Red Hat Fuse 7.9

移行ガイド

Red Hat Fuse 7.9 への移行

Last Updated: 2021-10-31

Red Hat Fuse 7.9 移行ガイド

Red Hat Fuse 7.9 への移行

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドでは、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つ情報を提供します。

目次

前書き

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、本ガイドの手順に従う前に、Red Hat Fuse 7.0 Migration Guide の手順に従ってください。

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、CTO である Chris Wright のメッセージ をご覧ください。

第1章 FUSE ONLINE のアップグレード

Fuse 7.8 のインストール方法に関係なく (インストールスクリプトまたは OperatorHub)、OpenShift OperatorHub を使用して Fuse 7.8 から 7.9 にアップグレードします。

Fuse 7.9 へのアップグレードで、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。変更が不要であっても、稼働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。

1.1. OPERATORHUB を使用した FUSE ONLINE のアップグレード

Fuse Online 7.8 のインストール方法に関係なく (インストールスクリプトまたは OperatorHub)、OpenShift OperatorHub を使用して Fuse Online 7.8 から 7.9 にアップグレードします。

Fuse Online 7.9 にアップグレードする手順は、既存の Fuse Online をインストールした方法によって異なります。

- OperatorHub を使用して Fuse Online 7.8 または以前の 7.9 バージョンがインストールしている。
- インストールスクリプトを使用して Fuse Online 7.8 をインストールしている。

注記: Fuse Online 7.9 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は、Fuse Online 7.9 にアップグレードする場合、OCP 4.6 以降にアップグレードする必要があります。

以前の Fuse Online OperatorHub インストールからのアップグレード

Fuse Online 7.8 または以前の 7.9 バージョンから新しい Fuse Online 7.9 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 Online 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 Online 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。

Fuse Online 7.8 インストールスクリプトインストールからのアップグレード

インストールスクリプトを使用して Fuse Online 7.8 をインストールした場合、以下の手順のステップに従って OperatorHub を使用して Fuse Online 7.9 にアップグレードします。

前提条件

- 既存の Fuse Online 7.8 インストールスクリプトのインストールには、Operator、オペランド、および syndesis カスタムリソースが含まれます。

手順

Fuse Online Operator を使用してアップグレードするには、以下を行います。

1. OpenShift Web コンソールで、**Operators** > **OperatorHub** をクリックします。
2. Fuse Online 7.9 Operator を選択してから **Install** をクリックします。
3. Fuse Online 7.8 のインストールを含む namespace を選択し、**Install** をクリックします。
4. アップグレードに成功したら、既存のインテグレーションをすべて再パブリッシュする必要があります。

1.2. インストールスクリプトを使用した FUSE ONLINE の以前のバージョンのアップグレード

インストールスクリプトを使用して Fuse Online 7.8 をインストールした場合、「Upgrading Fuse Online by using the OperatorHub」の説明に従って、 OperatorHub を使用してバージョン 7.9 にアップグレードします。

インストールスクリプトを使用して、バージョン 7.7 およびそれ以前のバージョンの Fuse Online をインストールした場合:

- OCP で Fuse Online バージョン 7.7 を実行している場合は、7.8 にアップグレード してから 7.9 にアップグレードする必要があります。
- OCP で Fuse Online バージョン 7.6 を実行している場合は、7.7 にアップグレード してから 7.8 にアップグレードする必要があります。
- OCP で Fuse Online バージョン 7.5 を実行している場合は、7.6 にアップグレード してから 7.7 にアップグレードする必要があります。
- OCP で Fuse Online バージョン 7.4 を実行している場合は、7.5 にアップグレード してから 7.6 にアップグレードする必要があります。
- OCP で Fuse Online バージョン 7.3 を実行している場合は、7.4 にアップグレード してから 7.5 にアップグレードする必要があります。
- OCP で Fuse Online バージョン 7.2 を実行している場合は、7.3 にアップグレード する必要があります。
- OCP で Fuse Online バージョン 7.1 を実行している場合は、7.2 にアップグレード する必要があります。

1.3. FUSE ONLINE インテグレーションのアップグレード

Fuse Online 7.9 にアップグレードする場合、既存のインテグレーションに変更を加える必要があるかどうかを判別する必要があります。

「Camel Migration Considerations」に記載の Apache Camel の更新を確認します。

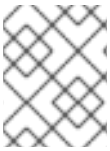
インフラストラクチャーのアップグレード中およびアップグレード後に、既存のインテグレーションは古いバージョンの Fuse Online ライブラリーおよび依存関係で実行されるため、インテグレーションを変更する必要がない場合でも、稼働中のインテグレーションを再パブリッシュする必要があります。更新されたバージョンで実行するには、再パブリッシュする必要があります。

手順

インテグレーションを再パブリッシュするには、Fuse Online 環境で以下を行います。

1. Fuse Online の左側のナビゲーションパネルで **Integrations** をクリックします。
2. インテグレーションごとに以下を行います。
   a. インテグレーションエントリーの右側で  **をクリックし、Edit** を選択します。
   b. Fuse Online で編集するインテグレーションが表示されたら、右上の **Publish** をクリックします。

パブリッシュを行うと、最新の Fuse Online 依存関係を使用して再ビルドが強制されます。



注記

インテグレーションの要素に、更新が必要な新しい依存関係がある場合、Fuse Online ユーザーインターフェースに警告が表示されます。

1.3.1. Camel の移行に関する考慮事項

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際、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 の更新

- Camel が 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 へアップグレードされました。Camel は Spring 4.3.x でも動作しますが、今後のリリースでは Spring 5.x が Spring の最小バージョンになります。
- Karaf 4.2 へのアップグレードKaraf 4.1 で Camel を実行することはできますが、本リリースでは、Karaf 4.2 のみを公式にサポートしています。
- toD DSL を使用して最適化され、可能な場合は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。
- 読み取りロック idempotent/idempotent-changed を持つ File2 コンシューマーは、ファイルが処理中と見なされる際にリリースタスクを遅らせてウィンドウを拡張するように構成できるようになりました。これは、共有のべき等リポジトリーでアクティブ/アクティブクラスター設定で使用でき、他のノードが処理済みファイルを処理可能なファイルとしてすばやく認識しないようにします (readLockRemoveOnCommit=true がある場合にのみ必要です)。
- 要求/応答モードで Netty4 プロデューサーにカスタム要求/応答相関 ID マネージャーの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て、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 プロデューサーは、endpointProperties を使用して REST 設定での設定をサポートするようになりました。
- Kafka コンポーネントは、Camel メッセージと Kafka メッセージ間のヘッダーマッピングを制御するためのカスタム実装をプラグインする 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 ヘッダーが可能である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、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) へのルートを登録できる Service Registry SPI があります。
- SEDA コンポーネントのデフォルトのキューサイズが無制限ではなく 1000 になりました。
- 以下の重要な問題が修正されました。
  - camel-cxf コンシューマーで CXF 継続タイムアウトの問題が修正されました。これは、コンシューマーが呼び出しの SOAP クライアントのタイムアウトをトリガーするのではなく、データで応答を返す原因となっていました。
  - 堅牢な一方向操作を使用しているときに、camel-cxf コンシューマーが UoW を解放しない問題を修正しました。
  - AdviceWith の使用と onException などでの weave メソッドの使用が機能しない問題を修正しました。
  - 最初に呼び出された next() メソッド呼び出しでイテレーターが例外をスローすると、メッセージ本文を反復しているときに、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のある問題を修正しました。
  - autoCommitEnable=false の場合に、Kafka コンシューマーが自動コミットしないように修正しました。
  - ファイルコンシューマーが、none であるのはずの markerFile をデフォルトで読み取りロックとして使用していた問題が修正されました。
  - 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 (最初は -1) を提供する問題を修正しました。
  - Java DSL のコンテンツベースルーターが述語時におけるプロパティープレースホルダーを解決しない場合の問題が修正されました。

Camel 2.23 の更新

- Spring Boot 2.1 へのアップグレード
- spring-boot 自動設定を使用して、追加のコンポーネントレベルのオプションを設定できるようになりました。これらのオプションは、ツール支援の 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、および言語のすべての Spring Boot 自動設定オプションを含むドキュメントセクションを追加しました。
- 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、動的な式に基づく相関グループをサポートするようになったため、メッセージを別の throttled セットにグループ化できるようになりました。
- Hystrix EIP では、Camel のエラーハンドラーへの継承が可能になり、再配信でエラー処理を有効にしている場合に Hystrix EIP ブロック全体を再試行できるようになりました。

- SQL および ElSql コンシューマーは、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、単純な式を使用した Bean の呼び出しに限定されていることに注意してください。
- swagger-restdsl maven プラグインは、Swagger 仕様ファイルからの DTO モデルクラスの生成をサポートするようになりました。
- 以下の重要な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達しないように修正されているため、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化された場合に Tracer が機能しない問題を修正しました。
  - XML ドキュメントの組み込み型コンバーターは、解析エラーを stdout に出力する場合がありますが、これはロギング API を使用して出力するように修正されました。
  - メッセージ本文がストリーミングベースの場合、charset オプションを使用した SFTP 書き込みファイルが機能しない問題を修正しました。
  - 複数のルートをルーティングしてそれらを単一の親スパンにグループ化するときに、Zipkin ルート ID が再利用されないように修正しました。
  - ホスト名に数字の IP アドレスが含まれている場合に、バグがあった HTTP エンドポイントを使用するときに最適化された toD を修正しました。
  - 一時キューを介した要求/応答と手動確認モードの使用に関する RabbitMQ の問題を修正しました。(要求/応答を可能にするために必要な) 一時キューを確認しません。
  - OPTIONS リクエストの Allow ヘッダーで許可されたすべての HTTP 動詞を返さない可能性があるさまざまな HTTP コンシューマーコンポーネントを修正しました (rest-dsl を使用する場合など)。
  - FluentProducerTemplate のスレッドセーフティーの問題を修正しました。

第2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。

2.1. 作業を始める前に

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。

- 最新の 1.5.x バージョンへのアップグレード
  - 開始する前に、最新の 1.5.x の利用可能なバージョンにアップグレードしてください。これは、その行の最新の依存関係に対して確実にビルドするためです。
- 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 の依存関係管理で 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性のある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は、アップグレード時に確認が必要となる場合があります。これを標準の自動設定を使用して置き換えることができる場合は、アップグレードする前に行ってください。
- 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。

2.2. 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、適切に **application.properties/application.yml** を更新する必要があります。新しい **spring-boot-properties-migrator** モジュールを利用して、更新することができます。プロジェクトに依存関係として追加すると、アプリケーションの環境を分析し、起動時に診断を出力するだけでなく、実行時に一時的にプロパティーを移行します。

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必ず削除してください。

第3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 の移行に関する考慮事項」 の説明に従って、Apache Camel の更新を考慮する必要があります。
- Fuse プロジェクトの Maven 依存関係を更新して、正しいバージョンの Fuse を使用していることを確認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ビルドします。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のデフォルトの動作では、Maven は外部リポジトリーを検索し、必要なアーティファクトをダウンロードします。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 プロジェクトでそれらを更新する方法について説明します。

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

3.1. CAMEL の移行に関する考慮事項

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際、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 の更新

- Camel が 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 へアップグレードされました。Camel は Spring 4.3.x でも動作しますが、今後のリリースでは Spring 5.x が Spring の最小バージョンになります。
- Karaf 4.2 へのアップグレードKaraf 4.1 で Camel を実行することはできますが、本リリースでは、Karaf 4.2 のみを公式にサポートしています。
- toD DSL を使用して最適化され、可能な場合は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。
- 読み取りロック idempotent/idempotent-changed を持つ File2 コンシューマーは、ファイルが処理中と見なされる際にリリースタスクを遅らせてウィンドウを拡張するように構成できるようになりました。これは、共有のべき等リポジトリーでアクティブ/アクティブクラスター設定で使用でき、他のノードが処理済みファイルを処理可能なファイルとしてすばやく認識しないようにします (readLockRemoveOnCommit=true がある場合にのみ必要です)。
- 要求/応答モードで Netty4 プロデューサーにカスタム要求/応答相関 ID マネージャーの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て、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 プロデューサーは、endpointProperties を使用して REST 設定での設定をサポートするようになりました。
- Kafka コンポーネントは、Camel メッセージと Kafka メッセージ間のヘッダーマッピングを制御するためのカスタム実装をプラグインする 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 ヘッダーが可能である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、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) へのルートを登録できる Service Registry SPI があります。
- SEDA コンポーネントのデフォルトのキューサイズが無制限ではなく 1000 になりました。
- 以下の重要な問題が修正されました。
  - camel-cxf コンシューマーで CXF 継続タイムアウトの問題が修正されました。これは、コンシューマーが呼び出しの SOAP クライアントのタイムアウトをトリガーするのではなく、データで応答を返す原因となっていました。
  - 堅牢な一方向操作を使用しているときに、camel-cxf コンシューマーが UoW を解放しない問題を修正しました。
  - AdviceWith の使用と onException などでの weave メソッドの使用が機能しない問題を修正しました。
  - 最初に呼び出された next() メソッド呼び出しでイテレーターが例外をスローすると、メッセージ本文を反復しているときに、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のある問題を修正しました。
  - autoCommitEnable=false の場合に、Kafka コンシューマーが自動コミットしないように修正しました。
  - ファイルコンシューマーが、none であるのはずの markerFile をデフォルトで読み取りロックとして使用していた問題が修正されました。
  - 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 (最初は -1) を提供する問題を修正しました。
  - Java DSL のコンテンツベースルーターが述語時におけるプロパティープレースホルダーを解決しない場合の問題が修正されました。

Camel 2.23 の更新

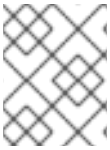
- Spring Boot 2.1 へのアップグレード
- spring-boot 自動設定を使用して、追加のコンポーネントレベルのオプションを設定できるようになりました。これらのオプションは、ツール支援の spring-boot コンポーネントメタデータ JSON ファイル記述子に含まれます。
- すべてのコンポーネント、データ形式、および言語のすべての Spring Boot 自動設定オプションを含むドキュメントセクションを追加しました。
- すべての Camel Spring Boot スターター JAR には、Spring Boot の自動設定を最適化するために JAR に **META-INF/spring-autoconfigure-metadata.properties** ファイルが追加されるようになりました。

- Throttler は、動的な式に基づく相関グループをサポートするようになったため、メッセージを別の throttled セットにグループ化できるようになりました。
- Hystrix EIP では、Camel のエラーハンドラーへの継承が可能になり、再配信でエラー処理を有効にしている場合に Hystrix EIP ブロック全体を再試行できるようになりました。
- SQL および ElSql コンシューマーは、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、単純な式を使用した Bean の呼び出しに限定されていることに注意してください。
- swagger-restdsl maven プラグインは、Swagger 仕様ファイルからの DTO モデルクラスの生成をサポートするようになりました。
- 以下の重要な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達しないように修正されているため、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化された場合に Tracer が機能しない問題を修正しました。
  - XML ドキュメントの組み込み型コンバーターは、解析エラーを stdout に出力する場合がありますが、これはロギング API を使用して出力するように修正されました。
  - メッセージ本文がストリーミングベースの場合、charset オプションを使用した SFTP 書き込みファイルが機能しない問題を修正しました。
  - 複数のルートをルーティングしてそれらを単一の親スパンにグループ化するときに、Zipkin ルート ID が再利用されないように修正しました。
  - ホスト名に数字の IP アドレスが含まれている場合に、バグがあった HTTP エンドポイントを使用するときに最適化された toD を修正しました。
  - 一時キューを介した要求/応答と手動確認モードの使用に関する RabbitMQ の問題を修正しました。(要求/応答を可能にするために必要な) 一時キューを確認しません。
  - OPTIONS リクエストの Allow ヘッダーで許可されたすべての HTTP 動詞を返さない可能性があるさまざまな HTTP コンシューマーコンポーネントを修正しました (rest-dsl を使用する場合など)。
  - FluentProducerTemplate のスレッドセーフティーの問題を修正しました。

3.2. MAVEN の依存関係について

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 を実行す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 最新のリリースノート 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** 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。

3.3. 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>7.9.0.fuse-sb2-790065-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>
```

注記

Spring Boot の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>7.9.0.fuse-sb2-790065-redhat-00001</fuse.version>
 <spring-boot.version>2.3.9.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4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 の移行に関する考慮事項」 の説明に従って、Apache Camel の更新を考慮する必要があります。
- Fuse プロジェクトの Maven 依存関係を更新して、正しいバージョンの Fuse を使用していることを確認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ビルドします。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のデフォルトの動作では、Maven は外部リポジトリーを検索し、必要なアーティファクトをダウンロードします。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 プロジェクトでそれらを更新する方法について説明します。

  - 「Maven の依存関係について」
  - 「Fuse プロジェクトの Maven 依存関係の更新」
- 「Java EE 依存関係のアップグレード」 の説明に従って、アップグレードしたバージョンの Java EE 依存関係を使用していることを確認するには、Fuse プロジェクトの Maven 依存関係を更新する必要があります。

4.1. CAMEL の移行に関する考慮事項

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際、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 の更新

- Camel が 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 へアップグレードされました。Camel は Spring 4.3.x でも動作しますが、今後のリリースでは Spring 5.x が Spring の最小バージョンになります。
- Karaf 4.2 へのアップグレードKaraf 4.1 で Camel を実行することはできますが、本リリースでは、Karaf 4.2 のみを公式にサポートしています。
- toD DSL を使用して最適化され、可能な場合は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。
- 読み取りロック idempotent/idempotent-changed を持つ File2 コンシューマーは、ファイルが処理中と見なされる際にリリースタスクを遅らせてウィンドウを拡張するように構成できるようになりました。これは、共有のべき等リポジトリーでアクティブ/アクティブクラスター設定で使用でき、他のノードが処理済みファイルを処理可能なファイルとしてすばやく認識しないようにします (readLockRemoveOnCommit=true がある場合にのみ必要です)。

- 要求/応答モードで Netty4 プロデューサーにカスタム要求/応答相関 ID マネージャーの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て、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 プロデューサーは、endpointProperties を使用して REST 設定での設定をサポートするようになりました。
- Kafka コンポーネントは、Camel メッセージと Kafka メッセージ間のヘッダーマッピングを制御するためのカスタム実装をプラグインする 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 ヘッダーが可能である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、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) へのルートを登録できる Service Registry SPI があります。
- SEDA コンポーネントのデフォルトのキューサイズが無制限ではなく 1000 になりました。
- 以下の重要な問題が修正されました。
  - camel-cxf コンシューマーで CXF 継続タイムアウトの問題が修正されました。これは、コンシューマーが呼び出しの SOAP クライアントのタイムアウトをトリガーするのではなく、データで応答を返す原因となっていました。
  - 堅牢な一方向操作を使用しているときに、camel-cxf コンシューマーが UoW を解放しない問題を修正しました。
  - AdviceWith の使用と onException などでの weave メソッドの使用が機能しない問題を修正しました。
  - 最初に呼び出された next() メソッド呼び出しでイテレーターが例外をスローすると、メッセージ本文を反復しているときに、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のある問題を修正しました。
  - autoCommitEnable=false の場合に、Kafka コンシューマーが自動コミットしないように修正しました。
  - ファイルコンシューマーが、none であるのはずの markerFile をデフォルトで読み取りロックとして使用していた問題が修正されました。
  - 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 (最初は -1) を提供する問題を修正しました。
  - Java DSL のコンテンツベースルーターが述語時におけるプロパティープレースホルダーを解決しない場合の問題が修正されました。

Camel 2.23 の更新

- Spring Boot 2.1 へのアップグレード
- spring-boot 自動設定を使用して、追加のコンポーネントレベルのオプションを設定できるようになりました。これらのオプションは、ツール支援の 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、および言語のすべての Spring Boot 自動設定オプションを含むドキュメントセクションを追加しました。

- 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、動的な式に基づく相関グループをサポートするようになったため、メッセージを別の throttled セットにグループ化できるようになりました。
- Hystrix EIP では、Camel のエラーハンドラーへの継承が可能になり、再配信でエラー処理を有効にしている場合に Hystrix EIP ブロック全体を再試行できるようになりました。
- SQL および ElSql コンシューマーは、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、単純な式を使用した Bean の呼び出しに限定されていることに注意してください。
- swagger-restdsl maven プラグインは、Swagger 仕様ファイルからの DTO モデルクラスの生成をサポートするようになりました。
- 以下の重要な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達しないように修正されているため、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化された場合に Tracer が機能しない問題を修正しました。
  - XML ドキュメントの組み込み型コンバーターは、解析エラーを stdout に出力する場合がありますが、これはロギング API を使用して出力するように修正されました。
  - メッセージ本文がストリーミングベースの場合、charset オプションを使用した SFTP 書き込みファイルが機能しない問題を修正しました。
  - 複数のルートをルーティングしてそれらを単一の親スパンにグループ化するときに、Zipkin ルート ID が再利用されないように修正しました。
  - ホスト名に数字の IP アドレスが含まれている場合に、バグがあった HTTP エンドポイントを使用するときに最適化された toD を修正しました。
  - 一時キューを介した要求/応答と手動確認モードの使用に関する RabbitMQ の問題を修正しました。(要求/応答を可能にするために必要な) 一時キューを確認しません。
  - OPTIONS リクエストの Allow ヘッダーで許可されたすべての HTTP 動詞を返さない可能性があるさまざまな HTTP コンシューマーコンポーネントを修正しました (rest-dsl を使用する場合など)。
  - FluentProducerTemplate のスレッドセーフティーの問題を修正しました。

4.2. MAVEN の依存関係について

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。

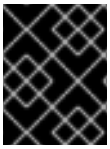
Fuse を実行する各コンテナーには専用の BOM ファイルがあります。

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 最新のリリースノート 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** 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。

4.3. 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>7.9.0.fuse-sb2-790065-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. **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。

4.4. JAVA EE 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>
```

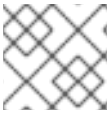
```
<dependency>
  <groupId>jakarta.inject</groupId>
  <artifactId>jakarta.inject-api</artifactId>
</dependency>
```

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

以下の手順では、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. JBoss EAP のマイナーリリースを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』の手順に従ってください。
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 アプリケーションを再コンパイルまたは再デプロイする必要はありません。

4.6. FUSE と JBOSS EAP の同時アップグレード

以下の手順では、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について説明します。たとえば、JBoss EAP 7.2 上の Fuse 7.7 から JBoss EAP 7.3 上の Fuse 7.8 に移行する場合などです。

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 ランタイムの両方の新規インストールを実行することを推奨しています。

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5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 の移行に関する考慮事項」 の説明に従って、Apache Camel の更新を考慮する必要があります。
- Fuse プロジェクトの Maven 依存関係を更新して、正しいバージョンの Fuse を使用していることを確認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ビルドします。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のデフォルトの動作では、Maven は外部リポジトリーを検索し、必要なアーティファクトをダウンロードします。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 プロジェクトでそれらを更新する方法について説明します。

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

5.1. CAMEL の移行に関する考慮事項

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際、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 の更新

- Camel が 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 へアップグレードされました。Camel は Spring 4.3.x でも動作しますが、今後のリリースでは Spring 5.x が Spring の最小バージョンになります。
- Karaf 4.2 へのアップグレードKaraf 4.1 で Camel を実行することはできますが、本リリースでは、Karaf 4.2 のみを公式にサポートしています。
- toD DSL を使用して最適化され、可能な場合は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。
- 読み取りロック idempotent/idempotent-changed を持つ File2 コンシューマーは、ファイルが処理中と見なされる際にリリースタスクを遅らせてウィンドウを拡張するように構成できるようになりました。これは、共有のべき等リポジトリーでアクティブ/アクティブクラスター設定で使用でき、他のノードが処理済みファイルを処理可能なファイルとしてすばやく認識しないようにします (readLockRemoveOnCommit=true がある場合にのみ必要です)。
- 要求/応答モードで Netty4 プロデューサーにカスタム要求/応答相関 ID マネージャーの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て、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 プロデューサーは、endpointProperties を使用して REST 設定での設定をサポートするようになりました。
- Kafka コンポーネントは、Camel メッセージと Kafka メッセージ間のヘッダーマッピングを制御するためのカスタム実装をプラグインする 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 ヘッダーが可能である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、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) へのルートを登録できる Service Registry SPI があります。
- SEDA コンポーネントのデフォルトのキューサイズが無制限ではなく 1000 になりました。
- 以下の重要な問題が修正されました。
  - camel-cxf コンシューマーで CXF 継続タイムアウトの問題が修正されました。これは、コンシューマーが呼び出しの SOAP クライアントのタイムアウトをトリガーするのではなく、データで応答を返す原因となっていました。
  - 堅牢な一方向操作を使用しているときに、camel-cxf コンシューマーが UoW を解放しない問題を修正しました。
  - AdviceWith の使用と onException などでの weave メソッドの使用が機能しない問題を修正しました。
  - 最初に呼び出された next() メソッド呼び出しでイテレーターが例外をスローすると、メッセージ本文を反復しているときに、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のある問題を修正しました。
  - autoCommitEnable=false の場合に、Kafka コンシューマーが自動コミットしないように修正しました。
  - ファイルコンシューマーが、none であるのはずの markerFile をデフォルトで読み取りロックとして使用していた問題が修正されました。
  - 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 (最初は -1) を提供する問題を修正しました。
  - Java DSL のコンテンツベースルーターが述語時におけるプロパティープレースホルダーを解決しない場合の問題が修正されました。

Camel 2.23 の更新

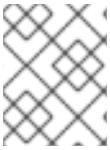
- Spring Boot 2.1 へのアップグレード
- spring-boot 自動設定を使用して、追加のコンポーネントレベルのオプションを設定できるようになりました。これらのオプションは、ツール支援の spring-boot コンポーネントメタデータ JSON ファイル記述子に含まれます。
- すべてのコンポーネント、データ形式、および言語のすべての Spring Boot 自動設定オプションを含むドキュメントセクションを追加しました。
- すべての Camel Spring Boot スターター JAR には、Spring Boot の自動設定を最適化するために JAR に **META-INF/spring-autoconfigure-metadata.properties** ファイルが追加されるようになりました。

- Throttler は、動的な式に基づく相関グループをサポートするようになったため、メッセージを別の throttled セットにグループ化できるようになりました。
- Hystrix EIP では、Camel のエラーハンドラーへの継承が可能になり、再配信でエラー処理を有効にしている場合に Hystrix EIP ブロック全体を再試行できるようになりました。
- SQL および ElSql コンシューマーは、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、単純な式を使用した Bean の呼び出しに限定されていることに注意してください。
- swagger-restdsl maven プラグインは、Swagger 仕様ファイルからの DTO モデルクラスの生成をサポートするようになりました。
- 以下の重要な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達しないように修正されているため、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化された場合に Tracer が機能しない問題を修正しました。
  - XML ドキュメントの組み込み型コンバーターは、解析エラーを stdout に出力する場合がありますが、これはロギング API を使用して出力するように修正されました。
  - メッセージ本文がストリーミングベースの場合、charset オプションを使用した SFTP 書き込みファイルが機能しない問題を修正しました。
  - 複数のルートをルーティングしてそれらを単一の親スパンにグループ化するときに、Zipkin ルート ID が再利用されないように修正しました。
  - ホスト名に数字の IP アドレスが含まれている場合に、バグがあった HTTP エンドポイントを使用するときに最適化された toD を修正しました。
  - 一時キューを介した要求/応答と手動確認モードの使用に関する RabbitMQ の問題を修正しました。(要求/応答を可能にするために必要な) 一時キューを確認しません。
  - OPTIONS リクエストの Allow ヘッダーで許可されたすべての HTTP 動詞を返さない可能性があるさまざまな HTTP コンシューマーコンポーネントを修正しました (rest-dsl を使用する場合など)。
  - FluentProducerTemplate のスレッドセーフティーの問題を修正しました。

5.2. MAVEN の依存関係について

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 を実行す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 最新のリリースノート 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** 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

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>7.9.0.fuse-sb2-790065-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. **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。

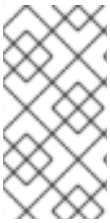
```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

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 KARAF の FUSE スタンドアロンのアップグレード

Fuse on Apache Karaf アップグレードメカニズムにより、更新されたバージョンの Fuse on Karaf を再インストールしなくても、修正を Apache Karaf コンテナーに適用できます。また、アップグレードが原因で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。

アップグレードのインストーラーファイルは、Fuse on Apache Karaf をインストールする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 の最新バージョンの Fuse on Apache Karaf 用インストールアーカイブをダウンロードします (例: **fuse-karaf-7.9.0.fuse-790071-redhat-00001.zip**)。

- 「Fuse on Karaf のアップグレードの影響」
- 「Karaf 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」

6.1. FUSE ON KARAF のアップグレードの影響

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、静的ファイルなどのファイルが更新されます。
- 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。そのため、コンテナーインスタンスの削除後もパッチは保持されます。
- 機能リポジトリーファイルや機能自体など、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、ロールパッチ後にインストールされるすべての機能は、正しいパッチが適用された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、パッチログで、その処理方法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。これは、プロパティーを変更したのがユーザーかパッチかを検出します。変更は、1 つの側のみがプロパティーが変更した場合に保持されます。
- インストールに追加された **すべての** 変更を追跡し (静的ファイルを含む)、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。

6.2. KARAF 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について説明します。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完了し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、必ず Fuse on Apache Karaf インストールの完全なバックアップを作成します。
- コンテナーが稼働していない場合は起動します。

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-7.9.0.fuse-790071-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、以下のコマンドを実行します。

```
patch:add file:///path/to/fuse-karaf-7.9.0.fuse-790071-redhat-00001.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施時にコンテナーに加えられた変更のログが生成されますが、コンテナーに実際の変更は加えられません。シミュレーションログを確認し、コンテナーに加える変更を把握します。

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-7.9.0.fuse-790071-redhat-00001
```

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ならば、以下のようにビルド番号 790071 でバンドルを検索できます。

```
karaf@root()> bundle:list -l | grep 790071
 22 │ Active │  80 │ 7.9.0.fuse-790071-redhat-00001  │
mvn:org.jboss.fuse.modules/fuse-pax-transx-tm-narayana/7.9.0.fuse-790071-redhat-
00001
188 │ Active │  80 │ 7.9.0.fuse-790071-redhat-00001  │
mvn:org.jboss.fuse.modules.patch/patch-commands/7.9.0.fuse-790071-redhat-00001
```

注記

アップグレード後、コンテナーを再起動すると、ウェルカムバナーに新しいバージョンとビルド番号も表示されます。

6.3. FUSE ON KARAF のアップグレードのロールバック

場合によっては、アップグレードが機能しなかったり、コンテナーに新しい問題が発生したり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。以下の一連の手順で、その方法を説明します。

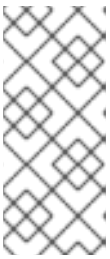
前提条件

- 最近 Fuse on Karaf をアップグレードしている。
- 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** コマンドを呼び出します。

```
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のに、コンテナーを再起動する必要があります。この場合、コンテナーが自動的に再起動されます。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発生する可能性があります。これらのエラーは、開始または停止した OSGi サービスに関連し、無視しても問題はありません。