Red Hat AMQ 2021.Q3

AMQ Clients の概要

AMQ Clients 2.10 での使用

Last Updated: 2023-02-23

Red Hat AMQ 2021.Q3 AMQ Clients の概要

AMQ Clients 2.10 での使用

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、AMQ Clients 2.10 の機能およびコンポーネントについて説明します。また、本リリースでサポートされる一般的なユースケースと設計パターンについても説明します。

目次

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。これは大規模な取り組みであるため、これらの変更は今後の複数のリリースで段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージをご覧ください。

AMQ Clients は AMQP 1.0 および JMS クライアント、アダプター、およびライブラリーのスイートです。AMQ Clients には、既存のアプリケーションへの統合を可能にする JMS 2.0 サポートおよび新しいイベント駆動型 API が含まれます。

AMQ Clients は Red Hat AMQ の一部です。詳細は、「Red Hat AMQ 7 の概要 」を参照してください。

第1章 主な特長

- AMQP 1.0 のオープン標準プロトコル
- 業界標準 API - JMS 1.1 および 2.0
- 新しいイベント駆動型 API: 高速かつ効率的なメッセージングで場所を問わず統合
- 広範な言語サポート: C++、Java、JavaScript、Python、Ruby および .NET
- 幅広い可用性: Linux、Windows、および JVM ベースの環境

第2章 コンポーネント

2.1. AMQP CLIENTS

AMQ Clients には、AMQP 1.0 メッセージング API のスイートが含まれています。AMQP は、優れたメッセージング機能を備えた ISO 規格の汎用メッセージングプロトコルです。AMQ Broker および AMQ Interconnect は AMQP 1.0 をサポートするため、AMQP 1.0 クライアントと相互運用できます。

- AMQ C++
- AMQ JavaScript
- AMQ JMS (Java)
- AMQ .NET
- AMQ Python
- AMQ Ruby

2.2. JMS クライアント

AMQ Clients には、広範に使用されている Java Message Service (JMS) API の実装が複数含まれます。

- AMQ JMS - AMQ JMS は AMQP 1.0 を完全にサポートし、AMQ AMQP 1.0 サーバーまたはサービスと連携します。
- AMQ Core Protocol JMS - AMQ には、ActiveMQ Artemis Core プロトコルを基にした既存のアプリケーションをサポートするため、AMQ Core Protocol JMS クライアントが含まれます。

2.3. アダプターおよびライブラリー

AMQ Clients には、他のプラットフォームやコンポーネントと統合するためのコンポーネントが含まれています。

- AMQ JMS Pool: AMQ には、JMS リソースを効率的に使用するために、AMQ JMS Pool ライブラリーが含まれています。これにより、JMS API で定義される標準のライフサイクル以外の接続リソースを再利用できます。
- AMQ Spring Boot Starter - AMQ Spring Boot Starter を使用すると、AMQP 1.0 メッセージングを使用するスタンドアロン Spring アプリケーションを構築できます。

2.4. コンポーネントの互換性

以下の表は、AMQ Clients コンポーネントでサポートされる言語、プラットフォーム、プロトコル、およびサーバーを示しています。

コンポーネント	言語	プラットフォーム	プロトコル	サーバーおよびサービス

コンポーネント	言語	プラットフォーム	プロトコル	サーバーおよびサービス
AMQ C++	C++	linux、Windows	AMQP 1.0	AMQ Broker、AMQ Interconnect、および A-MQ 6
AMQ JavaScript	JavaScript	Linux、Windows、ブラウザー	AMQP 1.0	AMQ Broker、AMQ Interconnect、および A-MQ 6
AMQ JMS	Java	JVM	AMQP 1.0	AMQ Broker、AMQ Interconnect、および A-MQ 6
AMQ .NET	C#	linux、Windows	AMQP 1.0	AMQ Broker、AMQ Interconnect、および A-MQ 6
AMQ Python	Python	linux、Windows	AMQP 1.0	AMQ Broker、AMQ Interconnect、および A-MQ 6
AMQ Ruby	Ruby	Linux	AMQP 1.0	AMQ Broker、AMQ Interconnect、および A-MQ 6
AMQ Spring Boot Starter	Java	JVM	AMQP 1.0	AMQ Broker、AMQ Interconnect、および A-MQ 6
AMQ Core Protocol JMS	Java	JVM	コアプロトコル	AMQ Broker および A-MQ 6
AMQ JMS プール	Java	JVM	-	-

詳細は、「Red Hat AMQ 7 でサポートされる構成」を参照してください。

第3章 イベント駆動型の API

AMQ Clients で提供される API の多くは、非同期のイベント駆動型 API です。これには、C++、JavaScript、Python、Ruby API が含まれます。

これらの API は、ネットワークアクティビティーに合わせてアプリケーションイベント処理を実行することで機能します。ライブラリーは、ネットワーク I/O および実行イベントを監視します。イベントハンドラーはメインライブラリースレッド上で順次実行します。

イベントハンドラーはメインライブラリースレッドで実行されるため、長時間実行されるブロック操作をハンドラーコードに含めることはできません。イベントハンドラーでブロックすると、すべてのライブラリーの実行がブロックされます。長時間にわたるブロック操作を実行する必要がある場合は、別のスレッドで呼び出す必要があります。イベント駆動型の API には、ライブラリースレッドとアプリケーションスレッド間の調整をサポートする、クロススレッド通信機能が含まれています。

イベントハンドラーにおけるブロックの回避

イベントハンドラーで長時間実行されるブロックを呼び出すと、すべてのライブラリーの実行が停止し、ライブラリーが他のイベントを処理しなくなってしまい、定期的なタスクの実行ができなくなります。長時間実行されるブロック手順は必ず、別のアプリケーションスレッドで開始します。

第4章 AMQP

AMQP は、メッセージを確実に送受信するためのオープンインターネットプロトコルです。これは、複数のソフトウェアベンダーや主要な機関によりサポートされます。AMQP 1.0 は 2012 年に OASIS 標準と 2014 年に ISO 標準になりました。

- セッション多重化機能のあるフレーム化プロトコル
- ピアツーピアおよびクライアントサーバー接続に対応
- 標準タイプシステムでロスレスデータ交換を実現
- フロー制御、ハートビート、およびリソース制限を提供して分散システムの信頼性を強化
- 領域効率の高いバイナリーエンコーディングおよびパイプを使用してレイテンシーを軽減

4.1. AMQP 配信の保証

解決用の AMQP モデルは、メッセージ配信のライフサイクルに基づいています。リンクの最後には、メッセージ転送を表すエンティティーが作成され、しばらくの間存在し、最後に「settled」になります。Settled になると、完了を待たずに次に進めます。配送ライフサイクルには、約 4 つのイベントがあります。

- 配信が送信側で作成される。
- 配信が受信側で作成される。
- 配信が送信者に設定される。
- 配信が受信側で設定される。

送信側と受信側が同時に動作しているため、これらのイベントはさまざまな順序で発生する可能性があります。また、これらのイベントの順序により、メッセージ配信の保証が異なります。

At-most-once 配信

At-most-once 配信は、「presettled」または「fire and forget」配信とも呼ばれます。

1. 配信が送信側で作成される。
2. 配信が送信者に設定される。
3. 配信が受信側で作成される。
4. 配信が受信側で設定される。

この設定では、送信側が配信を解決 (つまり、完了を待たずに実行) してから、受信側に到達し、配信中に問題が発生した場合には、送信側は再送信するためのベースがありません。

このモードは、定期的なセンサーデータなど、一時的にメッセージが損失しても許容できるアプリケーションや、また、アプリケーション自体が障害と再送信を検出できるアプリケーションの場合に適しています。

at-least-once 配信

1. 配信が送信側で作成される。
2. 配信が受信側で作成される。

3. 配信が受信側で設定される。

4. 配信が送信者に設定される。

この設定では、受信側が受信した時点で配信を解決し、送信側は、受信側が解決した時点で、配信を解決します。配信中に問題が発生すると、送信側は再送信できます。ただし、受信側がすでに配信について完了を待たずに次に進んでいるので、再送信すると、メッセージの配信が重複してしまいます。アプリケーションは一意のメッセージ ID を使用することで重複を除外できます。

第5章 重要事項

5.1. 優先クライアント

通常、AMQP 1.0 標準仕様をサポートする AMQ クライアントが、新しいアプリケーションの開発に推奨されます。ただし、以下の例外が適用されます。

- 実装で分散トランザクションが必要な場合は、AMQ Core Protocol JMS クライアントを使用します。
- (たとえば IoT アプリケーション用) で MQTT または STOMP が必要な場合は、コミュニティーがサポートする MQTT クライアントまたは STOMP クライアントを使用します。

5.2. レガシークライアント

- **AMQ OpenWire JMS クライアントが非推奨になる**
  AMQ OpenWire JMS クライアントが AMQ 7 で非推奨になりました。このクライアントのユーザーは、AMQ JMS または AMQ Core Protocol JMS に移行することが推奨されます。
- **CMS および NMS API が非推奨になる**
  ActiveMQ CMS および NMS メッセージング API は AMQ 7 で非推奨となりました。CMS API のユーザーは AMQ C++ に移行し、NMS API のユーザーは AMQ .NET に移行することが推奨されます。CMS および NMS API の機能が AMQ 7 では縮小されている可能性があります。
- **レガシー AMQ C++ クライアントの非推奨**
  AMQ C++ クライアント (以前は MRG Messaging で提供されていた C++ クライアント) は、AMQ 7 で非推奨となりました。この API のユーザーは、AMQ C++ に移行することが推奨されます。
- **コア API はサポート対象外**
  Artemis Core API クライアントはサポートされません。このクライアントは、サポートされる AMQ Core Protocol JMS クライアントとは異なります。

第6章 重要なリンク

- Red Hat AMQ 7 でサポートされる構成
- Red Hat AMQ 7 コンポーネントの詳細

改訂日時: 2021-08-29 15:55:28 +1000