL 现在支持客户端请求验证 REST 服务是否可以验证 Content-Type/Accept 标头。
- Camel 现在有一个 Service Registry SPI，它允许您使用 Camel 实现或 Spring Cloud 将路由注册到服务 registry（如 consul、etcd 或 zookeeper）。
- SEDA 组件现在的默认队列大小为 1000，而不是无限。
- 以下值得注意的问题已被修复：
 - 修复了带有 camel-cxf 消费者的 CXF 持续超时问题，可能会导致消费者返回带有数据的响应，而不是触发调用 SOAP 客户端的超时。
 - 修复了 camel-cxf 消费者在使用强大的单向操作时不会发布 UoW。
 - 修复了使用 AdviceWith 并使用 weave 方法 onException 等问题。
 - 修复了并行处理和流模式中的 Splitter 可能会阻止，当迭代消息正文时，第一次调用 next () 方法调用时，迭代消息正文。
 - 修复了 Kafka 消费者，如果 autoCommitEnable=false，则不会自动提交。
 - 修复了文件消费者默认使用 markerFile 作为 read-lock，它应该没有。
 - 修复了使用带有 Kafka 的手动提交的问题，以提供当前的记录偏移，而不是之前（第一个为 -1）。
 - 修复了 Java DSL 中基于内容的路由路由器，可能无法解析 predicates 中的属性占位符。

Camel 2.23 更新

- 升级到 Spring Boot 2.1。
- 现在，可以使用 spring-boot 自动配置来配置额外的组件级别选项。这些选项包含在 spring-boot 组件元数据 JSON 文件描述符中，以获得工具协助。
- 添加了一个文档部分，其中包含所有组件、数据格式和语言的所有 Spring Boot auto 配置选项。
- 所有 Camel Spring Boot starter JAR 现在在其 JAR 中包含 META-INF/spring-autoconfigure-metadata.properties 文件，以优化 Spring Boot 自动配置。
- Throttler 现在支持基于动态表达式的关联组，以便您可以将消息分组到不同的节流集合中。

- Hystrix EIP 现在允许继承 Camel 的错误处理程序，以便在启用了带有 redeliveries 的错误处理时再次重试整个 Hystrix EIP 块。
- SQL 和 EISql 用户现在支持路由表单中的动态查询参数。请注意，这个功能仅限于使用简单表达式调用 Bean。
- swagger-restdsl maven 插件现在支持从 Swagger 规格文件生成 DTO 模型类。
- 以下值得注意的问题已被修复：
 - 修正了 Aggregator2，不会传播控制标头以强制完成所有组，因此如果其他聚合器 EIP 在路由过程中正在使用另一个聚合器 EIP，则不会再次发生。
 - 修复了在错误处理程序中重新传送被激活时的 Tracer 无法正常工作。
 - XML 文档的内置类型转换器可能会将错误输出到 stdout，现在它已被修复为使用日志记录 API 的输出。
 - 修复了当消息正文基于流处理时，使用 charset 选项编写文件的 SFTP 无法正常工作。
 - 修复了当在多个路由路由时不会重复使用的 Zipkin root id，将它们分组到一个父 span 中。
 - 修复了在主机名包含带有数字的 IP 地址时，使用 HTTP 端点进行优化的问题。
 - 修复了在临时队列中带有请求/回复的 RabbitMQ 的问题，并使用手动确认模式。它不会确认临时队列（需要进行请求/回复）。
 - 修复了各种 HTTP 消费者组件，它们可能无法在 Allow header for OPTIONS 请求（比如使用 rest-dsl 时）返回所有允许的 HTTP 动词。
 - 修复了 FluentProducerTemplate 的 thread-safety 问题。

6.2. 关于 MAVEN 依赖项

Maven Bill of Materials (BOM) 文件的目的是提供一组策展的 Maven 依赖项版本，这些版本可以很好地协同工作，从而为您为每个 Maven 工件单独定义版本。

每个容器都有一个专用的 BOM 文件，在其中运行 Fuse。



注意

您可以在此处找到这些 BOM 文件：<https://github.com/jboss-fuse/redhat-fuse>。或者，访问有关 BOM 文件更新的信息 [的最新发行注记](#)。

Fuse BOM 提供以下优点：

- 定义 Maven 依赖项的版本，以便在将依赖项添加到 pom.xml 文件时不需要指定版本。
- 定义一组对特定版本的 Fuse 经过全面测试并支持的策展依赖关系。
- 简化 Fuse 的升级。



重要

红帽仅支持由 Fuse BOM 定义的一组依赖项。

6.3. 更新 FUSE 项目的 MAVEN 依赖项

若要升级用于 JBoss EAP 的 Fuse 应用，请更新项目的 Maven 依赖项。

流程

1. 打开项目的 `pom.xml` 文件。
2. 在项目的 `pom.xml` 文件中添加 `dependenciesManagement` 元素（或者，可能在父 `pom.xml` 文件中），如下例所示：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.13.0.fuse-7_13_0-00012-redhat-00001</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-eap-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```

3. 保存 `pom.xml` 文件。

在将 BOM 指定为 `pom.xml` 文件中的依赖项后，可以在没有指定工件版本的情况下将 Maven 依赖项添加到 `pom.xml` 文件中。例如，要为 `camel-velocity` 组件添加依赖项，您可以在 `pom.xml` 文件中的 `dependencies` 元素中添加以下 XML 片段：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

注意如何在这个依赖项定义中省略 `version` 元素。

6.4. 升级 JAVA EE 依赖项

在 Fuse 7.8 中，BOM 文件中的一些受管依赖项具有更新的 `groupId` 或 `artifactId` 属性，因此您必须相应地更新项目的 `pom.xml` 文件。

流程

1. 打开项目的 `pom.xml` 文件。
2. 要将 `org.jboss.spec.javaax.transaction` 版本从 1.2 改为 1.3，将 `org.jboss.spec.javaax.servlet` 版本从 3.1 改为 4.0，如下例所示：

```
<dependency>
  <groupId>org.jboss.spec.javaax.transaction</groupId>
  <artifactId>jboss-transaction-api_1.3_spec</artifactId>
</dependency>

<dependency>
  <groupId>org.jboss.spec.javaax.servlet</groupId>
  <artifactId>jboss-servlet-api_4.0_spec</artifactId>
</dependency>
```

3. 要从 Java EE API 迁移到 Jakarta EE，请为每个 `groupId` 替换 `javax` TOTP 替换为 `jakarta` prerequisites，并修改各个依赖项的 `artifactId`，如下例所示：

```
<dependency>
  <groupId>jakarta.validation</groupId>
  <artifactId>jakarta.validation-api</artifactId>
</dependency>

<dependency>
  <groupId>jakarta.enterprise</groupId>
  <artifactId>jakarta.enterprise.cdi-api</artifactId>
</dependency>

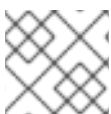
<dependency>
  <groupId>jakarta.inject</groupId>
  <artifactId>jakarta.inject-api</artifactId>
</dependency>
```

6.5. 在 JBOSS EAP 安装中升级现有 FUSE

以下流程描述了如何在 JBoss EAP 安装中升级现有 Fuse。

流程

1. 要从一个 JBoss EAP 次版本升级到另一个版本，您应该按照 [JBoss EAP 补丁和升级指南](#) 中的说明进行操作。
2. 要更新 Fuse，您必须在 JBoss EAP 安装程序上运行 Fuse，如在 [JBoss EAP 上安装](#) 指南所述。



注意

您不需要重新编译或重做 Fuse 应用程序。

6.6. 同时升级 FUSE 和 JBOSS EAP

以下流程描述了如何将 Fuse 安装和 JBoss EAP 运行时升级，例如，如果您是从 JBoss EAP 7.2 上的 Fuse 7.7 迁移到 JBoss EAP 7.3 上的 Fuse 7.8。



警告

在升级 Fuse 和 JBoss EAP 运行时，红帽建议您对 Fuse 和 JBoss EAP 运行时执行全新安装。

流程

1. 要执行 JBoss EAP 运行时的新安装，请按照在 [JBoss EAP 上安装](#) 指南中的说明进行操作。
2. 要执行 Fuse 的新安装，请在 JBoss EAP 安装程序上运行 Fuse，如在 [JBoss EAP 指南中的安装](#) 所述。

第 7 章 在 KARAF 独立升级 FUSE 应用程序

要在 Karaf 上升级 Fuse 应用程序：

- 您应该考虑 Apache Camel 更新，如 [第 7.1 节 “Camel 迁移注意事项”](#) 所述。
- 您必须更新 Fuse 项目的 Maven 依赖项，以确保您使用正确的 Fuse 版本。

通常，您可以使用 Maven 构建 Fuse 应用程序。Maven 是 Apache 的免费开源构建工具。Maven 配置在 Fuse 应用的 `pom.xml` 文件中定义。在构建 Fuse 项目时，默认行为是 Maven 搜索外部存储库并下载所需的工件。您可以将 Fuse Bill of Materials (BOM) 的依赖项添加到 `pom.xml` 文件中，以便 Maven 构建过程获取正确的 Fuse 支持的工件集合。

以下小节提供了有关 Maven 依赖项的信息，以及如何在 Fuse 项目中更新它们。

- [第 7.2 节 “关于 Maven 依赖项”](#)
- [第 7.3 节 “更新 Fuse 项目的 Maven 依赖项”](#)

7.1. CAMEL 迁移注意事项

使用 `MongoClients` 工厂创建到 MongoDB 的连接

在 Fuse 7.12 中，使用 `com.mongodb.client.MongoClient` 而不是 `com.mongodb.MongoClient` 创建与 MongoDB 的连接（请注意完整路径中的额外 `.client` 子软件包）。

如果任何现有 Fuse 应用程序使用 `camel-mongodb` 组件，您必须：

- 更新您的应用，以创建连接 bean 作为 `com.mongodb.client.MongoClient` 实例。
例如，创建一个到 MongoDB 的连接，如下所示：

```
import com.mongodb.client.MongoClient;
```

然后您可以创建 `MongoClient` bean，如下例所示：

```
return MongoClient.create("mongodb://admin:password@192.168.99.102:32553");
```

- 如果需要，请重构与 `MongoClient` 类公开的方法相关的任何代码。

Camel 2.23

红帽 Fuse 使用 Apache Camel 2.23。升级到 Fuse 7.8 时，您应该考虑对 Camel 2.22 和 2.23 的以下更新。

Camel 2.22 更新

- Camel 从 Spring Boot v1 升级到 v2，因此不再支持 v1。
- 升级到 Spring Framework 5。Camel 应该与 Spring 4.3.x 一起工作，但在未来的 Spring 5.x 将会是最低的 Spring 版本。
- 升级到 Karaf 4.2。您可以在 Karaf 4.1 上运行 Camel，但我们在此版本中仅正式支持 Karaf 4.2。
- 使用 toD DSL 优化，以将端点和制作者重新用于组件。例如，基于 HTTP 的组件现在将利用动态 URI 发送到同一主机的生成者 (HTTP 客户端)。

- 现在，可以配置具有 read-lock idempotent/idempotent-changed 的 File2 使用者，以延迟发行任务，以便在文件被视为进程中时扩展窗口，这可用于具有共享的幂等存储库的 active/active 集群设置，以确保其他节点没有太快地将已处理的文件作为文件（如果您有 readLockRemoveOnCommit=true 时才需要）。
- 允许以 request/reply 模式在 Netty4 producer 上插件自定义请求/回复 ID 管理器实现。Twitter 组件现在默认使用扩展模式来支持大于 140 个字符的 tweets
- REST DSL 生成者现在支持使用 endpointProperties 在 REST 配置中配置。
- Kafka 组件现在支持 HeaderFilterStrategy 来插件自定义实现，以控制 Camel 和 Kafka 信息之间的标头映射。
- REST DSL 现在支持客户端请求验证 REST 服务是否可以验证 Content-Type/Accept 标头。
- Camel 现在有一个 Service Registry SPI，它允许您使用 Camel 实现或 Spring Cloud 将路由注册到服务 registry（如 consul、etcd 或 zookeeper）。
- SEDA 组件现在的默认队列大小为 1000，而不是无限。
- 以下值得注意的问题已被修复：
 - 修复了带有 camel-cxf 消费者的 CXF 持续超时问题，可能会导致消费者返回带有数据的响应，而不是触发调用 SOAP 客户端的超时。
 - 修复了 camel-cxf 消费者在使用强大的单向操作时不会发布 UoW。
 - 修复了使用 AdviceWith 并使用 weave 方法 onException 等问题。
 - 修复了并行处理和流模式中的 Splitter 可能会阻止，当迭代消息正文时，第一次调用 next() 方法调用时，迭代消息正文。
 - 修复了 Kafka 消费者，如果 autoCommitEnable=false，则不会自动提交。
 - 修复了文件消费者默认使用 markerFile 作为 read-lock，它应该没有。
 - 修复了使用带有 Kafka 的手动提交的问题，以提供当前的记录偏移，而不是之前（第一个为 -1）。
 - 修复了 Java DSL 中基于内容的路由路由器，可能无法解析 predicates 中的属性占位符。

Camel 2.23 更新

- 升级到 Spring Boot 2.1。
- 现在，可以使用 spring-boot 自动配置来配置额外的组件级别选项。这些选项包含在 spring-boot 组件元数据 JSON 文件描述符中，以获得工具协助。
- 添加了一个文档部分，其中包含所有组件、数据格式和语言的所有 Spring Boot auto 配置选项。
- 所有 Camel Spring Boot starter JAR 现在在其 JAR 中包含 META-INF/spring-autoconfigure-metadata.properties 文件，以优化 Spring Boot 自动配置。
- Throttler 现在支持基于动态表达式的关联组，以便您可以将消息分组到不同的节流集合中。
- Hystrix EIP 现在允许继承 Camel 的错误处理程序，以便在启用了带有 redeliveries 的错误处理时再次重试整个 Hystrix EIP 块。

- SQL 和 EISql 用户现在支持路由表单中的动态查询参数。请注意，这个功能仅限于使用简单表达式调用 Bean。
- swagger-restdsl maven 插件现在支持从 Swagger 规格文件生成 DTO 模型类。
- 以下值得注意的问题已被修复：
 - 修正了 Aggregator2，不会传播控制标头以强制完成所有组，因此如果其他聚合器 EIP 在路由过程中正在使用另一个聚合器 EIP，则不会再次发生。
 - 修复了在错误处理程序中重新传送被激活时的 Tracer 无法正常工作。
 - XML 文档的内置类型转换器可能会将错误输出到 stdout，现在它已被修复为使用日志记录 API 的输出。
 - 修复了当消息正文基于流处理时，使用 charset 选项编写文件的 SFTP 无法正常工作。
 - 修复了当在多个路由路由时不会重复使用的 Zipkin root id，将它们分组到一个父 span 中。
 - 修复了在主机名包含带有数字的 IP 地址时，使用 HTTP 端点进行优化的问题。
 - 修复了在临时队列中带有请求/回复的 RabbitMQ 的问题，并使用手动确认模式。它不会确认临时队列（需要进行请求/回复）。
 - 修复了各种 HTTP 消费者组件，它们可能无法在 Allow header for OPTIONS 请求（比如使用 rest-dsl 时）返回所有允许的 HTTP 动词。
 - 修复了 FluentProducerTemplate 的 thread-safety 问题。

7.2. 关于 MAVEN 依赖项

Maven Bill of Materials (BOM) 文件的目的是提供一组策展的 Maven 依赖项版本，这些版本可以很好地协同工作，从而为您为每个 Maven 工件单独定义版本。

每个容器都有一个专用的 BOM 文件，在其中运行 Fuse。

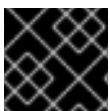


注意

您可以在此处找到这些 BOM 文件：<https://github.com/jboss-fuse/redhat-fuse>。或者，访问有关 BOM 文件更新的信息 [最新发行注记](#)。

Fuse BOM 提供以下优点：

- 定义 Maven 依赖项的版本，以便在将依赖项添加到 pom.xml 文件时不需要指定版本。
- 定义一组对特定版本的 Fuse 经过全面测试并支持的策展依赖关系。
- 简化 Fuse 的升级。



重要

红帽仅支持由 Fuse BOM 定义的一组依赖项。

7.3. 更新 FUSE 项目的 MAVEN 依赖项

若要升级用于 Karaf 的 Fuse 应用，请更新项目的 Maven 依赖项。

流程

1. 打开项目的 `pom.xml` 文件。
2. 在项目的 `pom.xml` 文件中添加 `dependenciesManagement` 元素（或者，可能在父 `pom.xml` 文件中），如下例所示：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.13.0.fuse-7_13_0-00012-redhat-00001</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-karaf-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```

3. 保存 `pom.xml` 文件。

在将 BOM 指定为 `pom.xml` 文件中的依赖项后，可以在没有指定工件版本的情况下将 Maven 依赖项添加到 `pom.xml` 文件中。例如，要为 `camel-velocity` 组件添加依赖项，您可以在 `pom.xml` 文件中的 `dependencies` 元素中添加以下 XML 片段：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

注意如何在这个依赖项定义中省略 `version` 元素。

第 8 章 在 KARAF 上升级 FUSE STANDALONE

通过 Apache Karaf 升级机制上的 Fuse，您可以将修复应用到 Apache Karaf 容器，而无需在 Karaf 上重新安装 Fuse 的更新版本。如果升级导致了与部署的应用程序相关的问题，它还允许您回滚升级。

升级安装程序文件是 *用来在 Apache Karaf 上安装 Fuse 的文件*。



注意

要获取升级安装程序文件，请进入红帽客户门户的 Downloads 页面，并下载 Apache Karaf 上 Fuse 的最新版本的安装存档（例如 `fuse-karaf-7.13.0.fuse-7_13_0-00012-redhat-00001.zip`）。

- [第 8.1 节 “将 Fuse 升级到 Karaf 的影响”](#)
- [第 8.2 节 “在 Karaf 上升级 Fuse Standalone”](#)
- [第 8.3 节 “在 Karaf 上回滚 Fuse 的升级”](#)

8.1. 将 FUSE 升级到 KARAF 的影响

升级机制可以对任何安装文件进行更新，包括捆绑包 JAR 和静态文件（如 `etc/` 目录下的配置文件）。Apache Karaf 升级过程上的 Fuse：

- 更新任何文件，包括捆绑 JAR、配置文件和任何静态文件。
- 对当前容器实例（及其在 `data/` 目录下的运行时存储）和底层安装进行补丁。因此，在删除容器实例后会保留补丁。
- 更新与 Karaf 功能相关的所有文件，包括功能存储库文件和功能本身。因此，推出补丁后安装的任何功能都将引用正确的补丁依赖项。
- 如有必要，更新配置文件（例如 `etc/` 下的文件），自动合并您已与补丁所做的配置更改所做的任何配置更改。如果发生合并冲突，请参阅补丁日志来了解如何处理它们的详细信息。
- 大多数合并冲突会自动解决。例如，补丁机制检测到属性文件的属性级别的冲突。它检测它是更改任何属性的用户或补丁。如果只更改属性，则更改会被保留。
- 跟踪对安装进行的所有更改（包括静态文件），以便可以回滚补丁。



注意

回滚补丁机制使用内部 git 存储库（位于 `patches/.management/history` 下）来跟踪所做的更改。

8.2. 在 KARAF 上升级 FUSE STANDALONE

以下说明指导您在 Apache Karaf 上升级 Fuse。在开始升级过程前，请确保所有先决条件都已完成。

先决条件

- 在升级前，确保您已在 Apache Karaf 安装上进行 Fuse 的完整备份。
- 启动容器（如果尚未运行）。

提示

如果容器在后台运行（或远程运行），请使用 SSH 控制台客户端 `bin/client` 连接到容器。

- 通过调用 `patch:add` 命令将升级安装程序文件添加到容器环境中。例如，要添加 `fuse-karaf-7.13.0.fuse-7_13_0-00012-redhat-00001.zip` 升级安装程序文件：

```
patch:add file:///path/to/fuse-karaf-7.13.0.fuse-7_13_0-00012-redhat-00001.zip
```



注意

`patch:find` 命令只能用于查找和添加容器环境的最新热修复补丁；它不能被应用完整的升级补丁。

流程

- 运行 `patch:update` 命令。不需要重启容器。

```
karaf@root(>) patch:update
Current patch mechanism version: 7.1.0.fuse-710023-redhat-00001
New patch mechanism version detected: 7.2.0.fuse-720035-redhat-00001
Uninstalling patch features in version 7.1.0.fuse-710023-redhat-00001
Installing patch features in version 7.2.0.fuse-720035-redhat-00001
```

- 调用 `patch:list` 命令来显示升级安装程序列表。在此列表中，`[name]` 标题下的条目是升级 ID。例如：

```
karaf@root(>) patch:list
[name] [installed] [rollup] [description]
fuse-karaf-7.2.0.fuse-720035-redhat-00001 false true fuse-karaf-7.2.0.fuse-720035-redhat-00001
```

- 通过调用 `patch:simulate` 命令并为您要应用的升级指定升级 ID 来模拟升级，如下所示：

```
karaf@root(>) patch:simulate fuse-karaf-7.2.0.fuse-720035-redhat-00001
INFO : org.jboss.fuse.modules.patch.patch-management (226): Installing rollup patch "fuse-karaf-7.2.0.fuse-720035-redhat-00001"
===== Repositories to remove (9):
- mvn:io.hawt/hawtio-karaf/2.0.0.fuse-710018-redhat-00002/xml/features
...
===== Repositories to add (9):
- mvn:io.hawt/hawtio-karaf/2.0.0.fuse-720044-redhat-00001/xml/features
...
===== Repositories to keep (10):
- mvn:org.apache.activemq/artemis-features/2.4.0.amq-711002-redhat-1/xml/features
...
===== Features to update (100):
[name] [version] [new version]
aries-blueprint 4.2.0.fuse-710024-redhat-00002 4.2.0.fuse-720061-redhat-00001
...
===== Bundles to update as part of features or core bundles (100):
[symbolic name] [version] [new location]
io.hawt.hawtio-log 2.0.0.fuse-710018-redhat-00002
```

```

mvn:io.hawt/hawtio-log/2.0.0.fuse-720044-redhat-00001
...
===== Bundles to reinstall as part of features or core bundles (123):
[symbolic name]                [version]                [location]
com.fasterxml.jackson.core.jackson-annotations      2.8.11
mvn:com.fasterxml.jackson.core/jackson-annotations/2.8.11
...
Simulation only - no files and runtime data will be modified.
karaf@root(>

```

这会生成执行升级时将对容器进行的更改日志，但不会对容器进行任何更改。检查模拟日志，以了解要对容器进行的更改。

4. 通过调用 **patch:install** 命令并为您要应用的升级指定升级 ID 来升级容器。例如：

```
karaf@root(> patch:install fuse-karaf-7.13.0.fuse-7_13_0-00012-redhat-00001
```

5. 通过搜索其中一个升级工件来验证升级。例如，如果您只将 Fuse 7.1.0 升级到 Fuse 7.2.0，您可以搜索构建号 7.2.0.fuse-720035-redhat-00001 的捆绑包，如下所示：

```

karaf@root(> bundle:list -l | grep 7.2.0.fuse-720035-redhat-00001
22 | Active | 80 | 7.2.0.fuse-720035-redhat-00001 |
mvn:org.jboss.fuse.modules/fuse-pax-transx-tm-narayana/7.2.0.fuse-720035-redhat-00001
188 | Active | 80 | 7.2.0.fuse-720035-redhat-00001 |
mvn:org.jboss.fuse.modules.patch/patch-commands/7.2.0.fuse-720035-redhat-00001

```



注意

升级后，在重启容器时，您还会在 Welcome banner 中看到新版本和构建号。

8.3. 在 KARAF 上回滚 FUSE 的升级

有时，升级可能无法正常工作，或者可能会给容器带来新的问题。在这些情况下，您可以使用 **patch:rollback** 命令轻松回滚升级并将系统恢复到之前的状态。这组指导您完成此流程。

先决条件

- 您最近在 Karaf 上升级了 Fuse。
- 您需要回滚升级。

流程

1. 调用 **patch:list** 命令，以获取最近安装的补丁的升级 ID **UPGRADE_ID**。
2. 调用 **patch:rollback** 命令，如下所示：

```
patch:rollback UPGRADE_ID
```



注意

在某些情况下，容器需要重启才能回滚升级。在这些情况下，容器会自动重启。由于 OSGi 运行时具有高度动态性质，在重启过程中，您可能会看到一些与不兼容类相关的严重错误。这些错误与刚才启动或停止且可以安全地忽略的 OSGi 服务相关。