# Red Hat Fuse 7.8

## 移行ガイド

Red Hat Fuse 7.8 への移行

Last Updated: 2023-07-25

# Red Hat Fuse 7.8 移行ガイド

Red Hat Fuse 7.8 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

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。

# 第1章 FUSE ONLINE のアップグレード

Fuse Online のアップグレードプロセスは、Fuse Online が Red Hat OpenShift Online または OpenShift Container Platform (OCP) にインストールされているかによって異なります。

- **OpenShift Online**: Fuse 7.8 のリリース時に、OpenShift Online の Fuse Online インフラストラクチャーは自動的にアップグレードされます。OpenShift Online で実行されている Fuse Online インテグレーションのアップグレード で説明されているように、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。

- **OCP** - オンサイトの OpenShift Container Platform で実行されている Fuse Online 環境をアップグレードするには、アップグレード前の Docker イメージへのアクセス の説明にしたがって、まず Fuse バックアップおよびリストアデータベースの Docker イメージ へのアクセスを設定する必要があります。
  Fuse のアップグレード手順は、Fuse のインストール方法によって異なります。

  - OperatorHub を使用して Fuse をインストールした場合は、Upgrading Fuse by using the OperatorHub を参照してください。

  - インストールスクリプトを使用して Fuse をインストールした場合は、インストールスクリプトを 使用した Fuse のアップグレード を 参照してください。

  インテグレーションをアップグレードするには、Camel 移行に関する考慮事項 で説明されているように、Apache Camel の更新も考慮 する必要があります。

## 1.1. アップグレード前の DOCKER イメージへのアクセス

デフォルトでは、Fuse Online のアップグレードプロセスでは、**registry.redhat.io** からプルするのではなく、Fuse Online バックアップおよびリストアデータベースの Docker イメージを **docker.io** レジストリーからプルします（既知の問題 https://issues.redhat.com/browse/ENTESB-15364を参照）。**docker.io** は無料ダウンロードにサービス制限を適用するため、バックアップおよび復元データベースイメージをダウンロードしようとすると、Fuse Online のアップグレードプロセスで以下のエラーが発生する可能性があります。

```
error: code = Unknown desc = toomanyrequests: You have reached your pull rate limit
```

Docker Hub のプルレート制限に関する詳細は、Docker のドキュメント (https://www.docker.com/increase-rate-limits) を参照してください。

Docker の制限によるエラーが発生しないようにするには、Fuse Online のアップグレードを開始する前に Docker イメージにアクセスできることを確認してください。Docker イメージにアクセスする方法は、お使いの OpenShift Container Platform (OCP) バージョンによって異なります。

- OCP 4.6 の場合は、OCP 4.6 での Docker イメージへのアクセス の説明にしたがって、Docker Hub クレデンシャルを既存の syndesis pull secret に追加します。Docker クレデンシャルを syndesis pull secret に追加すると、Docker Hub のアカウントタイプに応じて、API に対する Docker Hub の制限が引き上げられます。

- OCP 3.11 の場合は、OCP 3.11 での Docker イメージへのアクセス で説明されているように、Fuse Online バックアップおよびリストアデータベースイメージを **docker.io** からローカルキャッシュに プルします。

### 1.1.1. OCP 4.6 での Docker イメージへのアクセス

Docker の制限エラーが発生しないようにするため、Fuse Online のアップグレードを開始する前に、Docker クレデンシャルを syndesis pull secret に追加します。

### 前提条件

OpenShift クラスターの管理者権限を持っている必要があります。

### 手順

Docker Hub クレデンシャルを既存の syndesis pull secret に追加するには、以下を行います。

1. 管理者のクレデンシャルで OpenShift にログインします。

```
oc login -u system:admin
```

2. 以下のコマンドを実行して、syndesis pull secret を取得します。

```
 oc get secrets syndesis-pull-secret -o=custom-columns=SECRET:.data.* --no-headers |
base64 -d | jq
```

このコマンドを実行すると、以下のような出力が返されます。ここで <AUTH> は base64 でエンコードされたユーザー名とパスワードに置き換えられます。

```
 {
   "auths":{
     "registry.redhat.io":{
       "username":"<AUTH>",
       "password":"<AUTH>",
       "auth":"<AUTH>"
     },
     <You can have more auths elements here>
  }
 }
```

3. Docker Hub レジストリークレデンシャルを追加するには、pull secret を編集し、PULL_SECRET 変数に保存します。

```
 PULL_SECRET=$(base64 -w 0 <<EOF
{
  "auths":{
    "registry.redhat.io":{
      "username":"<AUTH>",
      "password":"<AUTH>",
      "auth":"<AUTH>"
    },
    <You can have more auths elements here>
    },
    "https://index.docker.io/v1/": {
      "auth": "<AUTH>"
    }
  }
}
EOF
)
```

4. syndesis pull secret にパッチを適用するには、以下のコマンドを実行します。

```
oc apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: syndesis-pull-secret
data:
  .dockerconfigjson: $PULL_SECRET
type: kubernetes.io/dockerconfigjson
EOF
```

**次のステップ**

- OperatorHub を使用して Fuse Online をインストールした場合は、OperatorHub を使用した Fuse Online のアップグレード を参照してください。
- インストールスクリプトを使用して Fuse Online をインストールした場合は、インストールスクリプトを使用した Fuse Online のアップグレード を参照してください。

### 1.1.2. OCP 3.11 での Docker イメージへのアクセス

Docker の制限エラーが発生しないようにするため、OCP 3.11 で Fuse Online のアップグレードプロセスを開始する前に、必要なイメージを **docker.io** からローカルキャッシュにプルする必要があります。

**前提条件**

OpenShift クラスターの管理者権限を持っている必要があります。

**手順**

1. OpenShift ユーザー名と OpenShift クラスター URL を使用して OpenShift ノードにログインします。

   ```
   ssh login <user>@node1.<clusterUrl>
   ```

2. ユーザー名とパスワードを使用して docker にログインします。

   ```
   sudo docker login -u <username> -p <password> docker.io
   ```

3. PostgreSQL をプルする以下のコマンドを実行します。

   ```
   sudo docker pull crunchydata/crunchy-pgdump:centos7-10.11-4.2.1
   sudo docker pull crunchydata/crunchy-pgrestore:centos7-10.11-4.2.1
   sudo docker pull centos:7
   ```

   PostgreSQL イメージは、OpenShift ノードの内部 docker レジストリーにプルされます。

4. 各 OpenShift ノードに、前述の 3 つのステップを実行します。

**次のステップ**

インストールスクリプトを使用した Fuse Online のアップグレード の手順にしたがって、Fuse Online をアップグレードします。

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

Fuse Online 7.8 は OCP 4.6 より OperatorHub で利用できます。OCP 4.5 を使用している場合は、OCP 4.6 にアップグレードしてから Fuse Online 7.8 をインストールする必要があります。

**注記:** インストールスクリプトを使用して Fuse Online 7.7 をインストールした場合、インストールスクリプトを使用した Fuse Online のアップグレード の説明にしたがって、インストールスクリプトを使用して Fuse Online 7.8 にアップグレードする必要があります。

Fuse Online のインストール時に、**fuse-online-v7.n** 形式でチャネルを指定します。**n** は現在のリリース番号に置き換えます。たとえば、Fuse Online 7.8 の場合、チャネルは **fuse-online-v7.8** です。

Fuse Online 7.8 の古いバージョンから Fuse Online 7.8 の新しいバージョンへのアップグレードは、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。

OCP 4.6 以降で、Fuse Online 7.7 を 7.8 にアップグレードする場合は、以下の手順にしたがいます。

**手順**

1. Docker の制限エラーが発生しないようにするには、Fuse Online のアップグレードを開始する前に、アップグレード前の Docker イメージへのアクセス の説明にしたがって docker クレデンシャルを syndesis pull secret に追加します。

2. Fuse Online Operator をアップグレードするには、以下を行います。

   a. OpenShift Web コンソールで、**Operators** > **Installed Operators** とクリックします。

   b. **Fuse Online** Operator をクリックしてから、**Subscription** をクリックします。

   c. **Channel** の横にある **Edit** アイコンをクリックします。

   d. **fuse-online-v7.8** チャンネルをクリックし、**Save** をクリックします。

Fuse Online のインストール時に **手動** 更新を指定した場合、OpenShift ドキュメントの Manually approving a pending Operator upgrade セクションの手順にしたがって Operator の更新リクエストを承認します。

## 1.3. インストールスクリプトを使用した FUSE ONLINE のアップグレード

OperatorHub ではなく、インストールスクリプトを使用して Fuse Online をインストールした場合、Fuse Online をアップグレードする一般的な手順は次のとおりです。

- クラスター管理者は、Fuse Online バックアップおよびリストアデータベースの Docker イメージへのアクセスを設定します。

- OCP 3.11 の場合は、Fuse Online バックアップおよびリストアデータベースイメージを docker.io からローカルキャッシュにプルします。
- OCP 4.6 の場合は、Docker Hub クレデンシャルを既存の syndesis pull secret に追加します。

- 最新の Fuse Online リリースをダウンロードします。
- クラスター管理者から Fuse Online をアップグレードする権限を取得します。
- 更新スクリプトを実行します。

下記条件のアップグレード手順は同じです。

- Fuse Online 7.7 から Fuse Online 7.8
- Fuse Online 7.8 の古いバージョンから Fuse Online 7.8 の新しいバージョン

**前提条件**

- オンサイトで OCP に Fuse Online バージョン 7.7 がインストールされ、稼働している必要があります。**または**、OCP 3.11 に Fuse Online のバージョン 7.8 がインストールされ、稼働しており、新しいアプリケーションイメージへのアップグレードを検討中です。
  以前のバージョンの場合は以下が必要になります。
  - 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。
- **oc** クライアントツールをインストール済みであり、Fuse Online がインストールされている OCP クラスターに接続されている。
- クラスター管理者権限が必要です。この手順の最初の 2 つのステップで必要になります。

**手順**

1. Docker の制限エラーが発生しないようにするため、クラスター管理者は アップグレード前の Docker イメージへのアクセス の説明にしたがって、Docker イメージへのアクセスを設定します。
2. クラスター管理者は、Fuse Online パッケージをダウンロードしてして、特定のプロジェクトでユーザーに対して Fuse Online のアップグレード権限を割り当てる必要があります。

a. 以下の場所から Fuse Online インストールスクリプトが含まれるパッケージをダウンロードします。
https://github.com/syndesisio/fuse-online-install/releases/tag/1.11

ファイルシステムの任意の場所で、ダウンロードしたアーカイブを展開します。**fuse-online-install-1.11 ディレクトリーに** は、Fuse Online のアップグレード用のスクリプトとサポートファイルが含まれます。

b. 展開したアーカイブが含まれるディレクトリーに移動します。以下に例を示します。
**cd fuse-online-install-1.11**

c. 以下のように、クラスター管理者アカウントで OpenShift にログインします。
**oc login -u admin -p admin**

d. Fuse Online をアップグレードする必要のある OpenShift プロジェクトに切り替えます。
**oc project fuse-online-project**

e. Fuse Online のカスタムリソース定義を更新します。
**bash install_ocp.sh --setup**

f. そのプロジェクト内だけで、Fuse Online のアップグレード権限を割り当てます。たとえば、以下のコマンドは、**developer** ユーザーに、Fuse Online のアップグレード権限を割り当てます。クラスター管理者がこのコマンドを実行した後に、**developer** ユーザーは対象のプロジェクトでのみ (**fuse-online-project**) Fuse Online をアップグレードできます。
**bash install_ocp.sh --grant developer**

3. Fuse Online のアップグレード権限を割り当てられたユーザーは、アプリケーションを実行できます。

a. 以下のように、OpenShift にログインします。
**oc login -u developer**

b. 以下のように、Fuse Online をアップグレードするプロジェクトに切り替えます。
**oc project fuse-online-project**

c. アップグレードするバージョンを確認するには、以下のように **--version** オプションを指定して更新スクリプトを実行します。
**bash update_ocp.sh --version**

d. 以下のような更新スクリプトを実行します。
**bash update_ocp.sh**

このスクリプトの詳細を確認するには、**$ bash update_ocp.sh --help** を実行します。

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。

4. 次のように稼働中の Fuse Online インテグレーションをアップグレードします。

a. Fuse Online で、アップグレードするインテグレーションを選択します。

b. **Edit** を選択します。

c. **Publish** を選択してインテグレーションを再パブリッシュします。

インテグレーションの再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。

## 1.4. CAMEL 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。

# 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。

## 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 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」

## 3.1. CAMEL 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。

## 3.2. 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。

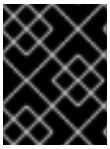
Fuse が実行される各コンテナーには専用の BOM ファイルがあります。

**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 ファイルの更新については、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** 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。

## 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。

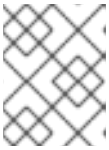
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.8.0.fuse-sb2-780038-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 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>7.8.0.fuse-sb2-780038-redhat-00001</fuse.version>
 <spring-boot.version>2.3.4.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 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。

## 4.1. CAMEL 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**

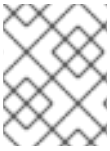
- Spring Boot 2.1 へのアップグレード。
- 追加のコンポーネントレベルのオプションが、spring-boot auto-configuration を使用して設定できるようになりました。これらのオプションは、ツールの支援のための spring-boot コンポーネントメタデータ JSON ファイル記述子に含まれます。

- 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。

## 4.2. 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。

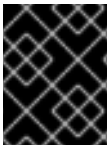


**注記**

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 ファイルの更新については、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** 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。

## 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>7.8.0.fuse-sb2-780038-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 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>
```

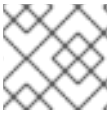
```
<dependency>
  <groupId>jakarta.inject</groupId>
  <artifactId>jakarta.inject-api</artifactId>
</dependency>
```

## 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、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 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 のパッチおよびアップグレードガイド 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。

## 4.6. 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 ランタイムの新規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 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

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。

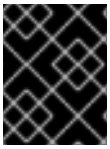
Fuse が実行される各コンテナーには専用の BOM ファイルがあります。

**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 ファイルの更新については、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** 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

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>7.8.0.fuse-sb2-780038-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 での FUSE スタンドアロンのアップグレード

本リリースには、メジャーおよびマイナーバージョンのコンポーネントに影響する多くのアップグレードが含まれています。ほとんどの OSGi バンドルは、次のメジャーバージョンまたは場合によってはマイナーバージョンを除外するバージョン範囲を設定します。

**警告**

Apache Karaf コンテナーを Fuse 7.8 にアップグレードするには、Fuse on Apache Karaf パッチメカニズムを使用しないでください。新しいインストールを実行し、設定やその他の変更ファイルが手動でコピーされる必要があります。

## 6.1. 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、Apache Karaf 上の Fuse 7.8 へのアップグレードについて説明します。

**手順**

1. Fuse on Apache Karaf をインストールするには、Installing on Apache Karaf の手順に従います。
2. Fuse 7.8 の新しい Fuse Karaf BOM を使用してアプリケーションを再構築します(BOM ファイルの更新に関する詳細は、最新の リリースノート を参照してください)。
3. 以前に Fuse 7.7 にインストールされていた機能のインストール
4. **etc** ディレクトリーを比較して、**ロギング**、**Web**、**Maven** などの設定変更の可能性を確認します。
5. **bin** ディレクトリーを比較して環境の変更（例： **bin/setenv** ）を比較します。
6. 再構築されたアプリケーションの Fuse 7.8 へのインストール

**注記**

アップグレード後、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 バナーに表示されます。