Red Hat Fuse 7.11

迁移指南

迁移至 Red Hat Fuse 7.11

Last Updated: 2023-07-25

Red Hat Fuse 7.11 迁移指南

迁移至 Red Hat Fuse 7.11

法律通告

摘要

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

目录

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

如果要从 Fuse 6 迁移到最新的 Fuse 7 版本，请按照本指南中的说明进行操作，请按照 Red Hat Fuse 7.0 Migration Guide 中的说明操作。

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。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看我们的 CTO Chris Wright 信息。

第 1 章 在 OPENSHIFT 上升级 FUSE

Fuse 7.11 包含更新，它可用于 OpenShift Container Platform(OCP) 4.9 或更高版本。如果您计划升级到 OCP 4.10，则必须将 Fuse 升级到 7.11，**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 ONLINE 升级 FUSE

为 Fuse 发布补丁和安全修复的时间、全新应用程序镜像。通过红帽的勘误更新频道，通知您这些更新信息。然后您可以升级 Fuse 镜像。

对于 OCP 4.x，使用 OpenShift OperatorHub 来按照使用 OperatorHub(OCP 4.x)升级 Fuse 中的步骤，从 Fuse 7.10 升级到 7.11。

您应该决定升级到 Fuse 7.11 是否需要您对现有集成进行更改。即使不需要任何更改，在升级 Fuse 时都必须重新发布所有正在运行的集成。

使用 OperatorHub 升级 Fuse(OCP 4.x)

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，如果要升级到 Fuse 7.11，则必须升级到 OCP 4.6 或更高版本。

3. 在 OCP 4.9 上，当您升级到 7.11 时，在 Fuse Operator 升级过程中会显示以下警告：
   W1219 18:38:58.064578 1 warnings.go:70] extensions/v1beta1 Ingress 在 v1.14+ 中被弃用，在 v1.22+ 中不可用，使用 networking.k8s.io/v1 Ingress

   这个警告会出现的原因是客户端（ Fuse 用于 Kubernetes/OpenShift API 初始化代码）访问已弃用的 Ingress 版本。这个警告不是完全使用已弃用 API 的指示，没有升级到 Fuse 7.11 的问题。

从 Fuse Online 7.10 或早期版本升级到较新的 Fuse Online 7.11 版本的升级过程取决于您安装 Fuse Online 时选择的 **批准策略** ：

- 对于自动更新，当有新版本的 Fuse Operator 可用时，OpenShift Operator Lifecycle Manager(OLM)会自动升级运行 Fuse Online 的运行实例，而无需人为干预。

- **对于 手动更新，则当有新版 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，才能将 Fuse Online operator 更新至新版本，如 OpenShift 文档中的 手动批准待处理的 Operator 升级 部分中所述。**

在基础架构升级期间和之后，现有集成将继续使用较老版本的 Fuse 库和依赖项运行。

要使用更新的 Fuse 在线版本运行现有集成，您必须重新发布集成。

第 3 章 升级到 SPRING BOOT 2

本章论述了如何从 Spring Boot 1 将应用程序升级到 Spring Boot 2.0。

3.1. 开始前

在开始迁移到 Spring Boot 2 之前，您必须查看系统要求和依赖项。

- 升级到最新的 1.5.x 版本
  - 在开始之前，请先升级到最新的 1.5.x 可用版本。这是为了确保您根据该行的最新依赖项进行构建。
- 检查依赖项
  - 迁移到 Spring Boot 2 将会导致升级多个依赖项。查看 1.5.x 的依赖项管理，它带有 2.0.x 的依赖项管理，以评估您的项目如何受到影响。
  - 识别不是由 Spring Boot 管理的依赖关系的兼容版本，然后为它们定义显式版本。
- 查看自定义配置
  - 您的项目定义的任何自定义配置可能需要在升级过程中进行审核。如果可以使用标准自动配置替换，请在升级前进行操作。
- 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。
  - 不再支持 Java 6 和 7。

3.2. 从 SPRING BOOT 1 升级到 SPRING BOOT 2

查看项目状态和依赖项的状态后，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，很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** 模块实现这一点。在作为依赖项添加到项目后，这不仅会分析应用的环境并在启动时打印诊断，还会在运行时临时迁移属性。

流程

1. 将 **spring-boot-properties-migrator** 模块添加到项目的 **pom.xml** 的 dependency 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 OPENSHIFT MAVEN 插件

fabric8-maven-plugin 已从 Fuse 7.11 完全移除。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

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

1. **将应用中的 src/main/fabric8 目录重命名为 src/main/jkube。**
2. **找到项目的 pom.xml 中的 org.jboss.redhat-fuse:fabric8-maven-plugin 依赖项，并将它改为 org.jboss.redhat-fuse:openshift-maven-plugin。**请参阅 **示例 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中，可能不再需要。**
4. **您可以创建一些示例测试，它们可用于在迁移后反映 API 更改。如需更多信息，**请参阅 **Spring Boot Camel Quickstart 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 上升级您的 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，以确保您正在使用正确的 Fuse 版本。

通常，您使用 Maven 来构建 Fuse 应用程序。Maven 是一个来自 Apache 的免费开源构建工具。Maven 配置在 Fuse 应用项目的 **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，以便 Maven 构建过程获取正确的 Fuse 支持的工件集。

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 factory 创建到 MongoDB 的连接

从 Fuse 7.11，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：

- 更新应用程序，以将连接 bean 创建为 **com.mongodb.client.MongoClient** 实例。
  例如，创建与 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 MongoClients.create("mongodb://admin:password@192.168.99.102:32553");
  ```

- 评估并根据需要重构与 MongoClient 类公开的方法相关的代码。

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 已从 Spring Boot v1 升级到 v2，因此不再支持 v1。
- 升级到 Spring Framework 5。Camel 还应与 Spring 4.3.x 配合工作，但在将来的版本中将转发 Spring 5.x 的最低版本。
- 升级至 Karaf 4.2。您可以在 Karaf 4.1 上运行 Camel，但在此版本中，我们只正式支持 Karaf 4.2。
- 使用 toD DSL 优化为可能的组件重复使用端点和制作者。例如，基于 HTTP 的组件现在将重复使用带有发送到同一主机的动态 URI 的生产者（HTTP 客户端）。

- 现在，可将具有 read-lock idempotent/idempotent-changed 的 file2 使用者配置为延迟发行任务，以在文件视为进程中时（仅在有共享幂等库的主动/主动集群设置中使用）。
- 允许在请求/reply 模式的 Netty4 producer 上插件自定义请求/关系 id 管理器实施。Twitter 组件现在默认使用扩展模式来支持超过 140 个字符的调整
- REST DSL 制作者现在支持使用 endpointProperties 在 REST 配置中配置。
- Kafka 组件现在支持 HeaderFilterStrategy 来插件自定义实现来控制 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，以验证 REST 服务是否可以使用 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 camel-cxf consumer 的 CXF continuation 超时问题，这可能会导致使用者返回带有数据的响应，而不是触发调用 SOAP 客户端的超时问题。
  - 修复了 camel-cxf consumer在使用强健的单向操作时不会发布 UoW。
  - 修复了使用 AdviceWith 以及在Exception 上使用 usave 方法等问题。
  - 修复了并行处理和流传输模式中的 Splitter 可能会阻断，当迭代邮件正文时，迭代邮件正文在第一个调用的 next（）方法调用中引发异常。
  - 修复了 Kafka 用户，在 autoCommitEnable=false 时不自动提交。
  - 固定的文件消费者使用标记文件（默认情况下为 read-lock），这应该没有。
  - 修复了将手动提交与 Kafka 一起使用，以提供当前的记录偏移，而不是上一个记录（第一个使用 -1）。
  - 修复了 Java DSL 中基于内容的路由器可能无法解析 predicates 时 中的属性占位符。

Camel 2.23 更新

- 升级至 Spring Boot 2.1。
- 现在，可以使用 spring-boot 自动配置来配置其他组件级选项。这些选项包含在 spring-boot 组件元数据 JSON 文件描述符中，用于工具帮助。
- 添加了包含所有组件、数据格式和语言的 Spring Boot auto 配置选项的文档部分。
- 现在，所有 Camel Spring Boot starter JARs 都会在其 JARs 中包括 **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 throttled 集。
- Hystrix EIP 现在允许 Camel 的错误处理程序继承，如果已使用红色类别启用了错误处理，您可以再次重试整个 Hystrix EIP 块。

- SQL 和 ElSql 使用者现在支持路由形式的动态查询参数。请注意，此功能可通过使用简单表达式调用 Bean。
- swagger-restdsl maven 插件现在支持从 Swagger 规格文件中生成 DTO 模型类。
- 以下值得注意的问题已被修复：
  - 修正了聚合器2，不会传播控制标头以强制完成所有组，因此以后路由过程中使用了另一个聚合器 EIP 时，它不会再次发生。
  - 修复了当错误处理器中重新发送被激活时的 Tracer 无法正常工作。
  - XML 文档的内置类型转换器可能会输出将错误解析到 stdout，该错误现在使用日志记录 API 固定为输出。
  - 修复了在基于消息正文时，使用 charset 选项的 SFTP 写入文件将无法正常工作。
  - 修复了当在多个路由之间进行路由时，Zipkin root id 不会被重复使用，从而将它们分组到一个父范围中。
  - 修复了在使用 HTTP 端点时，当主机名包含带有数字的 IP 地址时，被优化为D。
  - 修复了 RabbitMQ 与请求/回复超过临时队列的 RabbitMQ 的问题，并使用手动确认模式。它不会确认临时队列（需要可以发出请求/恢复）。
  - 修复了各种 HTTP 消费者组件，它们可能无法在 Allow header 中为 OPTIONS 请求返回所有允许的 HTTP 动词（比如使用 rest-dsl）。
  - 修复了 FluentProducerTemplate 中的 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精心设计的 Maven 依赖项版本，从而为您保存每个 Maven 工件的版本，从而单独定义每个 Maven 工件的版本。

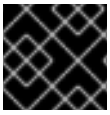
Fuse 的每个容器都有一个专用的 BOM 文件。

注意

您可以在此找到这些 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，这样您不需要在向 pom.xml 文件中添加依赖项时指定版本。
- 定义一组经过策展的依赖关系，这些依赖项是针对特定版本的 Fuse 完全测试和支持的。
- 简化 Fuse 升级.



重要

红帽只支持由 Fuse BOM 定义的一组依赖项。

5.3. 更新 FUSE 项目的 MAVEN 依赖项

要为 Spring Boot 升级 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 dependencyManagement 元素（或者在父 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>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</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

您还需要更新 Spring Boot 版本。这通常在 pom.xml 文件的 Fuse 版本下找到：

```
 <properties>
  <!-- configure the versions you want to use here -->
  <fuse.version>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</fuse.version>
  <spring-boot.version>2.5.13.RELEASE</spring-boot.version>
 </properties>
```

3. 保存 pom.xml 文件。

在将 BOM 指定为 pom.xml 文件中的依赖关系后，*可以在不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，若要为 camel-velocity 组件添加依赖项，您要将以下 XML 片段添加到 pom.xml 文件中的 dependencies 元素中：

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

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

注意 version 元素是如何从这个依赖关系定义中省略的。

第 6 章 在 JBOSS EAP 独立升级 FUSE 应用程序

在 JBoss EAP 上升级您的 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，以确保您正在使用正确的 Fuse 版本。
  通常，您使用 Maven 来构建 Fuse 应用程序。Maven 是一个来自 Apache 的免费开源构建工具。Maven 配置在 Fuse 应用项目的 **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，以便 Maven 构建过程获取正确的 Fuse 支持的工件集。

  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 factory 创建到 MongoDB 的连接

从 Fuse 7.11，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：

- 更新应用程序，以将连接 bean 创建为 **com.mongodb.client.MongoClient** 实例。
  例如，创建与 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 MongoClients.create("mongodb://admin:password@192.168.99.102:32553");
  ```
- 评估并根据需要重构与 MongoClient 类公开的方法相关的代码。

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 已从 Spring Boot v1 升级到 v2，因此不再支持 v1。
- 升级到 Spring Framework 5。Camel 还应与 Spring 4.3.x 配合工作，但在将来的版本中将转发 Spring 5.x 的最低版本。
- 升级至 Karaf 4.2。您可以在 Karaf 4.1 上运行 Camel，但在此版本中，我们只正式支持 Karaf 4.2。

- 使用 toD DSL 优化为可能的组件重复使用端点和制作者。例如，基于 HTTP 的组件现在将重复使用带有发送到同一主机的动态 URI 的生产者（HTTP 客户端）。
- 现在，可将具有 read-lock idempotent/idempotent-changed 的 file2 使用者配置为延迟发行任务，以在文件视为进程中时（仅在有共享幂等库的主动/主动集群设置中使用）。
- 允许在请求/reply 模式的 Netty4 producer 上插件自定义请求/关系 id 管理器实施。Twitter 组件现在默认使用扩展模式来支持超过 140 个字符的调整
- REST DSL 制作者现在支持使用 endpointProperties 在 REST 配置中配置。
- Kafka 组件现在支持 HeaderFilterStrategy 来插件自定义实现来控制 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，以验证 REST 服务是否可以使用 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 camel-cxf consumer 的 CXF continuation 超时问题，这可能会导致使用者返回带有数据的响应，而不是触发调用 SOAP 客户端的超时问题。
  - 修复了 camel-cxf consumer在使用强健的单向操作时不会发布 UoW。
  - 修复了使用 AdviceWith 以及在Exception 上使用 usave 方法等问题。
  - 修复了并行处理和流传输模式中的 Splitter 可能会阻断，当迭代邮件正文时，迭代邮件正文在第一个调用的 next（）方法调用中引发异常。
  - 修复了 Kafka 用户，在 autoCommitEnable=false 时不自动提交。
  - 固定的文件消费者使用标记文件（默认情况下为 read-lock），这应该没有。
  - 修复了将手动提交与 Kafka 一起使用，以提供当前的记录偏移，而不是上一个记录（第一个使用 -1）。
  - 修复了 Java DSL 中基于内容的路由器可能无法解析 predicates 时 中的属性占位符。

Camel 2.23 更新

- 升级至 Spring Boot 2.1。
- 现在，可以使用 spring-boot 自动配置来配置其他组件级选项。这些选项包含在 spring-boot 组件元数据 JSON 文件描述符中，用于工具帮助。
- 添加了包含所有组件、数据格式和语言的 Spring Boot auto 配置选项的文档部分。
- 现在，所有 Camel Spring Boot starter JARs 都会在其 JARs 中包括 **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 throttled 集。

- Hystrix EIP 现在允许 Camel 的错误处理程序继承，如果已使用红色类别启用了错误处理，您可以再次重试整个 Hystrix EIP 块。
- SQL 和 ElSql 使用者现在支持路由形式的动态查询参数。请注意，此功能可通过使用简单表达式调用 Bean。
- swagger-restdsl maven 插件现在支持从 Swagger 规格文件中生成 DTO 模型类。
- 以下值得注意的问题已被修复：
  - 修正了聚合器2，不会传播控制标头以强制完成所有组，因此以后路由过程中使用了另一个聚合器 EIP 时，它不会再次发生。
  - 修复了当错误处理器中重新发送被激活时的 Tracer 无法正常工作。
  - XML 文档的内置类型转换器可能会输出将错误解析到 stdout，该错误现在使用日志记录 API 固定为输出。
  - 修复了在基于消息正文时，使用 charset 选项的 SFTP 写入文件将无法正常工作。
  - 修复了当在多个路由之间进行路由时，Zipkin root id 不会被重复使用，从而将它们分组到一个父范围中。
  - 修复了在使用 HTTP 端点时，当主机名包含带有数字的 IP 地址时，被优化为D。
  - 修复了 RabbitMQ 与请求/回复超过临时队列的 RabbitMQ 的问题，并使用手动确认模式。它不会确认临时队列（需要可以发出请求/恢复）。
  - 修复了各种 HTTP 消费者组件，它们可能无法在 Allow header 中为 OPTIONS 请求返回所有允许的 HTTP 动词（比如使用 rest-dsl）。
  - 修复了 FluentProducerTemplate 中的 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精心设计的 Maven 依赖项版本，从而为您保存每个 Maven 工件的版本，从而单独定义每个 Maven 工件的版本。

Fuse 的每个容器都有一个专用的 BOM 文件。

注意

您可以在此找到这些 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，这样您不需要在向 pom.xml 文件中添加依赖项时指定版本。
- 定义一组经过策展的依赖关系，这些依赖项是针对特定版本的 Fuse 完全测试和支持的。
- 简化 Fuse 升级.

重要

红帽只支持由 Fuse BOM 定义的一组依赖项。

6.3. 更新 FUSE 项目的 MAVEN 依赖项

要升级 JBoss EAP 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 dependencyManagement 元素（或者在父 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>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</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 文件中的依赖关系后，*可以在不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，若要为 camel-velocity 组件添加依赖项，您要将以下 XML 片段添加到 pom.xml 文件中的 dependencies 元素中：

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

注意 version 元素是如何从这个依赖关系定义中省略的。

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已更新 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.javax.transaction 版本从 1.2 改为 1.3，并将org.jboss.spec.javax.servlet 版本从 3.1 改为 4.0，请将依赖项更新，如下例所示：

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

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

3. 要从 Java EE API 迁移到 Jakarta EE，请将 javax.* 替换为 jakarta.*，为每个组Id 修改 artifactId，如以下示例所示：

```
<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升级 JBoss EAP 安装中的现有 Fuse。

流程

1. 要从一个 JBoss EAP 次要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，必须在 JBoss EAP 安装程序中运行 Fuse，如 安装 JBoss EAP 指南中所述。

注意

您应该不需要重新编译或缩减 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，如果您从 JBoss EAP 7.2 上的 Fuse 7.7 迁移到 JBoss EAP 7.3 上的 Fuse 7.8）。

警告

升级 Fuse 和 JBoss EAP 运行时，红帽建议您对 Fuse 和 JBoss EAP 运行时进行全新的安装。

流程

1. 要执行 JBoss EAP 运行时的新安装，请按照 安装 JBoss EAP 指南中的说明操作。

2. 要执行 Fuse 的新安装，请在 JBoss EAP 安装程序中运行 Fuse，如 安装 JBoss EAP 指南中所述。

第 7 章 在 KARAF 独立升级 FUSE 应用程序

在 Karaf 上升级您的 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，以确保您正在使用正确的 Fuse 版本。

通常，您使用 Maven 来构建 Fuse 应用程序。Maven 是一个来自 Apache 的免费开源构建工具。Maven 配置在 Fuse 应用项目的 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，以便 Maven 构建过程获取正确的 Fuse 支持的工件集。

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 factory 创建到 MongoDB 的连接

从 Fuse 7.11，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：

- 更新应用程序，以将连接 bean 创建为 com.mongodb.client.MongoClient 实例。

  例如，创建与 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 MongoClients.create("mongodb://admin:password@192.168.99.102:32553");
```

- 评估并根据需要重构与 MongoClient 类公开的方法相关的代码。

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 已从 Spring Boot v1 升级到 v2，因此不再支持 v1。
- 升级到 Spring Framework 5。Camel 还应与 Spring 4.3.x 配合工作，但在将来的版本中将转发 Spring 5.x 的最低版本。
- 升级至 Karaf 4.2。您可以在 Karaf 4.1 上运行 Camel，但在此版本中，我们只正式支持 Karaf 4.2。
- 使用 toD DSL 优化为可能的组件重复使用端点和制作者。例如，基于 HTTP 的组件现在将重复使用带有发送到同一主机的动态 URI 的生产者（HTTP 客户端）。
- 现在，可将具有 read-lock idempotent/idempotent-changed 的 file2 使用者配置为延迟发行任务，以在文件视为进程中时（仅在有共享幂等库的主动/主动集群设置中使用）。
- 允许在请求/reply 模式的 Netty4 producer 上插件自定义请求/关系 id 管理器实施。Twitter 组件现在默认使用扩展模式来支持超过 140 个字符的调整
- REST DSL 制作者现在支持使用 endpointProperties 在 REST 配置中配置。
- Kafka 组件现在支持 HeaderFilterStrategy 来插件自定义实现来控制 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，以验证 REST 服务是否可以使用 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 camel-cxf consumer 的 CXF continuation 超时问题，这可能会导致使用者返回带有数据的响应，而不是触发调用 SOAP 客户端的超时问题。

  - 修复了 camel-cxf consumer在使用强健的单向操作时不会发布 UoW。

  - 修复了使用 AdviceWith 以及在Exception 上使用 usave 方法等问题。

  - 修复了并行处理和流传输模式中的 Splitter 可能会阻断，当迭代邮件正文时，迭代邮件正文在第一个调用的 next（）方法调用中引发异常。

  - 修复了 Kafka 用户，在 autoCommitEnable=false 时不自动提交。

  - 固定的文件消费者使用标记文件（默认情况下为 read-lock），这应该没有。

  - 修复了将手动提交与 Kafka 一起使用，以提供当前的记录偏移，而不是上一个记录（第一个使用 -1）。

  - 修复了 Java DSL 中基于内容的路由器可能无法解析 predicates 时 中的属性占位符。

Camel 2.23 更新

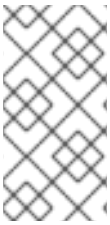
- 升级至 Spring Boot 2.1。
- 现在，可以使用 spring-boot 自动配置来配置其他组件级选项。这些选项包含在 spring-boot 组件元数据 JSON 文件描述符中，用于工具帮助。
- 添加了包含所有组件、数据格式和语言的 Spring Boot auto 配置选项的文档部分。
- 现在，所有 Camel Spring Boot starter JARs 都会在其 JARs 中包括 META-INF/spring-autoconfigure-metadata.properties 文件，以优化 Spring Boot 自动配置。
- Throttler 现在支持基于动态表达式的关联组，以便您可以将消息分组到不同的 throttled 集。
- Hystrix EIP 现在允许 Camel 的错误处理程序继承，如果已使用红色类别启用了错误处理，您可以再次重试整个 Hystrix EIP 块。
- SQL 和 ElSql 使用者现在支持路由形式的动态查询参数。请注意，此功能可通过使用简单表达式调用 Bean。
- swagger-restdsl maven 插件现在支持从 Swagger 规格文件中生成 DTO 模型类。
- 以下值得注意的问题已被修复：
  - 修正了聚合器2，不会传播控制标头以强制完成所有组，因此以后路由过程中使用了另一个聚合器 EIP 时，它不会再次发生。
  - 修复了当错误处理器中重新发送被激活时的 Tracer 无法正常工作。
  - XML 文档的内置类型转换器可能会输出将错误解析到 stdout，该错误现在使用日志记录 API 固定为输出。

- 修复了在基于消息正文时，使用 charset 选项的 SFTP 写入文件将无法正常工作。
- 修复了当在多个路由之间进行路由时，Zipkin root id 不会被重复使用，从而将它们分组到一个父范围中。
- 修复了在使用 HTTP 端点时，当主机名包含带有数字的 IP 地址时，被优化为D。
- 修复了 RabbitMQ 与请求/回复超过临时队列的 RabbitMQ 的问题，并使用手动确认模式。它不会确认临时队列（需要可以发出请求/恢复）。
- 修复了各种 HTTP 消费者组件，它们可能无法在 Allow header 中为 OPTIONS 请求返回所有允许的 HTTP 动词（比如使用 rest-dsl）。
- 修复了 FluentProducerTemplate 中的 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精心设计的 Maven 依赖项版本，从而为您保存每个 Maven 工件的版本，从而单独定义每个 Maven 工件的版本。

Fuse 的每个容器都有一个专用的 BOM 文件。



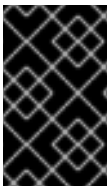
注意

您可以在此找到这些 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，这样您不需要在向 pom.xml 文件中添加依赖项时指定版本。

- 定义一组经过策展的依赖关系，这些依赖项是针对特定版本的 Fuse 完全测试和支持的。
- 简化 Fuse 升级.



重要

红帽只支持由 Fuse BOM 定义的一组依赖项。

7.3. 更新 FUSE 项目的 MAVEN 依赖项

要升级 Karaf 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 dependencyManagement 元素（或者在父 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>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</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 文件中的依赖关系后，*可以在不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，若要为 camel-velocity 组件添加依赖项，您要将以下 XML 片段添加到 pom.xml 文件中的 dependencies 元素中：

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

注意 version 元素是如何从这个依赖关系定义中省略的。

第 8 章 在 KARAF 上升级 FUSE STANDALONE

通过 Apache Karaf 升级机制的 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。如果升级导致部署的应用程序出现问题，它也允许您回滚升级。

升级安装程序文件与您用于在 Apache Karaf 上安装 Fuse 的 *文件相同*。

注意

要获取升级安装程序文件，请转至红帽客户门户的 Downloads 页面，再下载 Apache Karaf 上 Fuse 的最新安装存档（例如 fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip）。

- 第 8.1 节 “在 Karaf 上升级 Fuse 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. 在 KARAF 上升级 FUSE 的影响

升级机制可以对任何安装文件进行更新，包括 捆绑 JAR 和 静态文件 （例如，etc/ 目录下的配置文件）。Apache Karaf 升级过程中的 Fuse：

- 更新任何文件，包括捆绑 JAR、配置文件和任何静态文件。
- 在 data/ 目录下补丁当前容器实例（及其运行时存储）和底层安装。因此，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，在回滚补丁后安装的任何功能都会引用正确的补丁依赖关系。
- 如有必要，更新配置文件（如 etc/下的文件），自动合并您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，请参阅补丁日志以了解它们的处理方式。

- 大多数合并冲突会自动解决。例如，补丁机制会在属性文件的属性级别检测到冲突。它检测到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（包括静态文件）的所有更改，以便可以回滚补丁。

注意

rollup patching 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。确保在开始升级过程前完成所有先决条件。

先决条件

- 在升级前，请确保您在 Apache 手册安装有完好的 Fuse 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），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的环境中。例如，要添加 fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip 升级安装程序文件：

```
patch:add file:///path/to/fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip
```

注意

patch:find 命令仅用于查找和添加容器环境的最新热修复补丁；它无法用于应用完整的升级补丁。

流程

1. 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
```

2. **调用 patch:list 命令，以显示升级安装程序的列表。在这个列表中，[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
```

3. **通过调用 patch:simulate 命令并指定您要应用的升级 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()>
```

这会生成升级时对容器所做的更改的日志，但不会对容器进行任何实际更改。查看模拟日志，以了解容器要进行的更改。

4. 通过调用 **patch:install** 命令，并指定要应用的升级的升级 **ID** 来升级容器。例如：

```
karaf@root()> patch:install fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003
```

5. 通过搜索其中一个升级工件来验证升级。例如，如果您刚刚将 **Fuse 7.1.0** 升级到 **Fuse 7.2.0**，您可以搜索构建号 **7_11_1-00017-redhat-00001** 的捆绑包，如下所示：

```
karaf@root()> bundle:list -l | grep 7_11_1-00017-redhat-00001
 22 │ Active │  80 │ 7.11.1.fuse-7_11_1-00013-redhat-00003 │
mvn:org.jboss.fuse.modules/fuse-pax-transx-tm-narayana/7.11.1.fuse-7_11_1-00013-
redhat-00003
188 │ Active │  80 │ 7.11.1.fuse-7_11_1-00013-redhat-00003 │
mvn:org.jboss.fuse.modules.patch/patch-commands/7.11.1.fuse-7_11_1-00013-redhat-
00003
```

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

偶尔，升级可能无法工作，或者可能会向容器引入新的问题。在这些情况下，您可以使用 **patch:rollback** 命令，轻松将系统回滚到之前的状态。这一组的步骤会指导您完成此步骤。

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，容器需要重启来回滚升级。在这些情况下，容器会自动重启。由于 **OSGi** 运行时的高度动态性，重启过程中可能会看到一些与不兼容类相关的错误。这些错误与刚才启动或停止的 **OSGi** 服务相关，可以安全地忽略。