Red Hat Fuse 7.11

移行ガイド

Red Hat Fuse 7.11 への移行

Last Updated: 2023-07-25

Red Hat Fuse 7.11 移行ガイド

Red Hat Fuse 7.11 への移行

法律上の通知

概要

本ガイドは、Fuse インストールを Red Hat Fuse の最新バージョンにアップグレードする際に役立ちます。

目次

はじめに 3

多様性を受け入れるオープンソースの強化 4

第1章 FUSE ON OPENSHIFT のアップグレード 5

第2章 FUSE ONLINE のアップグレード 6

第3章 SPRING BOOT 2 へのアップグレード 7
3.1. 作業を開始する前に 7
3.2. SPRING BOOT 1 から SPRING BOOT 2 へのアップグレード 7

第4章 FABRIC8 MAVEN プラグインから OPENSHIFT MAVEN プラグインへの移行 9

第5章 SPRING BOOT スタンドアロンでの FUSE アプリケーションのアップグレード 10
5.1. CAMEL 移行に関する考慮事項 10
5.2. MAVEN 依存関係について 13
5.3. FUSE プロジェクトの MAVEN 依存関係の更新 13

第6章 JBOSS EAP スタンドアロンでの FUSE アプリケーションのアップグレード 15
6.1. CAMEL 移行に関する考慮事項 15
6.2. MAVEN 依存関係について 18
6.3. FUSE プロジェクトの MAVEN 依存関係の更新 18
6.4. JAVA EE の依存関係のアップグレード 19
6.5. 既存の FUSE ON JBOSS EAP インストールのアップグレード 20
6.6. FUSE と JBOSS EAP の同時アップグレード 20

第7章 KARAF スタンドアロンでの FUSE アプリケーションのアップグレード 22
7.1. CAMEL 移行に関する考慮事項 22
7.2. MAVEN 依存関係について 25
7.3. FUSE プロジェクトの MAVEN 依存関係の更新 25

第8章 KARAF での FUSE スタンドアロンのアップグレード 27
8.1. FUSE ON KARAF のアップグレードによる影響 27
8.2. KARAF での FUSE スタンドアロンのアップグレード 28
8.3. FUSE ON KARAF のアップグレードのロールバック 30

はじめに

本ガイドでは、Red Hat Fuse および Fuse アプリケーションの更新に関する情報を提供します。

注記

Fuse 6 から最新の Fuse 7 リリースに移行する場合は、本ガイドの手順に従う前に、Red Hat Fuse 7.0 Migration Guide の手順に従う必要があります。

2章*Fuse Online のアップグレード*

5章*Spring Boot スタンドアロンでの Fuse アプリケーションのアップグレード*

4章*Fabric8 Maven プラグインから Openshift Maven プラグインへの移行*

6章*JBoss EAP スタンドアロンでの Fuse アプリケーションのアップグレード*

7章*Karaf スタンドアロンでの Fuse アプリケーションのアップグレード*

8章*Karaf での Fuse スタンドアロンのアップグレード*

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 FUSE ON OPENSHIFT のアップグレード

Fuse 7.11 には、OpenShift Container Platform (OCP) 4.9 以降と連携できるようにする更新が含まれています。OCP 4.10 にアップグレードする場合は、OCP をバージョン 4.10 にアップグレードする **前** に、Fuse をバージョン 7.11 にアップグレードする必要があります。以前のバージョンの Fuse (7.10 より前) は OCP 4.9 以降をサポートしません。

詳細は Fuse on OpenShift ガイド を参照してください。

第2章 FUSE ONLINE のアップグレード

時々、パッチとセキュリティー修正が組み込まれた新しいアプリケーションイメージが Fuse に対してリリースされます。これらの更新は、Red Hat のエラータ更新チャネルを介して通知されます。その後、Fuse イメージをアップグレードできます。

OCP 4.x の場合、Upgrading Fuse by using the OperatorHub (OCP 4.x) の手順に従い、OpenShift OperatorHub を使用して Fuse 7.10 から 7.11 にアップグレードします。

Fuse 7.11 へのアップグレードにより、既存のインテグレーションを変更する必要があるかどうかを判断する必要があります。変更が必要ない場合でも、Fuse をアップグレードするときに、実行中のインテグレーションを再パブリッシュする必要があります。

OperatorHub(OCP 4.x) を使用した Fuse のアップグレード

OpenShift OperatorHub を使用して Fuse Online 7.10 から 7.11 にアップグレードします。

1. Fuse Online 7.9.x から Fuse 7.10.1 にアップグレードする場合は、リリースノートの **Upgrading from Fuse 7.9.x to 7.10.1 requires manual upgrade steps** の説明にしたがって、最初に Fuse Online 7.10.0 に手動でアップグレードする必要があります。

2. Fuse 7.11 には OpenShift Container Platform (OCP) 4.6 以降が必要です。OCP 4.5 以前を使用している場合、Fuse 7.11 にアップグレードするには、OCP 4.6 以降にアップグレードする必要があります。

3. OCP 4.9 では、7.11 にアップグレードすると、Fuse Operator のアップグレードプロセス時に以下の警告が表示されます。
   W1219 18:38:58.064578 1 warnings.go:70] extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress

   この警告は、クライアント (Fuse が Kubernetes/OpenShift API 初期化コードに使用する) が非推奨の Ingress バージョンにアクセスするために表示されます。この警告は、非推奨の API が完全に使用されていることを示すものではなく、Fuse 7.11 へアップグレードすることに問題はありません。

Fuse Online 7.10 以前のバージョンから、Fuse Online 7.11 の新しいバージョンへのアップグレードプロセスは、Fuse Online のインストール時に選択した **Approval Strategy** によって異なります。

- **Automatic** (自動) 更新の場合、新しいバージョンの Fuse Operator が使用できるようになると、人的な介入なしで OpenShfit Operator Lifecycle Manager (OLM) によって、Fuse Online の稼働中のインスタンスが自動的にアップグレードされます。

- **Manual** (手動) 更新の場合、Operator の新しいバージョンが使用できるようになると、OLM によって更新リクエストが作成されます。クラスター管理者は、OpenShift ドキュメントの **Manually approving a pending Operator upgrade**セクションで説明されているように、更新リクエストを手動で承認して Fuse Online Operator を新しいバージョンに更新する必要があります。

インフラストラクチャーのアップグレード中およびアップグレード後も、既存のインテグレーションは引き続き Fuse ライブラリーおよび依存関係の古いバージョンで実行されます。

更新された Fuse Online バージョンで既存のインテグレーションを実行するには、インテグレーションを再パブリッシュする必要があります。

第3章 SPRING BOOT 2 へのアップグレード

本章では、アプリケーションを Spring Boot 1 から Spring Boot 2.0 にアップグレードする方法を説明します。

3.1. 作業を開始する前に

Spring Boot 2 への移行を開始する前に、システム要件と依存関係を確認する必要があります。

- 最新バージョンの 1.5.x へのアップグレード
  - 開始する前に、利用可能な最新バージョンの 1.5.x にアップグレードします。これは、そのラインの最新の依存関係に対してビルドするようにするためです。
- 依存関係の確認
  - Spring Boot 2 への移行により、多くの依存関係がアップグレードされます。2.0.x の依存関係管理に対して 1.5.x の依存関係管理を確認し、プロジェクトへの影響を評価します。
  - Spring Boot によって管理されない依存関係の互換バージョンを特定し、それらの明示的なバージョンを定義します。
- カスタム設定の確認
  - プロジェクトが定義するカスタム設定を、アップグレード時に確認する必要がある場合があります。これを標準の auto-configuration を使用して置き換えることができる場合は、アップグレードの前に行います。
- システム要件の確認
  - Spring Boot 2.0 には Java 8 以降が必要です。
  - Spring Framework 5.0 も必要です。
  - Java 6 および 7 はサポート対象外になりました。

3.2. SPRING BOOT 1 から SPRING BOOT 2 へのアップグレード

プロジェクトとその依存関係の状態を確認したら、Spring Boot 2.x の最新のメンテナーンスリリースにアップグレードします。段階的にアップグレードすることをお勧めします。たとえば、最初に Spring Boot 1.5 から Spring Boot 2.0 にアップグレードしてから 2.1 にアップグレードし、続いて Spring Boot 2 の最新のメンテナーンスリリースにアップグレードします。

設定プロパティーの移行

Spring Boot 2.0 では、多くの設定プロパティーの名前が変更または削除されました。したがって、それに応じて **application.properties/application.yml** を更新する必要があります。これは、新しい **spring-boot-properties-migrator** モジュールを利用して実行できます。プロジェクトに依存関係として追加されると、起動時にアプリケーションの環境を分析して診断を出力するだけでなく、実行時にプロパティーを一時的に移行します。

手順

1. **spring-boot-properties-migrator** モジュールをプロジェクトの **pom.xml** の依存関係セクションに追加します。

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

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

注記

移行が完了したら、プロジェクトの依存関係からこのモジュールを削除してください。

第4章 FABRIC8 MAVEN プラグインから OPENSHIFT MAVEN プラグインへの移行

fabric8-maven-plugin は Fuse 7.11 から完全に削除されました。Fuse on OpenShift で Maven プロジェクトをビルドおよびデプロイに代わりに **openshift-maven-plugin** を使用することが推奨されます。

手順

以下の手順に従って、openshift-maven プラグインを使用できるようにアプリケーションを更新します。

1. アプリケーションの **src/main/fabric8** ディレクトリーの名前を **src/main/jkube** に変更します。

2. プロジェクトの pom.xml で **org.jboss.redhat-fuse:fabric8-maven-plugin** 依存関係を見つけ、これを **org.jboss.redhat-fuse:openshift-maven-plugin** に変更します。pom.xml のサンプルを参照してください。

3. 依存関係を確認します。たとえば、**org.arquillian.cube:arquillian-cube-openshift**、**org.jboss.arquillian.junit:arquillian-junit-container**、**io.fabric8:kubernetes-assertions** は、本書の例では使用されなくなり、今後は必要なくなる可能性があります。

4. 移行後に API の変更を反映するために使用できるサンプルテストを作成することができます。詳細は、Spring Boot Camel quickstart のサンプルテストを参照してください。

関連情報

- OpenShift Maven plugin.

第5章 SPRING BOOT スタンドアロンでの FUSE アプリケーションのアップグレード

Spring Boot で Fuse アプリケーションをアップグレードするには、以下を行います。

- 「Camel 移行に関する考慮事項」で説明するように、Apache Camel の更新について考慮する必要があります。
- Fuse プロジェクトの Maven 依存関係を更新し、Fuse の正しいバージョンを使用するようにする必要があります。

通常は、Maven を使用して Fuse アプリケーションを構築します。Maven は、Apache の無料のオープンソースビルドツールです。Maven 設定は Fuse アプリケーションプロジェクトの **pom.xml** ファイルで定義されます。Fuse プロジェクトのビルド中、Maven が外部リポジトリーを探し、必要なアーティファクトをダウンロードするのがデフォルトの動作になります。Maven ビルドプロセスで Fuse がサポートするアーティファクトの正しいセットを選択できるように、Fuse Bill of Materials (BOM) の依存関係を **pom.xml** ファイルに追加します。

以下のセクションでは、Maven の依存関係と Fuse プロジェクトでの更新方法を説明します。

- 「Maven 依存関係について」
- 「Fuse プロジェクトの Maven 依存関係の更新」

5.1. CAMEL 移行に関する考慮事項

MongoClients ファクトリーを使用した MongoDB への接続の作成

Fuse 7.11 から、**com.mongodb.MongoClient** の代わりに **com.mongodb.client.MongoClient** を使用して、MongoDB への接続を作成します (フルパスの追加の **.client** サブパッケージに注意してください)。

既存の Fuse アプリケーションのいずれかが **camel-mongodb** コンポーネントを使用する場合は、以下を行う必要があります。

- アプリケーションを更新して、接続 Bean を **com.mongodb.client.MongoClient** インスタンスとして作成します。
  たとえば、以下のように MongoDB への接続を作成します。

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

  続いて、以下の例のように MongoClient Bean を作成できます。

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

- MongoClient クラスによって公開されたメソッドに関連するコードを評価し、必要に応じリファクターリングします。

Camel 2.23

Red Hat Fuse は Apache Camel 2.23 を使用します。Fuse 7.8 にアップグレードする場合、以下の Camel 2.22 および 2.23 への更新を考慮する必要があります。

Camel 2.22 への更新

- Camel は Spring Boot v1 から v2 にアップグレードされたため、v1 はサポート対象外になりました。
- Spring Framework 5 へのアップグレード。Camel は Spring 4.3.x でも動作しますが、今後 Spring 5.x が今後のリリースで最小の Spring バージョンになります。
- Karaf 4.2 へのアップグレード。Camel は Karaf 4.1 で実行できますが、本リリースでは Karaf 4.2 のみを公式にサポートします。
- toD DSL の使用が最適化され、可能な限りコンポーネントのエンドポイントとプロデューサーを再利用します。たとえば、HTTP ベースのコンポーネントは、同じホストに送信する動的 URI でプロデューサー (HTTP クライアント) を再利用するようになりました。
- read-lock idempotent/idempotent-changed を持つ File2 コンシューマーは、ファイルが処理中であると見なされるウィンドウを拡張するためにリリースタスクを遅らせるように設定できるようになりました。これは、他のノードが、処理されたファイルを処理可能なファイルとしてすぐに認識しないように、共有の idempotent レポジトリーを持つアクティブ/アクティブクラスター設定で使用できます (readLockRemoveOnCommit=true がある場合にのみ必要です)。
- リクエスト/リプライモードの Netty4 プロデューサーでカスタムリクエスト/リプライ相関 ID マネージャー実装をプラグインできるようにします。Twitter コンポーネントはデフォルトで拡張モードを使用し、140 文字を超えるツイートをサポートするようになりました。
- REST DSL プロデューサーが endpointProperties を使用して REST 設定で設定されるようになりました。
- Kafka コンポーネントは、Camel と Kafka メッセージ間のヘッダーマッピングを制御するために、カスタム実装をプラグインするための HeaderFilterStrategy をサポートするようになりました。
- REST DSL は、REST サービスで Content-Type/Accept ヘッダーが使用可能なことを検証するためのクライアント要求検証をサポートするようになりました。
- Camel には Service Registry SPI を持つようになりました。これにより、Camel 実装または Spring Cloud を使用してサービスレジストリー (consul、etcd、zookeeper など) にルートを登録できるようになりました。
- SEDA コンポーネントのデフォルトキューサイズが、無制限ではなく 1000 になりました。
- 以下の注目すべき問題が修正されました。
  - camel-cxf コンシューマーでの CXF の継続タイムアウトの問題が修正されました。この問題により、呼び出し元 SOAP クライアントにタイムアウトをトリガーする代わりに、コンシューマーがデータで応答を返す可能性がありました。
  - 堅牢な一方向操作を使用する場合に camel-cxf コンシューマーが UoW をリリースしない問題が修正されました。
  - onException などで AdviceWith や weave メソッドを使用しても機能しない問題が修正されました。
  - 並列処理およびストリーミングモードの Splitter がブロックされる可能性があり、イテレータが最初に呼び出された next() メソッド呼び出しで例外を出力したときにメッセージ本文を反復する場合がありましたが、この問題が修正されました。
  - autoCommitEnable=false の場合、Kafka コンシューマーが自動コミットされないように修正されました。

- ファイルコンシューマーはデフォルトで markerFile を read-lock として使用していましたが、これが修正され、なくなりました。
- Kafka で手動コミットを使用して、以前のレコードオフセットではなく現在のレコードオフセットを提供するように修正されました (最初の場合は -1)。
- 述語の場合に、Java DSL のコンテンツベースのルーターがプロパティープレースホルダーを解決できない問題が解決されました。

Camel 2.23 への更新

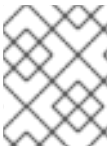
- Spring Boot 2.1 へのアップグレード。
- 追加のコンポーネントレベルのオプションが、spring-boot auto-configuration を使用して設定できるようになりました。これらのオプションは、ツールの支援のための spring-boot コンポーネントメタデータ JSON ファイル記述子に含まれます。
- すべてのコンポーネント、データ形式、および言語の Spring Boot 自動設定オプションがすべて含まれるドキュメントセクションが追加されました。
- すべての Camel Spring Boot スターター JAR には、Spring Boot の自動設定を最適化するために JAR に **META-INF/spring-autoconfigure-metadata.properties** ファイルが追加されるようになりました。
- Throttler は動的表現に基づいた相関グループをサポートするようになりました。これにより、メッセージを異なるスロットルセットにグループ化できるようになりました。
- Hystrix EIP では、再配信でのエラー処理が有効になっている場合に、Hystrix EIP ブロック全体を再度リトライできるように Camel のエラーハンドラーの継承が可能になりました。
- SQL および ElSql コンシューマーが、ルート形式の動的クエリーパラメーターをサポートするようになりました。この機能は、Simple 式を使用した Bean の呼び出しに限定されることに注意してください。
- swagger-restdsl maven プラグインが、Swagger 仕様ファイルから DTO モデルクラスを生成できるようになりました。
- 以下の注目すべき問題が修正されました。
  - Aggregator2 は、すべてのグループの完了を強制するための制御ヘッダーを伝播しないように修正されました。そのため、ルーティング中に後で別のアグリゲーター EIP が使用されても、再び発生することはありません。
  - エラーハンドラーで再配信が有効な場合にトレーサーが機能しない問題が修正されました。
  - XML ドキュメントの組み込み型コンバーターは、stdout に解析エラーを出力する可能性がありますが、ロギング API を使用して出力するように修正されました。
  - メッセージボディーがストリーミングベースであった場合、charset オプションを使用した SFTP のファイル書き込みが動作し無い問題が修正されました。
  - 複数のルートをルーティングして 1 つの親スパンにグループ化する場合に、Zipkin ルート ID が再利用されないように修正されました。
  - ホスト名に数字の IP アドレスが含まれる場合に、HTTP エンドポイントを使用する場合に最適化された toD にバグがあった問題が修正されました。

- RabbitMQ の一時キューを経由した要求/応答、および手動確認モード使用の問題が修正されました。一時キューを認識しませんでした (要求/応答を可能にするために必要)。
- OPTIONS リクエストの Allow ヘッダーで許可されるすべての HTTP 動詞を返さない可能性のあるさまざまな HTTP コンシューマーコンポーネントが修正されました (rest-dsl を使用する場合など)。
- FluentProducerTemplate のスレッドセーフの問題が修正されました。

5.2. MAVEN 依存関係について

Maven BOM (Bill of Materials) ファイルの目的は、正常に動作する Maven 依存関係バージョンのセットを提供し、各 Maven アーティファクトに対して個別にバージョンを定義する必要をなくすことです。

Fuse が実行される各コンテナーには専用の BOM ファイルがあります。



注記

これらの BOM ファイルは https://github.com/jboss-fuse/redhat-fuse にあります。または、BOM ファイル更新の詳細は latest Release Notes を参照してください。

Fuse BOM には以下の利点があります。

- Maven 依存関係のバージョンを定義するため、依存関係を **pom.xml** ファイルに追加するときにバージョンを指定する必要がありません。
- 特定バージョンの Fuse に対して完全にテストされ、完全にサポートする依存関係のセットを定義します。
- Fuse のアップグレードを簡素化します。

重要

Fuse BOM によって定義される依存関係のセットのみが Red Hat によってサポートされます。

5.3. FUSE プロジェクトの MAVEN 依存関係の更新

Spring Boot の Fuse アプリケーションをアップグレードするには、プロジェクトの Maven 依存関係を更新します。

手順

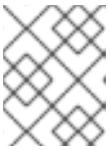
1. プロジェクトの **pom.xml** ファイルを開きます。
2. 以下の例のように、プロジェクトの **pom.xml** ファイル (または、場合によっては親 **pom.xml** ファイル) に **dependencyManagement** 要素を追加します。

```
<?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>
```



注記

Spring Boot バージョンも更新するようにしてください。これは通常、**pom.xml** ファイルの Fuse バージョンにあります。

```
  <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. **pom.xml** ファイルを保存します。

BOM を **pom.xml** ファイルで依存関係として指定した後、アーティファクトのバージョンを **指定せずに** Maven 依存関係を **pom.xml** ファイルに追加できるようになります。たとえば、**camel-velocity** コンポーネントの依存関係を追加するには、以下の XML フラグメントを **pom.xml** ファイルの **dependencies** 要素に追加します。

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

この依存関係の定義では、**version** 要素が省略されることに注意してください。

第6章 JBOSS EAP スタンドアロンでの FUSE アプリケーションのアップグレード

JBoss EAP で Fuse アプリケーションをアップグレードするには、以下を行います。

- 「Camel 移行に関する考慮事項」で説明するように、Apache Camel の更新について考慮する必要があります。
- Fuse プロジェクトの Maven 依存関係を更新し、Fuse の正しいバージョンを使用するようにする必要があります。
  通常は、Maven を使用して Fuse アプリケーションを構築します。Maven は、Apache の無料のオープンソースビルドツールです。Maven 設定は Fuse アプリケーションプロジェクトの **pom.xml** ファイルで定義されます。Fuse プロジェクトのビルド中、Maven が外部リポジトリーを探し、必要なアーティファクトをダウンロードするのがデフォルトの動作になります。Maven ビルドプロセスで Fuse がサポートするアーティファクトの正しいセットを選択できるように、Fuse Bill of Materials (BOM) の依存関係を **pom.xml** ファイルに追加します。

  以下のセクションでは、Maven の依存関係と Fuse プロジェクトでの更新方法を説明します。

  - 「Maven 依存関係について」
  - 「Fuse プロジェクトの Maven 依存関係の更新」
- 「Java EE の依存関係のアップグレード」で説明するように、Fuse プロジェクトの Maven 依存関係を更新し、アップグレードされたバージョンの Java EE の依存関係を使用するようにする必要があります。

6.1. CAMEL 移行に関する考慮事項

MongoClients ファクトリーを使用した MongoDB への接続の作成

Fuse 7.11 から、**com.mongodb.MongoClient** の代わりに **com.mongodb.client.MongoClient** を使用して、MongoDB への接続を作成します (フルパスの追加の **.client** サブパッケージに注意してください)。

既存の Fuse アプリケーションのいずれかが **camel-mongodb** コンポーネントを使用する場合は、以下を行う必要があります。

- アプリケーションを更新して、接続 Bean を **com.mongodb.client.MongoClient** インスタンスとして作成します。
  たとえば、以下のように MongoDB への接続を作成します。

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

  続いて、以下の例のように MongoClient Bean を作成できます。

  ```
  return MongoClients.create("mongodb://admin:password@192.168.99.102:32553");
  ```
- MongoClient クラスによって公開されたメソッドに関連するコードを評価し、必要に応じリファクターリングします。

Camel 2.23

Red Hat Fuse は Apache Camel 2.23 を使用します。Fuse 7.8 にアップグレードする場合、以下の Camel 2.22 および 2.23 への更新を考慮する必要があります。

Camel 2.22 への更新

- Camel は Spring Boot v1 から v2 にアップグレードされたため、v1 はサポート対象外になりました。
- Spring Framework 5 へのアップグレード。Camel は Spring 4.3.x でも動作しますが、今後 Spring 5.x が今後のリリースで最小の Spring バージョンになります。
- Karaf 4.2 へのアップグレード。Camel は Karaf 4.1 で実行できますが、本リリースでは Karaf 4.2 のみを公式にサポートします。
- toD DSL の使用が最適化され、可能な限りコンポーネントのエンドポイントとプロデューサーを再利用します。たとえば、HTTP ベースのコンポーネントは、同じホストに送信する動的 URI でプロデューサー (HTTP クライアント) を再利用するようになりました。
- read-lock idempotent/idempotent-changed を持つ File2 コンシューマーは、ファイルが処理中であると見なされるウィンドウを拡張するためにリリースタスクを遅らせるように設定できるようになりました。これは、他のノードが、処理されたファイルを処理可能なファイルとしてすぐに認識しないように、共有の idempotent レポジトリーを持つアクティブ/アクティブクラスター設定で使用できます (readLockRemoveOnCommit=true がある場合にのみ必要です)。
- リクエスト/リプライモードの Netty4 プロデューサーでカスタムリクエスト/リプライ相関 ID マネージャー実装をプラグインできるようにします。Twitter コンポーネントはデフォルトで拡張モードを使用し、140 文字を超えるツイートをサポートするようになりました。
- REST DSL プロデューサーが endpointProperties を使用して REST 設定で設定されるようになりました。
- Kafka コンポーネントは、Camel と Kafka メッセージ間のヘッダーマッピングを制御するために、カスタム実装をプラグインするための HeaderFilterStrategy をサポートするようになりました。
- REST DSL は、REST サービスで Content-Type/Accept ヘッダーが使用可能なことを検証するためのクライアント要求検証をサポートするようになりました。
- Camel には Service Registry SPI を持つようになりました。これにより、Camel 実装または Spring Cloud を使用してサービスレジストリー (consul、etcd、zookeeper など) にルートを登録できるようになりました。
- SEDA コンポーネントのデフォルトキューサイズが、無制限ではなく 1000 になりました。
- 以下の注目すべき問題が修正されました。
    - camel-cxf コンシューマーでの CXF の継続タイムアウトの問題が修正されました。この問題により、呼び出し元 SOAP クライアントにタイムアウトをトリガーする代わりに、コンシューマーがデータで応答を返す可能性がありました。
    - 堅牢な一方向操作を使用する場合に camel-cxf コンシューマーが UoW をリリースしない問題が修正されました。
    - onException などで AdviceWith や weave メソッドを使用しても機能しない問題が修正されました。
    - 並列処理およびストリーミングモードの Splitter がブロックされる可能性があり、イテレータが最初に呼び出された next() メソッド呼び出しで例外を出力したときにメッセージ本文を反復する場合がありましたが、この問題が修正されました。

- autoCommitEnable=false の場合、Kafka コンシューマーが自動コミットされないように修正されました。
- ファイルコンシューマーはデフォルトで markerFile を read-lock として使用していましたが、これが修正され、なくなりました。
- Kafka で手動コミットを使用して、以前のレコードオフセットではなく現在のレコードオフセットを提供するように修正されました (最初の場合は -1)。
- 述語の場合に、Java DSL のコンテンツベースのルーターがプロパティープレースホルダーを解決できない問題が解決されました。

Camel 2.23 への更新

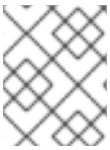
- Spring Boot 2.1 へのアップグレード。
- 追加のコンポーネントレベルのオプションが、spring-boot auto-configuration を使用して設定できるようになりました。これらのオプションは、ツールの支援のための spring-boot コンポーネントメタデータ JSON ファイル記述子に含まれます。
- すべてのコンポーネント、データ形式、および言語の Spring Boot 自動設定オプションがすべて含まれるドキュメントセクションが追加されました。
- すべての Camel Spring Boot スターター JAR には、Spring Boot の自動設定を最適化するために JAR に **META-INF/spring-autoconfigure-metadata.properties** ファイルが追加されるようになりました。
- Throttler は動的表現に基づいた相関グループをサポートするようになりました。これにより、メッセージを異なるスロットルセットにグループ化できるようになりました。
- Hystrix EIP では、再配信でのエラー処理が有効になっている場合に、Hystrix EIP ブロック全体を再度リトライできるように Camel のエラーハンドラーの継承が可能になりました。
- SQL および ElSql コンシューマーが、ルート形式の動的クエリーパラメーターをサポートするようになりました。この機能は、Simple 式を使用した Bean の呼び出しに限定されることに注意してください。
- swagger-restdsl maven プラグインが、Swagger 仕様ファイルから DTO モデルクラスを生成できるようになりました。
- 以下の注目すべき問題が修正されました。
  - Aggregator2 は、すべてのグループの完了を強制するための制御ヘッダーを伝播しないように修正されました。そのため、ルーティング中に後で別のアグリゲーター EIP が使用されても、再び発生することはありません。
  - エラーハンドラーで再配信が有効な場合にトレーサーが機能しない問題が修正されました。
  - XML ドキュメントの組み込み型コンバーターは、stdout に解析エラーを出力する可能性がありますが、ロギング API を使用して出力するように修正されました。
  - メッセージボディーがストリーミングベースであった場合、charset オプションを使用した SFTP のファイル書き込みが動作し無い問題が修正されました。
  - 複数のルートをルーティングして 1 つの親スパンにグループ化する場合に、Zipkin ルート ID が再利用されないように修正されました。

- ホスト名に数字の IP アドレスが含まれる場合に、HTTP エンドポイントを使用する場合に最適化された toD にバグがあった問題が修正されました。
- RabbitMQ の一時キューを経由した要求/応答、および手動確認モード使用の問題が修正されました。一時キューを認識しませんでした (要求/応答を可能にするために必要)。
- OPTIONS リクエストの Allow ヘッダーで許可されるすべての HTTP 動詞を返さない可能性のあるさまざまな HTTP コンシューマーコンポーネントが修正されました (rest-dsl を使用する場合など)。
- FluentProducerTemplate のスレッドセーフの問題が修正されました。

6.2. MAVEN 依存関係について

Maven BOM (Bill of Materials) ファイルの目的は、正常に動作する Maven 依存関係バージョンのセットを提供し、各 Maven アーティファクトに対して個別にバージョンを定義する必要をなくすことです。

Fuse が実行される各コンテナーには専用の BOM ファイルがあります。



注記

これらの BOM ファイルは https://github.com/jboss-fuse/redhat-fuse にあります。または、BOM ファイル更新の詳細は latest Release Notes を参照してください。

Fuse BOM には以下の利点があります。

- Maven 依存関係のバージョンを定義するため、依存関係を **pom.xml** ファイルに追加するときにバージョンを指定する必要がありません。
- 特定バージョンの Fuse に対して完全にテストされ、完全にサポートする依存関係のセットを定義します。
- Fuse のアップグレードを簡素化します。

重要

Fuse BOM によって定義される依存関係のセットのみが Red Hat によってサポートされます。

6.3. FUSE プロジェクトの MAVEN 依存関係の更新

JBoss EAP の Fuse アプリケーションをアップグレードするには、プロジェクトの Maven 依存関係を更新します。

手順

1. プロジェクトの **pom.xml** ファイルを開きます。
2. 以下の例のように、プロジェクトの **pom.xml** ファイル (または、場合によっては親 **pom.xml** ファイル) に **dependencyManagement** 要素を追加します。

```
<?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. **pom.xml** ファイルを保存します。

BOM を **pom.xml** ファイルで依存関係として指定した後、アーティファクトのバージョンを **指定せずに** Maven 依存関係を **pom.xml** ファイルに追加できるようになります。たとえば、**camel-velocity** コンポーネントの依存関係を追加するには、以下の XML フラグメントを **pom.xml** ファイルの **dependencies** 要素に追加します。

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

この依存関係の定義では、**version** 要素が省略されることに注意してください。

6.4. JAVA EE の依存関係のアップグレード

Fuse 7.8 では、BOM ファイルの一部の管理依存関係が **groupId** または **artifactId** プロパティーを更新するため、それに応じてプロジェクトの **pom.xml** ファイルを更新する必要があります。

手順

1. プロジェクトの **pom.xml** ファイルを開きます。
2. **org.jboss.spec.javax.transaction** バージョンを 1.2 から 1.3 に、 **org.jboss.spec.javax.servlet** バージョンを 3.1 から 4.0 に変更するには、以下の例のように依存関係を更新します。

```
<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. Java EE API から Jakarta EE に移行するには、以下の例のように、**groupId** ごとに **javax.*** を **jakarta.*** に置き換え、個別の依存関係の **artifactId** を変更します。

```
<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. 既存の FUSE ON JBOSS EAP インストールのアップグレード

以下の手順で、既存の Fuse on JBoss EAP インストールをアップグレードする方法を説明します。

手順

1. ある JBoss EAP マイナーリリースから別のマイナーリリースにアップグレードするには、JBoss EAP Patching and Upgrading Guide の手順に従う必要があります。

2. Fuse を更新するには、Installing on JBoss EAP ガイドの説明に従って、Fuse on JBoss EAP インストーラーを実行する必要があります。

注記

Fuse アプリケーションの再コンパイルまたは再デプロイは必要ありません。

6.6. FUSE と JBOSS EAP の同時アップグレード

以下の手順で、Fuse インストールと JBoss EAP ランタイムを同時にアップグレードする方法を説明します (例:JBoss EAP 7.2 上の Fuse 7.7 から JBoss EAP 7.3 上の Fuse 7.8 への移行)。

警告

Fuse と JBoss EAP ランタイムの**両方** をアップグレードする場合、Red Hat では Fuse と JBoss EAP ランタイム両方の新規インストールを実施することを推奨します。

手順

1. JBoss EAP ランタイムの新しいインストールを実行するには、Installing on JBoss EAP ガイドの手順に従います。

2. Fuse の新規インストールを実行するには、Installing on JBoss EAP ガイドの説明に従って、Fuse on JBoss EAP インストーラーを実行します。

第7章 KARAF スタンドアロンでの FUSE アプリケーションのアップグレード

Karaf で Fuse アプリケーションをアップグレードするには、以下を行います。

- 「Camel 移行に関する考慮事項」で説明するように、Apache Camel の更新について考慮する必要があります。
- Fuse プロジェクトの Maven 依存関係を更新し、Fuse の正しいバージョンを使用するようにする必要があります。

通常は、Maven を使用して Fuse アプリケーションを構築します。Maven は、Apache の無料のオープンソースビルドツールです。Maven 設定は Fuse アプリケーションプロジェクトの **pom.xml** ファイルで定義されます。Fuse プロジェクトのビルド中、Maven が外部リポジトリーを探し、必要なアーティファクトをダウンロードするのがデフォルトの動作になります。Maven ビルドプロセスで Fuse がサポートするアーティファクトの正しいセットを選択できるように、Fuse Bill of Materials (BOM) の依存関係を **pom.xml** ファイルに追加します。

以下のセクションでは、Maven の依存関係と Fuse プロジェクトでの更新方法を説明します。

- 「Maven 依存関係について」
- 「Fuse プロジェクトの Maven 依存関係の更新」

7.1. CAMEL 移行に関する考慮事項

MongoClients ファクトリーを使用した MongoDB への接続の作成

Fuse 7.11 から、**com.mongodb.MongoClient** の代わりに **com.mongodb.client.MongoClient** を使用して、MongoDB への接続を作成します (フルパスの追加の **.client** サブパッケージに注意してください)。

既存の Fuse アプリケーションのいずれかが **camel-mongodb** コンポーネントを使用する場合は、以下を行う必要があります。

- アプリケーションを更新して、接続 Bean を **com.mongodb.client.MongoClient** インスタンスとして作成します。
  たとえば、以下のように MongoDB への接続を作成します。

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

  続いて、以下の例のように MongoClient Bean を作成できます。

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

- MongoClient クラスによって公開されたメソッドに関連するコードを評価し、必要に応じリファクターリングします。

Camel 2.23

Red Hat Fuse は Apache Camel 2.23 を使用します。Fuse 7.8 にアップグレードする場合、以下の Camel 2.22 および 2.23 への更新を考慮する必要があります。

Camel 2.22 への更新

- Camel は Spring Boot v1 から v2 にアップグレードされたため、v1 はサポート対象外になりました。
- Spring Framework 5 へのアップグレード。Camel は Spring 4.3.x でも動作しますが、今後 Spring 5.x が今後のリリースで最小の Spring バージョンになります。
- Karaf 4.2 へのアップグレード。Camel は Karaf 4.1 で実行できますが、本リリースでは Karaf 4.2 のみを公式にサポートします。
- toD DSL の使用が最適化され、可能な限りコンポーネントのエンドポイントとプロデューサーを再利用します。たとえば、HTTP ベースのコンポーネントは、同じホストに送信する動的 URI でプロデューサー (HTTP クライアント) を再利用するようになりました。
- read-lock idempotent/idempotent-changed を持つ File2 コンシューマーは、ファイルが処理中であると見なされるウィンドウを拡張するためにリリースタスクを遅らせるように設定できるようになりました。これは、他のノードが、処理されたファイルを処理可能なファイルとしてすぐに認識しないように、共有の idempotent レポジトリーを持つアクティブ/アクティブクラスター設定で使用できます (readLockRemoveOnCommit=true がある場合にのみ必要です)。
- リクエスト/リプライモードの Netty4 プロデューサーでカスタムリクエスト/リプライ相関 ID マネージャー実装をプラグインできるようにします。Twitter コンポーネントはデフォルトで拡張モードを使用し、140 文字を超えるツイートをサポートするようになりました。
- REST DSL プロデューサーが endpointProperties を使用して REST 設定で設定されるようになりました。
- Kafka コンポーネントは、Camel と Kafka メッセージ間のヘッダーマッピングを制御するために、カスタム実装をプラグインするための HeaderFilterStrategy をサポートするようになりました。
- REST DSL は、REST サービスで Content-Type/Accept ヘッダーが使用可能なことを検証するためのクライアント要求検証をサポートするようになりました。
- Camel には Service Registry SPI を持つようになりました。これにより、Camel 実装または Spring Cloud を使用してサービスレジストリー (consul、etcd、zookeeper など) にルートを登録できるようになりました。
- SEDA コンポーネントのデフォルトキューサイズが、無制限ではなく 1000 になりました。
- 以下の注目すべき問題が修正されました。
  - camel-cxf コンシューマーでの CXF の継続タイムアウトの問題が修正されました。この問題により、呼び出し元 SOAP クライアントにタイムアウトをトリガーする代わりに、コンシューマーがデータで応答を返す可能性がありました。
  - 堅牢な一方向操作を使用する場合に camel-cxf コンシューマーが UoW をリリースしない問題が修正されました。
  - onException などで AdviceWith や weave メソッドを使用しても機能しない問題が修正されました。
  - 並列処理およびストリーミングモードの Splitter がブロックされる可能性があり、イテレータが最初に呼び出された next() メソッド呼び出しで例外を出力したときにメッセージ本文を反復する場合がありましたが、この問題が修正されました。
  - autoCommitEnable=false の場合、Kafka コンシューマーが自動コミットされないように修正されました。

- ファイルコンシューマーはデフォルトで markerFile を read-lock として使用していましたが、これが修正され、なくなりました。
- Kafka で手動コミットを使用して、以前のレコードオフセットではなく現在のレコードオフセットを提供するように修正されました (最初の場合は -1)。
- 述語の場合に、Java DSL のコンテンツベースのルーターがプロパティープレースホルダーを解決できない問題が解決されました。

Camel 2.23 への更新

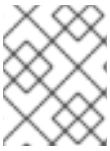
- Spring Boot 2.1 へのアップグレード。
- 追加のコンポーネントレベルのオプションが、spring-boot auto-configuration を使用して設定できるようになりました。これらのオプションは、ツールの支援のための spring-boot コンポーネントメタデータ JSON ファイル記述子に含まれます。
- すべてのコンポーネント、データ形式、および言語の Spring Boot 自動設定オプションがすべて含まれるドキュメントセクションが追加されました。
- すべての Camel Spring Boot スターター JAR には、Spring Boot の自動設定を最適化するために JAR に **META-INF/spring-autoconfigure-metadata.properties** ファイルが追加されるようになりました。
- Throttler は動的表現に基づいた相関グループをサポートするようになりました。これにより、メッセージを異なるスロットルセットにグループ化できるようになりました。
- Hystrix EIP では、再配信でのエラー処理が有効になっている場合に、Hystrix EIP ブロック全体を再度リトライできるように Camel のエラーハンドラーの継承が可能になりました。
- SQL および ElSql コンシューマーが、ルート形式の動的クエリーパラメーターをサポートするようになりました。この機能は、Simple 式を使用した Bean の呼び出しに限定されることに注意してください。
- swagger-restdsl maven プラグインが、Swagger 仕様ファイルから DTO モデルクラスを生成できるようになりました。
- 以下の注目すべき問題が修正されました。
  - Aggregator2 は、すべてのグループの完了を強制するための制御ヘッダーを伝播しないように修正されました。そのため、ルーティング中に後で別のアグリゲーター EIP が使用されても、再び発生することはありません。
  - エラーハンドラーで再配信が有効な場合にトレーサーが機能しない問題が修正されました。
  - XML ドキュメントの組み込み型コンバーターは、stdout に解析エラーを出力する可能性がありますが、ロギング API を使用して出力するように修正されました。
  - メッセージボディーがストリーミングベースであった場合、charset オプションを使用した SFTP のファイル書き込みが動作し無い問題が修正されました。
  - 複数のルートをルーティングして 1 つの親スパンにグループ化する場合に、Zipkin ルート ID が再利用されないように修正されました。
  - ホスト名に数字の IP アドレスが含まれる場合に、HTTP エンドポイントを使用する場合に最適化された toD にバグがあった問題が修正されました。

- RabbitMQ の一時キューを経由した要求/応答、および手動確認モード使用の問題が修正されました。一時キューを認識しませんでした (要求/応答を可能にするために必要)。
- OPTIONS リクエストの Allow ヘッダーで許可されるすべての HTTP 動詞を返さない可能性のあるさまざまな HTTP コンシューマーコンポーネントが修正されました (rest-dsl を使用する場合など)。
- FluentProducerTemplate のスレッドセーフの問題が修正されました。

7.2. MAVEN 依存関係について

Maven BOM (Bill of Materials) ファイルの目的は、正常に動作する Maven 依存関係バージョンのセットを提供し、各 Maven アーティファクトに対して個別にバージョンを定義する必要をなくすことです。

Fuse が実行される各コンテナーには専用の BOM ファイルがあります。



注記

これらの BOM ファイルは https://github.com/jboss-fuse/redhat-fuse にあります。または、BOM ファイル更新の詳細は latest Release Notes を参照してください。

Fuse BOM には以下の利点があります。

- Maven 依存関係のバージョンを定義するため、依存関係を **pom.xml** ファイルに追加するときにバージョンを指定する必要がありません。
- 特定バージョンの Fuse に対して完全にテストされ、完全にサポートする依存関係のセットを定義します。
- Fuse のアップグレードを簡素化します。

重要

Fuse BOM によって定義される依存関係のセットのみが Red Hat によってサポートされます。

7.3. FUSE プロジェクトの MAVEN 依存関係の更新

Karaf の Fuse アプリケーションをアップグレードするには、プロジェクトの Maven 依存関係を更新します。

手順

1. プロジェクトの **pom.xml** ファイルを開きます。
2. 以下の例のように、プロジェクトの **pom.xml** ファイル (または、場合によっては親 **pom.xml** ファイル) に **dependencyManagement** 要素を追加します。

```
<?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. **pom.xml** ファイルを保存します。

BOM を **pom.xml** ファイルで依存関係として指定した後、アーティファクトのバージョンを **指定せずに** Maven 依存関係を **pom.xml** ファイルに追加できるようになります。たとえば、**camel-velocity** コンポーネントの依存関係を追加するには、以下の XML フラグメントを **pom.xml** ファイルの **dependencies** 要素に追加します。

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

この依存関係の定義では、**version** 要素が省略されることに注意してください。

第8章 KARAF での FUSE スタンドアロンのアップグレード

Fuse on Apache Karaf のアップグレードメカニズムを使用すると、更新バージョンの Fuse on Karaf を再インストールする必要なく、Apache Karaf コンテナーに修正を適用できます。アップグレードによりデプロイされたアプリケーションで問題が発生した場合に、アップグレードをロールバックすることもできます。

アップグレードインストーラーファイルは、Fuse on Apache Karaf をインストールのに使用するファイルと **同じもの** です。

注記

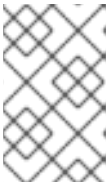
アップグレードのインストーラーファイルを取得するには、Red Hat カスタマーポータルの **Downloads** ページに移動し、Fuse on Apache Karaf のインストールアーカイブの最新バージョンをダウンロードします (例: **fuse-karaf-7.12.0.fuse-7_12_0-00019-redhat-00001.zip**)。

- 「Fuse on Karaf のアップグレードによる影響」
- 「Karaf での Fuse スタンドアロンのアップグレード」
- 「Fuse on Karaf のアップグレードのロールバック」

8.1. FUSE ON KARAF のアップグレードによる影響

アップグレードのメカニズムは、**バンドル JAR** および **静的ファイル** (たとえば **etc/** ディレクトリー下の設定ファイルなど) を含む、**すべての** インストールファイルへの更新を行うことができます。Fuse on Apache Karaf のアップグレードプロセス:

- バンドル JAR、設定ファイル、および静的ファイルを含むすべてのファイルを更新します。
- 現在のコンテナーインスタンス (および **data/** ディレクトリー下のランタイムストレージ) とベースのインストール両方にパッチを適用します。したがって、パッチはコンテナーインスタンスの削除後に維持されます。
- 機能リポジトリーファイルおよび機能自体を含む、Karaf 機能に関連するすべてのファイルを更新します。そのため、ロールアップパッチ後にインストールされる機能はすべて、パッチが適用された正しい依存関係を参照します。
- 必要な場合は、設定ファイル (例: **etc/** にあるファイル) を更新し、パッチによる設定変更に伴う設定変更を自動的にマージします。マージの競合が発生した場合は、処理方法の詳細についてパッチログを確認してください。
- マージの競合のほとんどは、自動的に解決されます。たとえば、パッチメカニズムはプロパティーファイルのプロパティーレベルで競合を検出します。プロパティーを変更したのがユーザーなのかパッチなのかを検出します。一方だけがプロパティーを変更している場合、変更が保持されます。
- (静的ファイルを含む) インストールに加えられた **すべての** 変更を追跡し、パッチをロールバックできるようにします。



注記

ロールアウトパッチメカニズムは、内部の git リポジトリー (**patches/.management/history** の場所にある) を使用して、変更を追跡します。

8.2. KARAF での FUSE スタンドアロンのアップグレード

以下の手順で、Fuse on Apache Karaf をアップグレードする方法を順を追って説明します。アップグレード手順を開始する前に、すべての前提条件が満たされていることを確認します。

前提条件

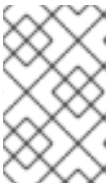
- アップグレードする前に、Fuse on Apache Karaf インストールの完全バックアップを作成している。
- コンテナーがまだ実行されていない場合は起動している。

ヒント

コンテナーがバックグラウンド (またはリモート) で実行されている場合は、SSH コンソールクライアント **bin/client** を使用してコンテナーに接続します。

- **patch:add** コマンドを呼び出して、アップグレードのインストーラーファイルをコンテナーの環境に追加します。たとえば、fuse-karaf-7.12.0.fuse-7_12_0-00019-redhat-00001.zip アップグレードインストーラーファイルを追加するには、次のようにします。

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



注記

この **patch:find** コマンドは、最新のホットフィックスパッチを探してコンテナーの環境に追加するためにのみ使用できます。完全なアップグレードパッチを適用するのに使用することはできません。

手順

1. **patch:update** コマンドを実行します。コンテナーを再起動する必要はありません。

```
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. **patch:list** コマンドを呼び出し、アップグレードインストーラーの一覧を表示します。このリストで、**[name]** 見出しの下にあるエントリーはアップグレード ID です。以下に例を示します。

```
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. 以下のように、**patch:simulate** コマンドを呼び出し、適用するアップグレードのアップグレード ID を指定して、アップグレードをシミュレートします。

```
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()>
```

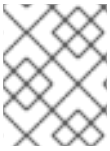
これにより、アップグレードの実行時にコンテナーに加えられる変更のログが生成されますが、実際にはコンテナーに何の変更も加えません。シミュレーションログを確認し、コンテナーに加えられる変更を確認します。

4. **patch:install** コマンドを呼び出し、適用するアップグレードのアップグレード ID を指定して、コンテナーをアップグレードします。以下に例を示します。

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

5. アップグレードアーティファクトのいずれかを検索して、アップグレードを検証します。たとえば、Fuse 7.1.0 を Fuse 7.2.0 にアップグレードしたばかりの場合、ビルド番号 7_11_0-00023-redhat-00001 のバンドルを次のように検索できます。

```
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
```



注記

アップグレード後にコンテナーを再起動すると、新しいバージョンとビルド番号が Welcome バナーにも表示されます。

8.3. FUSE ON KARAF のアップグレードのロールバック

アップグレードが機能しない場合や、コンテナーに新たな問題が発生することがあります。このような場合は、**patch:rollback** コマンドを使用して、簡単にアップグレードをロールバックし、システムを以前の状態に復元することができます。この一連の手順で、そのステップを順を追って説明します。

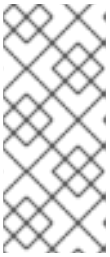
前提条件

- 最近 Fuse on Karaf がアップグレードされている。
- アップグレードをロールバックする必要がある。

手順

1. **patch:list** コマンドを呼び出し、最近インストールされたパッチのアップグレード ID **UPGRADE_ID** を取得します。
2. 以下のように **patch:rollback** コマンドを実行します。

```
patch:rollback UPGRADE_ID
```



注記

場合によっては、アップグレードをロールバックするためにコンテナーを再起動する必要があります。そのような場合には、コンテナーが自動的に再起動します。OSGi ランタイムの非常に動的な性質上、再起動時に互換性のないクラスに関連するエラーが表示される場合があります。これらのエラーは、起動または停止したばかりの OSGi サービスに関連しており、無視しても問題はありません。