Red Hat AMQ 7.7

OpenShift での AMQ Online の使用

AMQ Online 1.5 で使用する場合

Last Updated: 2023-02-03

Red Hat AMQ 7.7 OpenShift での AMQ Online の使用

AMQ Online 1.5 で使用する場合

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

このガイドでは、AMQ Online の使用方法について説明します。

目次

第1章 はじめに **4**
1.1. AMQ ONLINE の概要 4
1.2. サポートされる機能 5
1.3. AMQ ONLINE ユーザーのロール 6
1.4. サポートされる構成 7
1.5. 本書の表記慣例 7
1.5.1. 変数テキスト 7

第2章 アドレス空間の管理 **8**
2.1. アドレス空間 8
2.2. 標準アドレス空間 8
2.2.1. 標準アドレスタイプ 8
2.2.1.1. Queue 9
2.2.1.2. トピック 9
2.2.1.2.1. 階層トピックとワイルドカード 9
2.2.1.2.2. 階層トピックでサブスクライバーを作成する際の既知の問題 9
2.2.1.3. anycast 9
2.2.1.4. Multicast 10
2.2.1.5. Subscription 10
2.3. ブローカーアドレス空間 10
2.3.1. ブローカーアドレスの種類 10
2.3.1.1. Queue 10
2.3.1.2. トピック 11
2.3.1.2.1. 階層トピックとワイルドカード 11
2.3.1.2.2. 階層トピックでサブスクライバーを作成する際の既知の問題 11
2.4. アドレス空間計画 11
2.5. コマンドラインでの利用可能なアドレス空間プランの一覧表示 11
2.6. コマンドラインでの利用可能な認証サービスの一覧表示 12
2.7. アドレス空間の例 12
2.7.1. アドレス空間の例 12
2.7.2. 認証サービスを使用したアドレス空間の例 12
2.7.3. オーバーライドを許可する外部認証サービスを使用したアドレス空間の例 13
2.7.4. エンドポイントを外部に公開するアドレス空間の例 13
2.7.4.1. OpenShift LoadBalancer サービスの例 13
2.7.4.2. OpenShift ルートの例 14
2.7.5. アドレス空間証明書プロバイダーの設定例 15
2.7.5.1. openshift プロバイダー 15
2.7.5.2. selfsigned プロバイダー 15
2.7.5.3. certBundle プロバイダー 16
2.7.6. アドレス空間の例のエクスポート 17
2.7.6.1. ConfigMap および Secret タイプのエクスポートの例 17
2.7.6.2. Service タイプのエクスポートの例 17
2.8. アドレス空間ステータス出力の例 17
2.9. アドレス空間情報をアプリケーション名前空間にエクスポートする例 18
2.10. アドレス空間コネクターの例 19
2.10.1. SASL PLAIN を使用したアドレス空間コネクター 19
2.10.2. 相互 TLS を使用するアドレス空間コネクター 21
2.11. コマンド行を使用したアドレス空間の作成 22
2.12. RED HAT AMQ コンソールを使用したアドレス空間の作成 22
2.13. RED HAT AMQ コンソールを使用したアドレス空間に関連付けられたアドレス空間プランの変更 23
2.14. RED HAT AMQ コンソールを使用したアドレス空間に関連付けられた認証サービスの変更 23

2.15. RED HAT AMQ コンソールを使用したアドレス空間の削除 24
2.16. アドレス空間情報を取得するためのコマンド例 24
2.17. コマンドラインを使用したアドレス空間の置換 24

第3章 アドレスの管理 26
3.1. アドレス 26
3.2. アドレスプラン 26
3.2.1. アドレスの例 26
3.2.2. トピックとサブスクリプションアドレスの例 26
3.2.3. アドレス TTL 制限の例 27
3.2.4. アドレス転送の例 28
3.2.4.1. ローカルキューからリモート AMQP サーバーへのメッセージの転送 28
3.2.4.2. リモート AMQP サーバーからローカルキューへのメッセージの転送 29
3.3. コマンドラインを使用した利用可能なアドレスプランの一覧表示 29
3.4. コマンドラインを使用したアドレスの作成 29
3.5. RED HAT AMQ コンソールを使用したアドレスの作成 30
3.6. コマンドラインを使用したアドレスの置換 30

第4章 RED HAT AMQ コンソールの使用 32
4.1. RED HAT AMQ コンソールのユーザー権限 32
4.2. RED HAT AMQ コンソールへのアクセス 32
4.3. RED HAT AMQ コンソールを使用したメッセージおよびアプリケーション接続統計の表示 32
4.4. RED HAT AMQ コンソールを使用したエンドポイント情報の表示 34
4.5. キューとサブスクリプションのパージ 36
4.6. RED HAT AMQ コンソールを使用した接続の切断 36

第5章 ユーザーモデル 38
5.1. 認証 38
5.1.1. パスワード認証の種類 38
5.1.2. サービスアカウント認証タイプ 39
5.2. 承認 39
5.3. ユーザーの管理 39
5.3.1. コマンドラインでのユーザーの作成 39
5.3.2. コマンドラインを使用したユーザーの削除 40
5.3.3. コマンドラインを使用したユーザー権限の管理 40

第6章 アプリケーションの AMQ ONLINE への接続 42
6.1. 自己署名 CA 証明書の取得 42
6.2. クライアントの例 43
6.2.1. AMQ Online Python の例 43
6.2.1.1. 階層トピックでサブスクライバーを作成する際の既知の問題 43
6.2.2. AMQ Online JMS の例 44
6.2.3. AMQ Online JavaScript の例 45
6.2.3.1. WebSocket を使用した AMQ Online JavaScript の例 45
6.2.4. AMQ Online C++ の例 46
6.2.4.1. 階層トピックでサブスクライバーを作成する際の既知の問題 47
6.2.5. AMQ Online .NET の例 48

付録A サブスクリプションの使用 49
アカウントへのアクセス 49
サブスクリプションのアクティベート 49
zip および tar ファイルのダウンロード 49
パッケージ用のシステムの登録 49

付録B メッセージングテナント向けの AMQ ONLINE リソース 50

第1章 はじめに

1.1. AMQ ONLINE の概要

Red Hat AMQ Online は、マネージドサービスとしてメッセージングを配信するための OpenShift ベースのメカニズムです。Red Hat AMQ Online を使用すると、管理者は、クラウドまたはオンプレミスのいずれかで、クラウドネイティブのマルチテナントメッセージングサービスを設定できます。開発者は、Red Hat AMQ コンソールを使用してメッセージングをプロビジョニングできます。複数の開発チームがコンソールからブローカーとキューをプロビジョニングできます。各チームがソフトウェアをインストール、設定、デプロイメント、保守、またはパッチを適用する必要はありません。

AMQ Online は、ユースケースに応じてさまざまな種類のメッセージングをプロビジョニングできます。ユーザーは、Address Space を作成することでメッセージングリソースをリクエストできます。AMQ Online は現在、標準とブローカーの 2 つのアドレス空間タイプをサポートしており、それぞれセマンティクスが異なります。次の図は、各アドレス空間タイプのアーキテクチャーの概要を示しています。

図1.1 標準アドレス空間

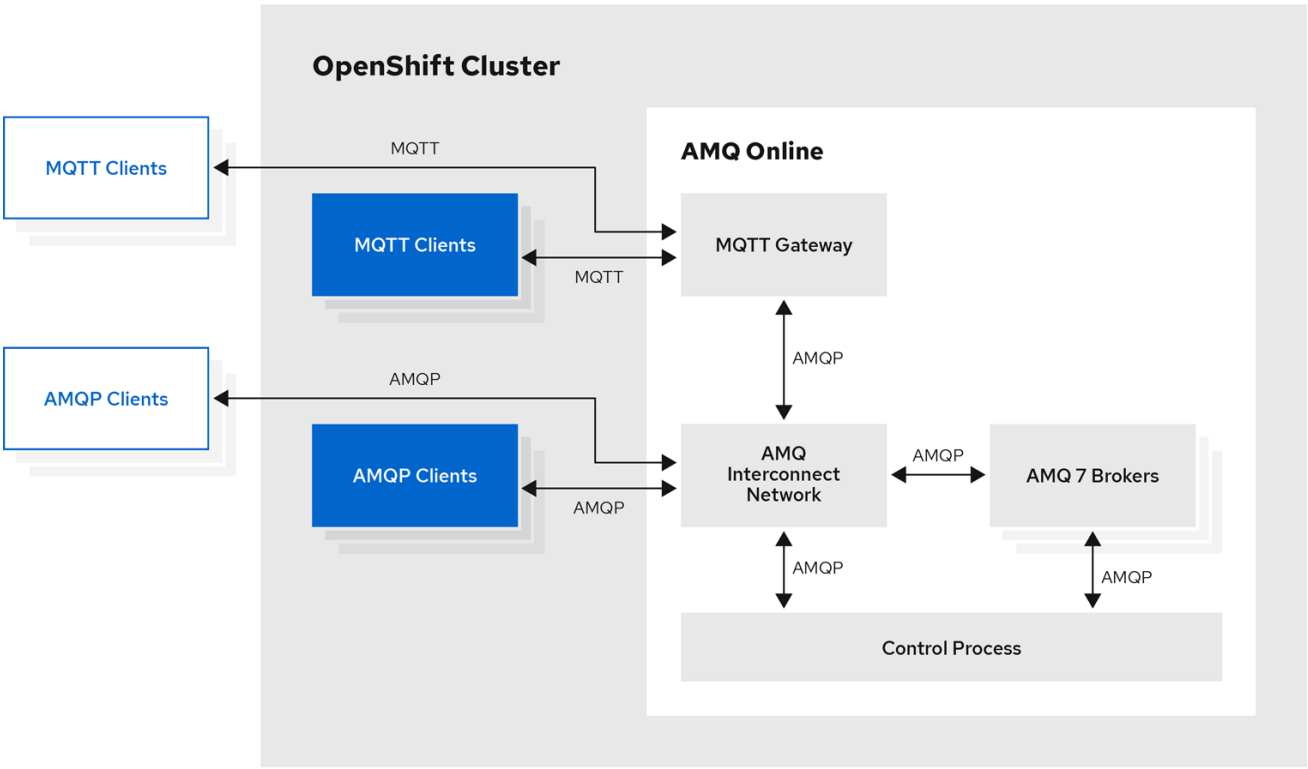
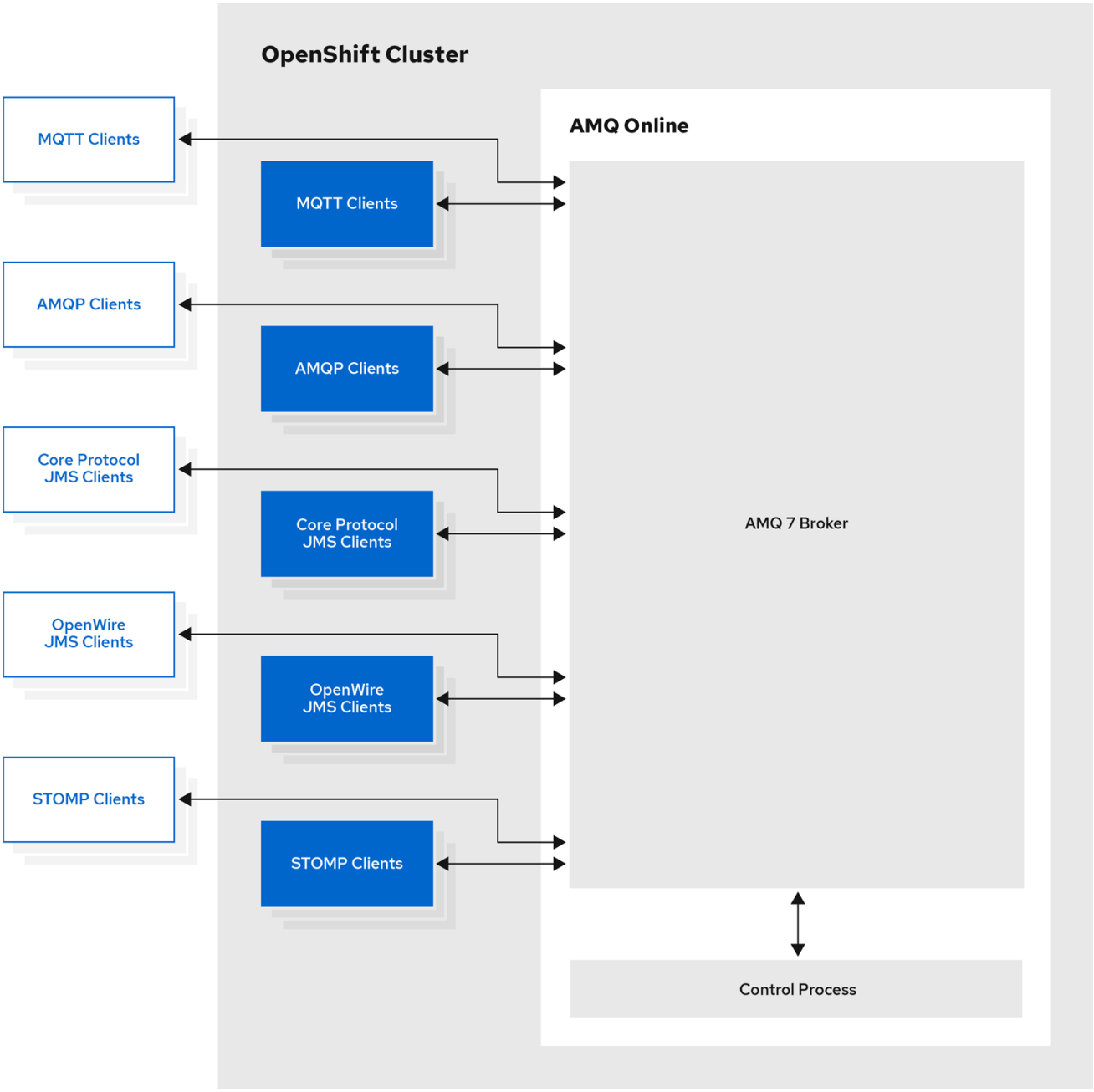


図1.2 ブローカーアドレス空間



1.2. サポートされる機能

次の表は、AMQ Online 1.5 でサポートされている機能を示しています。

表1.1 サポート対象機能の参照表

機能		ブローカーアドレス空間	標準アドレス空間
アドレスの種類	Queue	はい	はい
	トピック	はい	はい

機能		ブローカーアドレス空間	標準アドレス空間
	マルチキャスト	いいえ	はい
	anycast	いいえ	はい
	Subscription	いいえ	はい
メッセージングプロトコル	AMQP	はい	はい
	MQTT	はい	テクノロジープレビューとしてのみ提供
	CORE	はい	いいえ
	OpenWire	はい	いいえ
	STOMP	はい	いいえ
トランスポート	TCP	はい	はい
	WebSocket	はい	はい
永続サブスクリプション	JMS 永続サブスクリプション	はい	いいえ
	名前付き永続サブスクリプション	いいえ	はい
JMS	トランザクションサポート	はい	いいえ
	キューのセレクター	はい	いいえ
	メッセージ順序の保証 (優先順位付けを含む)	はい	いいえ
スケーラビリティー	スケーラブルな分散キューとトピック	いいえ	はい

1.3. AMQ ONLINE ユーザーのロール

AMQ Online ユーザーは、サービス管理者とメッセージングテナントの 2 つのユーザーロールに関して広く定義できます。組織の規模に応じて、これらのロールは同じユーザーまたは異なるユーザーによって実行される場合があります。

メッセージングテナントは、クラウドネイティブの API とツールの両方を使用して、メッセージングリソースを要求できます。メッセージングテナントは、メッセージングシステム内の特定のアドレス空間のユーザーとアクセス許可を管理したり、アドレス管理とアドレスを作成したりすることもできま

す。**OpenShift での AMQ Online の使用** では、これらのタスクを実行する方法を説明します。

サービス管理者のロールは、初期インストールとその後のアップグレードを実行します。サービス管理者は、ルーター、ブローカー、および管理コンポーネントの監視など、メッセージングインフラストラクチャーをデプロイメントおよび管理するだけでなく、アドレス空間計画やアドレス計画も作成します。AMQ Online のセットアップと管理、およびインフラストラクチャーとプランの設定方法は、**IOpenShift deno AMQ Online のインストールおよび管理** を参照してください。

1.4. サポートされる構成

AMQ Online でサポートされる設定の詳細は、Red Hat AMQ 7 でのサポート対象設定 を参照してください。

1.5. 本書の表記慣例

1.5.1. 変数テキスト

本書では、変数を含むコードブロックが紹介されていますが、これは、お客様のシステム環境に固有の値に置き換える必要があります。このドキュメントでは、そのようなテキストはイタリックモノスペースのスタイルで指定しています。

たとえば、次のコードブロックで、**my-namespace** はインストールで使用されている namespace に置き換えます。

```
sed -i 's/amq-online-infra/my-namespace/' install/bundles/enmasse-with-standard-authservice/*.yaml
```

第2章 アドレス空間の管理

AMQ Online は、OpenShift コマンドラインツールを使用したアドレス空間の管理をサポートするように設定されています。アドレス空間は、**oc** を使用して他の OpenShift リソースと同様に管理されます。

2.1. アドレス空間

アドレス空間は、1 つの接続 (プロトコルごと) を介してアクセスできるアドレスのグループです。アドレス空間のエンドポイントに接続されたクライアントは、そのアドレス空間内の承認されたアドレスとの間でメッセージを送受信できます。アドレス空間は、アドレス空間タイプで定義されているように、複数のプロトコルをサポートできます。

注記

既存のアドレス空間のエンドポイントは、変更できません。

AMQ Online には、次の 2 種類のアドレス空間があります。

- Standard (標準)
- Brokered (ブローカー)

2.2. 標準アドレス空間

標準アドレス空間は、AMQ Online のデフォルトのアドレス空間です。これは、取り付け可能なストレージユニットと組み合わせた AMQP ルーターネットワークで設定されます。クライアントは、1 つ以上のメッセージブローカーとの間でメッセージを転送するメッセージルーターに接続します。このアドレス空間タイプは、接続とアドレスが多数ある場合に適しています。ただし、標準アドレス空間には次の制限があります。

- XA トランザクションのサポート
- メッセージ順なし
- キューにセレクターなし
- キューのブラウズなし
- メッセージグループなし

クライアントは、AMQP または MQTT プロトコルを使用して、このアドレス空間に接続し、メッセージを送受信します。MQTT は qos2 または保持メッセージをサポートしていないことに注意してください。

2.2.1. 標準アドレスタイプ

標準アドレス空間は、次の異なるアドレスタイプを 5 つサポートしています。

- queue
- topic
- anycast

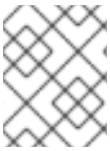
- multicast
- subscription

2.2.1.1. Queue

キューアドレスタイプは、ストアアンドフォワードキューです。このアドレスタイプは、分散ワークキューの実装、トラフィックバーストの処理、およびプロデューサーとコンシューマーを切り離す場合などのユースケースに適しています。キューは、複数のストレージユニット全体でシャード化できます。標準アドレス空間のキューでは、メッセージの順序が失われる可能性があります。

2.2.1.2. トピック

トピックアドレスタイプは、1..N のプロデューサーと 1..M のコンシューマーが存在するパブリッシュ/サブスクライブメッセージングパターンをサポートします。トピックアドレス対して発行された各メッセージは、そのアドレスの全サブスクライバーに転送されます。サブスクライバーは永続性がある場合があり、そのような場合には、メッセージはサブスクライバーが確認応答するまで保持されます。



注記

トピックでサブスクリプションを作成する場合に、そのトピックへの送信者は **トピック** ケイパビリティーを指定する必要があります。

2.2.1.2.1. 階層トピックとワイルドカード

トピックアドレスから受信するクライアントは、トピックアドレスをルートとしてワイルドカードアドレスを指定できます。ワイルドカードの動作は MQTT 構文に従います。

- / はセパレーターです。
- **+** はレベル 1 つと照合します。
- **#** 1 つ以上のレベルで照合します。

たとえば、次のようになります。

- **a/#/b** では **a/foo/b**、**a/bar/b**、および **a/foo/bar/b** がマッチします。
- **a/+/b** は、**a/foo/b** と **a/bar/b** がマッチしますが、**a/foo/bar** はマッチしません。

標準アドレス空間では、最初のレベルは常に定義済みのトピックアドレスでなければなりません。つまり、**#** と **+** は、サブスクライブしたアドレスの最初の文字には使用できません。

2.2.1.2.2. 階層トピックでサブスクライバーを作成する際の既知の問題

AMQ Online で階層トピックにサブスクライバーを作成すると、ブローカーが代わりに競合するコンシューマーとしてサブスクライバーを作成するという既知の問題が存在します (トピックではなくキューのようにアドレスを処理します)。クライアントの特定の回避策の詳細は、アプリケーションの AMQ Online への接続 の該当するクライアントの例のセクションを参照してください。

2.2.1.3. anycast

anycast アドレスタイプは、メッセージを 1 つのコンシューマーに送信するためのスケーラブルな直接アドレスです。anycast アドレスに送信されたメッセージは保存されず、代わりにコンシューマーに直接転送されます。この方法により、このアドレスの種類は、要求応答 (RPC) の使用や作業の分散に適

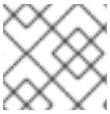
しています。これは永続性を必要としないため、最もコストがかからないアドレスタイプです。

2.2.1.4. Multicast

multicast アドレスタイプは、複数のコンシューマーにメッセージを送信するためのスケーラブルな直接アドレスです。multicast アドレスに送信されたメッセージは、そのアドレスでメッセージを受信するすべてのコンシューマーに転送されます。コンシューマーからのメッセージ受信確認はプロデューサーに伝播されないため、マルチキャストアドレスに送信できるのは事前に解決されたメッセージのみです。

2.2.1.5. Subscription

サブスクリプションアドレスタイプを使用すると、サブスクライバーがアタッチされていない場合でも、トピックに公開されたメッセージを保持するトピックに対してサブスクリプションを作成できます。コンシューマーは、アドレス構文 <topic-address>::<subscription-address> を使用してサブスクリプションにアクセスします。たとえば、トピック **mytopic** のサブスクリプション **mysub** の場合には、コンシューマーはアドレス **mytopic::mysub** からサブスクリプションにアクセスします。デフォルト設定では、サブスクリプションごとにコンシューマー 1 つのみが使用できます。この設定は、サブスクリプションアドレスの **maxConsumers** フィールドを編集することで変更できます。



注記

maxConsumers 設定は、既存のサブスクリプションでは変更できません。

2.3. ブローカーアドレス空間

ブローカーのアドレス空間は、ブローカー固有の機能をサポートするように設計されていますが、接続数とアドレス数の点でサイズの制限があります。このアドレス空間は、JMS トランザクション、メッセージグループ、およびキューとトピックのセレクターをサポートします。

クライアントは、次のプロトコルを使用して、このアドレス空間でメッセージを送受信するだけでなく、接続することもできます。

- AMQP
- CORE
- OpenWire
- MQTT
- STOMP

2.3.1. ブローカーアドレスの種類

ブローカーアドレス空間は、次の 2 つのアドレスタイプをサポートします。

- queue
- topic

2.3.1.1. Queue

キューアドレスタイプは、ストアアンドフォワードキューです。このアドレスタイプは、分散ワークキューの実装、トラフィックバーストの処理、およびプロデューサーとコンシューマーを切り離す場合

などのユースケースに適しています。ブローカーアドレス空間のキューは、セレクター、メッセージグループ、トランザクション、およびその他の JMS 機能をサポートします。メッセージの順序は、メッセージが解放されると失われる可能性があります。

2.3.1.2. トピック

トピックアドレスタイプは、1..N のプロデューサーと 1..M のコンシューマーが存在するパブリッシュ/サブスクライブメッセージングパターンをサポートします。トピックアドレス対して発行された各メッセージは、そのアドレスの全サブスクライバーに転送されます。サブスクライバーは永続性がある場合があり、そのような場合には、メッセージはサブスクライバーが確認応答するまで保持されます。

2.3.1.2.1. 階層トピックとワイルドカード

トピックアドレスから受信するクライアントは、トピックアドレスをルートとしてワイルドカードアドレスを指定できます。ワイルドカードの動作は MQTT 構文に従います。

- / はセパレーターです。
- **+** はレベル 1 つと照合します。
- **#** 1 つ以上のレベルで照合します。

たとえば、次のようになります。

- **a/#/b** では **a/foo/b**、**a/bar/b**、および **a/foo/bar/b** がマッチします。
- **a/+/b** は、**a/foo/b** と **a/bar/b** がマッチしますが、**a/foo/bar** はマッチしません。

2.3.1.2.2. 階層トピックでサブスクライバーを作成する際の既知の問題

AMQ Online で階層トピックにサブスクライバーを作成すると、ブローカーが代わりに競合するコンシューマーとしてサブスクライバーを作成するという既知の問題が存在します (トピックではなくキューのようにアドレスを処理します)。クライアントの特定の回避策の詳細は、アプリケーションの AMQ Online への接続 の該当するクライアントの例のセクションを参照してください。

2.4. アドレス空間計画

アドレス空間は、アドレス空間で使用できるリソース量を記述するアドレス空間計画で設定されます。アドレス空間プランはサービス管理者によって設定され、AMQ Online のインストールごとに異なる場合があります。

アドレス空間に必要なリソースが増減する場合は、アドレス空間計画を変更できます。

2.5. コマンドラインでの利用可能なアドレス空間プランの一覧表示

アドレス空間タイプで利用可能なアドレス空間プランを一覧表示できます。

手順

1. メッセージングテナントとしてログインします。

```
oc login -u developer
```

2. 利用可能なアドレス空間プランを示すスキーマを取得します (ブローカーアドレス空間タイプは **standard** から **brokered** に置き換えます)。

```
oc get addressspaceschema standard -o jsonpath='{.spec.plans[*].name}'
```

2.6. コマンドラインでの利用可能な認証サービスの一覧表示

アドレス空間タイプで利用可能な認証サービスを一覧表示できます。

手順

1. メッセージングテナントとしてログインします。

```
oc login -u developer
```

2. リストされた認証サービスを使用してスキーマを取得します (ブローカーアドレス空間タイプは **standard** から **brokered** に置き換えます)。

```
oc get addressspaceschema standard -o jsonpath='{.spec.authenticationServices}'
```

2.7. アドレス空間の例

2.7.1. アドレス空間の例

このアドレス空間の例では、**AddressSpace** を作成するために必要なオプションのみを示しています。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard (1)
  plan: standard-unlimited (2)
```

(1) アドレス空間のタイプは、**brokered** または **standard** のいずれかです。

(2) アドレス空間計画は、アドレス空間のタイプと、AMQ Online 管理者によって設定された内容によって異なります。利用可能なアドレス空間プランを表示するには、利用可能なアドレス空間プランの リスト を参照してください。

2.7.2. 認証サービスを使用したアドレス空間の例

このアドレス空間の例は、**AddressSpace** の認証サービスを設定する方法を示しています。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
```

```
plan: standard-unlimited
authenticationService:
  name: standard-authservice (1)
```

(1) 認証サービス名は、AMQ Online 管理者が設定した、使用可能な認証サービスによって異なります。アドレス空間タイプで使用可能な認証サービスを表示するには、使用可能な認証サービスのリスト を参照してください。

2.7.3. オーバーライドを許可する外部認証サービスを使用したアドレス空間の例

このアドレス空間の例は、外部認証サービスのホスト名、ポート番号、およびレルムをオーバーライドする方法を示しています。オーバーライドを指定できるかどうかは、AMQ Online 管理者が外部認証サービスをどのように設定しているかによって異なります。

メッセージングテナントがホスト名、ポート番号、およびレルムを上書きできるように外部認証サービスを設定する方法は、オーバーライドを許可する外部認証サービスの例 を参照してください。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  authenticationService:
    name: external-authservice (1)
    type: external
    overrides: (2)
      realm: amq-online-infra-space-standard-auth
      host: standard-authservice-amq-online-infra.apps.wfd-28d9.openshiftworkshop.com
      port: 5671
      caCertSecret:
        name: my-ca-cert
```

(1) 認証サービス名は、AMQ Online 管理者が設定した、使用可能な認証サービスによって異なります。アドレス空間タイプで使用可能な認証サービスを表示するには、使用可能な認証サービスのリスト を参照してください。

(2) オーバーライド値を指定します。

2.7.4. エンドポイントを外部に公開するアドレス空間の例

これらのアドレス空間の例は、**AddressSpace** の外部エンドポイントを設定して、OpenShift クラスター外のメッセージングエンドポイントにアクセスする方法を示しています。

2.7.4.1. OpenShift LoadBalancer サービスの例

OpenShift **LoadBalancer** サービスを介して **AddressSpace** エンドポイントを公開するには、**loadbalancer** タイプが使用されます。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
```

```
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  authenticationService:
    name: standard-authservice
  endpoints:
  - name: messaging (1)
    service: messaging (2)
    expose:
     type: loadbalancer (3)
     loadBalancerPorts: (4)
     - amqp
     - amqps
      annotations: (5)
        mykey: myvalue
      loadBalancerSourceRanges: (6)
      - 10.0.0.0/8
```

(1) (必須) エンドポイントの名前。指定した名前は、作成される OpenShift サービスの名前と、**AddressSpace** の status セクションのエンドポイントの名前に影響があります。

(2) (必須) エンドポイント用に設定されたサービス。**service** の有効な値は、**messaging** と **mqtt** です。ただし、**mqtt** サービスは、**standard** のアドレス空間タイプでのみサポートされています。

(3) (必須) 公開されるエンドポイントのタイプ。**loadbalancer** タイプは、OpenShift **LoadBalancer** サービスを作成します。有効な値は **route** と **loadbalancer** です。

(4) (必須) **LoadBalancer** サービスで公開されるポートのリスト。**メッセージング** サービスの場合には、有効な値は **amqp** と **amqps** です。

(5) (オプション) **LoadBalancer Service** オブジェクトに追加される一連のキーと値のアノテーションペア。

(6) (省略可能) ロードバランサーによって受け入れられる許可されたソース範囲。

2.7.4.2. OpenShift ルートの例

AddressSpace エンドポイントを OpenShift ルートとして公開するには、次の **route** タイプが使用されます。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  authenticationService:
    name: standard-authservice
  endpoints:
  - name: messaging (1)
    service: messaging (2)
```

```
  expose:
   type: route
   routeServicePort: amqps (3)
   routeTlsTermination: passthrough (4)
   routeHost: messaging.example.com (5)
```

(1) (必須) エンドポイントの名前。指定した名前は、作成される OpenShift サービスの名前と、**AddressSpace** の status セクションのエンドポイントの名前に影響があります。

(2) (必須) エンドポイント用に設定されたサービス。**service** の有効な値は、**messaging** または **mqtt** です。ただし、**mqtt** サービスは、**standard** のアドレス空間タイプでのみサポートされています。

(3) (必須) 公開するポートの名前。**route** タイプでは、TLS が有効なポートを 1 つだけ指定できます。**messaging** サービスの場合には、有効な値は **amqps** または **https** です。

(4) (必須) OpenShift ルートに使用される TLS 終了ポリシー。**messaging** サービスの場合、**amqps** ポートでは **passthrough** を指定する必要がありますが、**https** (websockets) では **reencrypt** も使用できます。

(5) (オプション) 作成されたルートに使用するホスト名。

2.7.5. アドレス空間証明書プロバイダーの設定例

次のアドレス空間の例は、さまざまな証明書プロバイダーを使用して **AddressSpace** のエンドポイントを設定する方法を示しています。証明書プロバイダーは、**AddressSpace** のエンドポイントに対して証明書を発行する方法を決定します。

2.7.5.1. openshift プロバイダー

openshift 証明書プロバイダーを使用して、OpenShift クラスター認証局 (CA) によって署名された証明書でエンドポイントを設定できます。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  authenticationService:
    name: standard-authservice
  endpoints:
  - name: messaging
    service: messaging
    cert:
     provider: openshift (1)
```

(1) (必須) 証明書プロバイダーのタイプ。有効な値は、**openshift** (OpenShift のみ)、**certBundle**、および **selfsigned** (デフォルト値) です。

2.7.5.2. selfsigned プロバイダー

selfsigned 証明書プロバイダーを使用して、自己署名証明書でエンドポイントを設定できます。これらの証明書の CA は、**AddressSpace** リソースの **status.caCert** フィールドにあります。

注記

実稼働環境で自己署名証明書を使用することはお勧めしません。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  authenticationService:
    name: standard-authservice
  endpoints:
  - name: messaging
    service: messaging
    cert:
      provider: selfsigned (1)
```

1. (必須) 証明書プロバイダーのタイプ。有効な値は、**openshift** (OpenShift のみ)、**certBundle**、および **selfsigned** (デフォルト値) です。

2.7.5.3. certBundle プロバイダー

certBundle 証明書プロバイダーを使用して、独自の CA によって署名されたユーザー提供の証明書でエンドポイントを設定できます。証明書のローテーションは、更新された証明書で **tlsKey** フィールドと **tlsCert** フィールドを更新してから、**AddressSpace** リソースを更新して実行できます。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  authenticationService:
    name: standard-authservice
  endpoints:
  - name: messaging
    service: messaging
    cert:
      provider: certBundle (1)
      tlsKey: Y2VydGJ1bmRsZXByb3ZpZGVyY2VydA== (2)
      tlsCert: Y2VydGJ1bmRsZXByb3ZpZGVyY2VydA== (3)
```

1. (必須) 証明書プロバイダーのタイプ。有効な値は、**openshift** (OpenShift のみ)、**certBundle**、および **selfsigned** (デフォルト値) です。

2. (必須) PEM 秘密鍵の base64 エンコード値 (preamble を含む)。

3 (必須) PEM 証明書の base64 エンコード値 (preamble を含む)。

2.7.6. アドレス空間の例のエクスポート

次の 3 つのエクスポートタイプを使用して、アドレス空間情報をエクスポートできます。

- **ConfigMap**
- **Secret**
- **Service**

2.7.6.1. ConfigMap および Secret タイプのエクスポートの例

この例は、**ConfigMap** エクスポートタイプで使用される形式を示しています。**Secret** エクスポートタイプの形式は、**ConfigMap** エクスポートタイプと同じキーを使用しますが、値は Base64 でエンコードされます。

```
service.host: messaging.svc
service.port.amqp: 5672
external.host: external.example.com
external.port: 5671
ca.crt: // PEM formatted CA
```

2.7.6.2. Service タイプのエクスポートの例

この例は、**Service** エクスポートタイプで使用される形式を示しています。

```
  externalName:  messaging.svc
    ports:
    - name: amqp
      port: 5672
      protocol: TCP
      targetPort: 5672
```

2.8. アドレス空間ステータス出力の例

AddressSpace リソースには、その状態とエンドポイントに関する情報を取得するために使用できる **status** フィールドが含まれています。次の出力は、**oc get addressspace myspace -o yaml** を実行して得られる出力の例です。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  ...
status:
  isReady: false 1
  messages:
    - "One or more deployments are not ready: "
  endpointStatuses: 2
```

```
  - name: messaging
    cert: aGVsbG8= (3)
    serviceHost: messaging-123.enmasse-infra.svc (4)
    servicePorts: (5)
      - name: amqp
        port: 5672
      - name: amqps
        port: 5671
    externalHost: messaging.example.com (6)
    externalPorts: (7)
      - name: amqps
        port: 443
```

(1) **status.isReady** フィールドは、**true** または **false** のいずれかです。

(2) **status.endpointStatuses** フィールドは、このアドレス空間で利用可能なエンドポイントに関する情報を提供します。

(3) **cert** フィールドには、特定のエンドポイントの base64 でエンコードされた証明書が含まれています。

(4) **serviceHost** フィールドには、特定のエンドポイントのクラスター内部ホスト名が含まれています。

(5) **servicePorts** フィールドには、クラスター内部ホストで使用可能なポートが含まれています。

(6) **externalHost** フィールドには、特定のエンドポイントの外部ホスト名が含まれています。

(7) **externalPorts** フィールドには、外部ホストで使用可能なポートが含まれています。

2.9. アドレス空間情報をアプリケーション名前空間にエクスポートする例

このアドレス空間の例は、**AddressSpace** リソースのエンドポイント情報を、メッセージングアプリケーションと同じ名前空間の **ConfigMap**、**Secret**、または **Service** にエクスポートする方法を示しています。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  authenticationService:
    name: standard-authservice
  endpoints:
  - name: messaging
    service: messaging
    exports:
    - kind: ConfigMap (1)
      name: my-config (2)
```

1. (必須) エクスポートのタイプ: **ConfigMap**、**Secret**、または **Service**。結果として得られる **ConfigMap** には、exports format の例 に示されている形式の値が含まれます。**Secret** の場合、同

2. (必須) 作成および更新するリソースの名前。

エンドポイント情報をエクスポートする場合、**system:serviceaccounts:_amq-online-infra_** グループには、エクスポートリストで指定された configmap を作成、更新、および削除する権限が付与されている必要があります。これには、次のような RBAC ロールとロールバインディングを作成します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: rbac
rules:
  - apiGroups: [ "" ]
    resources: [ "configmaps" ]
    verbs: [ "create" ]
  - apiGroups: [ "" ]
    resources: [ "configmaps" ]
    resourceNames: [ "my-config" ]
    verbs: [ "get", "update", "patch" ]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: rbac-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: rbac
subjects:
- kind: Group
  name: system:serviceaccounts:_amq-online-infra_
```

2.10. アドレス空間コネクターの例

standard アドレス空間タイプを別の AMQP サーバーと連携できます。**リモートアドレス接続** と **メッセージストアアンドフォワード** の 2 つの操作方法がサポートされています。

リモートアドレス接続では、リモート AMQP エンドポイントのアドレスをアドレス空間にマッピングします。たとえば、AMQ Online エンドポイントを使用して接続し、アクセスするホスト **messaging.example.com** で AMQP サーバーが実行されているとします。リモートアドレス接続を有効にするには、アドレス空間コネクターを作成する必要があります。

メッセージのストアアンドフォワードには、アドレス転送の有効化が含まれます。まず、アドレス空間コネクターを作成する必要があります。次に、アドレスごとにアドレスフォワーダーを作成する必要があります。アドレス転送の詳細は、アドレス転送の例 を参照してください。

次の例は、アドレス空間コネクターを設定する方法を示しています。

2.10.1. SASL PLAIN を使用したアドレス空間コネクター

認証に相互 TLS を使用しない場合は、SASL PLAIN を使用できます。ユーザー名とパスワードがプレーンテキストとして送信されるため、TLS を有効にしないことはお勧めしません。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  connectors:
  - name: remote1 1
    endpointHosts: 2
    - host: messaging.example.com
      port: 5672
    - host: messaging2.example.com
    idleTimeout: 3
    maxFrameSize: 4
    tls: {} 5
    credentials: 6
      username:
        value: test
      password:
        valueFromSecret:
          name: password-secret
          key: password.txt
    role: 7
    addresses: 8
    - name: p1
      pattern: "prices/*"
    - name: p2
      pattern: "clients/*/1"
```

1 (必須) コネクターの名前を指定します。すべてのリモートアドレスには、コネクター名とスラッシュ (/) が接頭辞として付けられます。

2 (必須) このコネクターのエンドポイントのリストを指定します。このリストには少なくとも 1 つのエントリーが含まれている必要があり、追加のエントリーはフェイルオーバーに使用されます。特に指定されていない場合、**port** フィールドの値は、AMQP (または TLS が有効な場合は AMQPS) の登録済み IANA ポートに設定されます。

3 (オプション) AMQP 接続のアイドルタイムアウト (秒)。0 はアイドルタイムアウトを無効にします。

4 (オプション) AMQP 接続の最大フレームサイズ。

5 (オプション) TLS を有効にします。コネクターは、デフォルトでグローバルルート CA を信頼します。カスタム CA を使用するには、**caCert** フィールドの値を指定します。

6 (省略可能) このコネクターに使用するユーザー名とパスワードの認証情報を指定します。値は、インラインで指定するか、シークレット内の場所を指定するオプションのキーと共にシークレットを参照して指定できます。シークレットは、**system:serviceaccounts:_amq-online-infra_** グループで読み取りできる必要があります。

7 (オプション) コネクターのロール。有効な値は、"normal"、"edge"、および "route-container" (デフォルト値) です。

8

(必須) リモートエンドポイントで公開されるアドレスに一致するパターンのリストを指定します。パターンは、スラッシュ / で区切られた 1 つ以上のトークンで設定されます。トークンは、* 文

2.10.2. 相互 TLS を使用するアドレス空間コネクター

クライアント TLS 証明書を設定すると、SASL EXTERNAL を認証に使用できるようになります。証明書は、インラインで指定するか、秘密の参照を使用して指定できます。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
  connectors:
  - name: remote1 1
    endpointHosts: 2
    - host: messaging.example.com
      port: 5671
    tls:
      caCert: 3
        valueFromSecret:
          name: remote-certs
          key: ca.crt
      clientCert: 4
        valueFromSecret:
          name: remote-certs
          key: tls.crt
      clientKey: 5
        valueFromSecret:
          name: remote-certs
          key: tls.key
  addresses:
  - name: p1
    pattern: "*"
```

1 (必須) コネクターの名前を指定します。すべてのリモートアドレスには、コネクター名とスラッシュ (/) が接頭辞として付けられます。

2 (必須) このコネクターのエンドポイントのリストを指定します。このリストには少なくとも 1 つのエントリーが含まれている必要があり、追加のエントリーはフェイルオーバーに使用されます。特に指定されていない場合、**port** フィールドの値は、AMQP (または TLS が有効な場合は AMQPS) の登録済み IANA ポートに設定されます。

3 (オプション) リモート接続で信頼する CA 証明書を指定します。参照されるシークレットは、**system:serviceaccounts:_amq-online-infra_** グループで読み取りできる必要があります。

4 (オプション) 相互 TLS 認証に使用するクライアント証明書を指定します。参照されるシークレットは、**system:serviceaccounts:_amq-online-infra_** グループで読み取りできる必要があります。

5

(オプション) 相互 TLS 認証に使用するクライアント秘密鍵を指定します。参照されるシークレットは、**system:serviceaccounts:_amq-online-infra_** グループで読み取りできる必要がありま

2.11. コマンド行を使用したアドレス空間の作成

AMQ Online では、標準のコマンドラインツールを使用してアドレス空間を作成します。

手順

1. メッセージングテナントとしてログインします。

```
oc login -u developer
```

2. メッセージングアプリケーションのプロジェクトを作成します。

```
oc new-project myapp
```

3. アドレス空間定義を作成します。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-unlimited
```

4. アドレス空間を作成します。

```
oc create -f standard-address-space.yaml
```

5. アドレス空間のステータスを確認します。

```
oc get addressspace myspace -o jsonpath={.status.isReady}
```

前のコマンドが **true** を出力すると、アドレス空間を使用する準備が整います。

2.12. RED HAT AMQ コンソールを使用したアドレス空間の作成

さまざまな証明書プロバイダーを使用してアドレス空間のエンドポイントを設定したり、エンドポイントを作成してメッセージングアプリケーションが使用できるようにアドレス空間を作成したりなど、Red Hat AMQ コンソールを使用して新しいアドレス空間を作成できます。

注記

アドレス空間のエンドポイントを設定しないことを選択した場合、システムは、AMQPS および AMQP-WSS の OpenShift ルートとしてエンドポイントのデフォルトセットを作成し、システム生成 (自己署名) 証明書およびクラスターサービスで保護します。

手順

1. Red Hat AMQ コンソールにログインします。
   Red Hat AMQ コンソールへのアクセス方法の詳細は、Red Hat AMQ Console へのアクセス を参照してください。
2. **アドレス空間の作成** をクリックします。インスタンスの作成ウィザードが開きます。
3. 必須フィールドに入力し、完了したら、**Finish** をクリックして新しいアドレス空間を作成します。

アドレス空間が正常に作成されたら、アドレス空間名をクリックして、新しく作成されたアドレス空間に関する情報 (メッセージングとアプリケーションの統計、エンドポイント情報など) を表示できます。

2.13. RED HAT AMQ コンソールを使用したアドレス空間に関連付けられたアドレス空間プランの変更

Red Hat AMQ コンソールを使用して、アドレス空間に関連付けられているアドレス空間プランを変更できます。

前提条件

- アドレス空間をすでに作成してある。詳細は Red Hat AMQ コンソールを使用したアドレス空間の作成 を参照してください。

手順

1. Red Hat AMQ コンソールにログインします。詳細は Red Hat AMQ コンソールへのアクセス を参照してください。
2. アドレス空間計画を変更するアドレス空間を見つけます。
3. 右端の列で、縦の省略記号アイコンをクリックし、**編集** を選択します。編集ウィンドウが開きます。
4. **アドレス空間プラン** フィールドで、リストから別のプランを選択し、**確認** をクリックします。そのアドレス空間のアドレス空間プランが変更されます。

2.14. RED HAT AMQ コンソールを使用したアドレス空間に関連付けられた認証サービスの変更

Red Hat AMQ コンソールを使用して、アドレス空間に関連付けられている認証サービスを変更できます。

前提条件

- アドレス空間をすでに作成してある。詳細は Red Hat AMQ コンソールを使用したアドレス空間の作成 を参照してください。

手順

1. Red Hat AMQ コンソールにログインします。詳細は Red Hat AMQ コンソールへのアクセス を参照してください。
2. 認証サービスを変更するアドレス空間を見つけます。

3. 右端の列で、縦の省略記号アイコンをクリックし、**編集** を選択します。編集ウィンドウが開きます。

4. **Authentication Service** フィールドで、リストから別の認証サービスを選択し、**Confirm** をクリックします。そのアドレス空間の認証サービスが変更されます。

2.15. RED HAT AMQ コンソールを使用したアドレス空間の削除

Red Hat AMQ コンソールを使用して、既存のアドレス空間を削除できます。

手順

1. Red Hat AMQ コンソールにログインします。
Red Hat AMQ コンソールへのアクセス方法の詳細は、Red Hat AMQ Console へのアクセス を参照してください。

2. 削除するアドレス空間を見つけます。

3. 右端の列で、縦の省略記号アイコンをクリックし、**削除** を選択します。削除確認ウィンドウが開きます。

4. **削除** をクリックして、選択内容を確認します。アドレス空間が削除されます。

2.16. アドレス空間情報を取得するためのコマンド例

次の表に、アドレス空間情報を取得するためのコマンドを示します。

表2.1 アドレス空間情報取得コマンド一覧

取得方法	実行コマンド
アドレス空間のステータス	**oc get addressspace myspace -o jsonpath={.status.isReady}**
メッセージングエンドポイントの base64 でエンコードされた PEM 証明書	**oc get addressspace myspace -o 'jsonpath={.status.caCert}'**
メッセージングエンドポイントのホスト名	**oc get addressspace myspace -o 'jsonpath={.status.endpointStatuses[?(@.name=="messaging")].externalHost}'**

2.17. コマンドラインを使用したアドレス空間の置換

プラン、エンドポイント、またはネットワークポリシーを変更するために、または **certBundle** 証明書プロバイダーを使用している場合は証明書を置き換えるために、アドレス空間を置き換えることができます。現在のアドレスのセットが新しいクォータ内に収まる場合に、プランを変更すると、AMQ Online は新しいプランを適用しようとします。そうでない場合は、**AddressSpace** リソースでエラーが発生します。

手順

1. メッセージングテナントとしてログインします。

```
oc login -u developer
```

2. メッセージングアプリケーションのプロジェクトを選択します。

```
oc project myapp
```

3. アドレス空間定義を更新します。

```
apiVersion: enmasse.io/v1beta1
kind: AddressSpace
metadata:
  name: myspace
spec:
  type: standard
  plan: standard-small
```

4. アドレス空間を置き換えます。

```
oc replace -f standard-address-space-replace.yaml
```

5. アドレス空間のステータスを確認します。

```
oc get addressspace myspace -o jsonpath={.status.isReady}
```

上記のコマンドが **true** を出力すると、アドレス空間が使用可能になります。

第3章 アドレスの管理

AMQ Online は、OpenShift コマンドラインツールと Red Hat AMQ コンソールを使用したアドレスの管理をサポートするように設定されています。**アドレス** リソースは、**oc** を使用して他の OpenShift API リソースと同様に管理できます。

3.1. アドレス

アドレスは、アドレス空間の一部であり、メッセージを送受信するための宛先を表します。アドレスには、そのアドレスとの間でメッセージを送受信するセマンティクスを定義するタイプがあります。

AMQ Online で使用できるアドレスの種類は、アドレス空間の種類によって異なります。

3.2. アドレスプラン

アドレスは、そのアドレスのリソース使用法を記述するアドレスプランで設定されます。アドレスプランはサービス管理者が設定しており、AMQ Online のインストール設定により異なる場合があります。作成できるアドレスの数と利用可能なプランは、アドレス空間プランによって適用されるクォータによって異なります。

一部のアドレスタイプでは、**plan** フィールドの変更もサポートされています。**standard** アドレス空間の **queue**、**anycast**、および **multicast** アドレスタイプは、新しいプランが許可されたクォータを超えない限り、プランの変更をサポートします。キューの場合、アドレスはブローカー間で動的に移行されるため、メッセージの並べ替えが発生する可能性があります。

3.2.1. アドレスの例

```
apiVersion: enmasse.io/v1beta1
kind: Address
metadata:
  name: myspace.myqueue (1)
spec:
  address: myqueue (2)
  type: queue (3)
  plan: standard-small-queue (4)
```

(1) アドレス名には、アドレス空間名とドットを接頭辞として付ける必要があります。アドレス名には、英数字のみを含めることができます。

(2) アドレスは、このアドレスリソースが表すメッセージングアドレスです。

(3) アドレスタイプは、このアドレスのセマンティクスを決定します。

(4) アドレスプランは、アドレスのリソース使用量を記述します。利用可能なプランを表示する方法の詳細は、利用可能なアドレスプランの一覧表示 を参照してください。

3.2.2. トピックとサブスクリプションアドレスの例

トピックとサブスクリプションを使用する場合、次の例に示すように、サブスクリプションは **topic:** フィールドを使用して所属するトピックを参照します。

```
apiVersion: enmasse.io/v1beta1
```

```
kind: Address
metadata:
  name: myspace.mytopic
spec:
  address: mytopic
  type: topic
  plan: standard-small-topic
```

```
apiVersion: enmasse.io/v1beta1
kind: Address
metadata:
  name: myspace.mysub
spec:
  address: mysub
  type: subscription
  plan: standard-small-subscription
  topic: mytopic 1
```

1 このサブスクリプションが参照するトピックのアドレス。

3.2.3. アドレス TTL 制限の例

```
apiVersion: enmasse.io/v1beta1
kind: Address
metadata:
  name: myspace.myqueue
spec:
  address: myqueue
  type: queue
  plan: standard-small-queue
  messageTtl: 1
    minimim: 30000
    maximum: 300000
```

1 (オプション) メッセージの存続可能時間 (TTL) を制限します。アドレスタイプの **queue** と **topic** にのみ適用されます。

messageTtl フィールドは、キューまたはトピックに書き込まれたメッセージの有効な **absolute-expiry-time** を制限するために使用されます。**maximum** と **minimum** の値はミリ秒単位で定義されます。システムは、次の値に基づいて特定のアドレスへの着信メッセージの TTL 値を調整します。

- TTL 値が **maximum** の値より大きいアドレスにメッセージが到着すると、システムはメッセージの TTL を最大値に変更します。
- TTL 値が **minimum** の値より小さいアドレスにメッセージが到着すると、システムはメッセージの TTL を最小値に変更します。

TTL が定義されていない状態で到着したメッセージは、TTL 値が無限大であると見なされます。

期限切れのメッセージは、**queue**、**subscription**、または一時トピックサブスクリプションから定期的に自動削除されます。これらのメッセージは失われます。これは 30 秒ごとに発生します。

TTL 制限は、アドレスプランによって課される場合もあります。プランとアドレスの両方に TTL 制限が課されている場合には、アドレス TTL 制限は TTL 制限をさらに絞りこむ以外手段はありません。アドレス **status** セクションには、有効な TTL 値が表示されます。

3.2.4. アドレス転送の例

フォワーダーを使用すると、以下が可能です。

- ローカルアドレスから AMQ Online の外部のリモート AMQP サーバーにメッセージを自動的に転送する、または
- リモート AMQP サーバーからローカルアドレスにメッセージを転送する。

アドレスフォワーダーを使用するには、最初にアドレス空間のリモート AMQP サーバーへのコネクターを設定する必要があります。アドレス空間コネクターの詳細は、アドレス空間コネクターの例 を参照してください。

アドレス転送は、**standard** アドレス空間タイプでのみサポートされており、**queue** および **subscritpion** アドレスタイプでのみサポートされています。**queue** アドレスタイプを使用すると、メッセージをリモート AMQP サーバーに転送したり、リモート AMQP サーバーからローカルキューに転送したりできます。**subscription** アドレスタイプでは、リモート AMQP アドレスへのフォワーダーを作成できますが、メッセージをサブスクリプションにコピーするフォワーダーを作成することはできません。つまり、**subscription** アドレスタイプは、例に示すように、**out** 方向の転送のみをサポートします。

次の例では、コネクター **remote1** がアドレス空間用に設定されていることを前提としています。

3.2.4.1. ローカルキューからリモート AMQP サーバーへのメッセージの転送

この例では、**myqueue** 内のメッセージは、アドレスが **clients/me/1** のリモート AMQP サーバーに転送されます。

```
apiVersion: enmasse.io/v1beta1
kind: Address
metadata:
  name: myspace.myqueue
spec:
  address: myqueue
  type: queue
  plan: standard-small-queue
  forwarders:
  - name: f1 (1)
    remoteAddress: remote1/clients/me/1 (2)
    direction: out (3)
```

(1) (必須) 一意の ID を確保するために使用されるフォワーダーの名前を指定します。

(2) (必須) メッセージの転送先のリモートアドレスを指定します。アドレスにはコネクター名の接頭辞を付ける必要があり、コネクターで定義されたアドレス一致パターンと同一である必要があります。

(3) (必須) メッセージフローの方向 (**out** または **in**) を指定します。**out** の値は、メッセージをリモートエンドポイントに転送します。**in** の値は、リモートエンドポイントからメッセージを転送します。

3.2.4.2. リモート AMQP サーバーからローカルキューへのメッセージの転送

この例では、リモート AMQP サーバー上のアドレス **prices/milk** からメッセージを受信します。その後、メッセージはローカルキュー **myqueue** に移動されます。

```
apiVersion: enmasse.io/v1beta1
kind: Address
metadata:
  name: myspace.myqueue
spec:
  address: myqueue
  type: queue
  plan: standard-small-queue
  forwarders:
  - name: f1 (1)
    remoteAddress: remote1/prices/milk (2)
    direction: in (3)
```

(1) (必須) 一意の ID を確保するために使用されるフォワーダーの名前を指定します。

(2) (必須) メッセージの転送先のリモートアドレスを指定します。アドレスにはコネクター名の接頭辞を付ける必要があり、コネクターで定義されたアドレス一致パターンと同一である必要があります。

(3) (必須) メッセージフローの方向 (**out** または **in**) を指定します。**out** の値は、メッセージをリモートエンドポイントに転送します。**in** の値は、リモートエンドポイントからメッセージを転送します。

3.3. コマンドラインを使用した利用可能なアドレスプランの一覧表示

queue などのアドレスタイプで使用可能なアドレスプランを一覧表示できます。

手順

1. メッセージングテナントとしてログインします。

```
oc login -u developer
```

2. リストされたアドレスプランを使用してスキーマを取得します (ブローカーアドレス空間タイプは **standard** から **brokered** に置き換えます)。

```
oc get addressspaceschema standard -o 'jsonpath={.spec.addressTypes[?(@.name=="queue")].plans[*].name}'
```

3.4. コマンドラインを使用したアドレスの作成

コマンドラインを使用してアドレスを作成できます。

手順

1. アドレス定義を作成します。

```
apiVersion: enmasse.io/v1beta1
kind: Address
metadata:
   name: myspace.myqueue
spec:
   address: myqueue
   type: queue
   plan: standard-small-queue
```

注記

異なるアドレス空間からのアドレスが競合しないようにするには、名前の前にアドレス空間名を付ける必要があります。

2. アドレスを作成します。

```
oc create -f standard-small-queue.yaml
```

3. アドレスを一覧表示します。

```
oc get addresses -o yaml
```

3.5. RED HAT AMQ コンソールを使用したアドレスの作成

Red Hat AMQ コンソールを使用して新しいアドレスを作成できます。作成できるアドレスのタイプは、アドレス空間のタイプによって決まります。

前提条件

- アドレス空間を作成しておく。詳しくは アドレス空間の作成 を参照してください。

手順

1. Red Hat AMQ コンソールにログインします。詳細は Red Hat AMQ コンソールへのアクセス を参照してください。
2. 新しいアドレスを作成するアドレス空間のアドレス空間リンクをクリックします。
3. **Create** をクリックします。Create new address ウィンドウが開きます。
4. 名前を入力し、アドレスの種類を選択します。**subscription** を選択した場合は、**topic** リストから、サブスクリプションを作成するトピック名を選択します。
5. **Next** をクリックします。
6. プランを選択し、**Next** をクリックします。
7. **Create** をクリックします。アドレスは Red Hat AMQ コンソールに表示されます。

3.6. コマンドラインを使用したアドレスの置換

手順

1. アドレス定義を作成します。

```
apiVersion: enmasse.io/v1beta1
kind: Address
metadata:
    name: myspace.myqueue
spec:
    address: myqueue
    type: queue
    plan: standard-xlarge-queue
```

2. アドレスを次のように置き換えます。

```
oc replace -f standard-xlarge-queue.yaml
```

3. アドレスを一覧表示します。

```
oc get addresses -o yaml
```

第4章 RED HAT AMQ コンソールの使用

Red Hat AMQ コンソールを使用して、アドレス空間の 作成 と アドレス空間の削除、アドレスの作成、メッセージと接続統計の表示 などのタスクを実行できます。

4.1. RED HAT AMQ コンソールのユーザー権限

Red Hat AMQ コンソールは、OpenShift RBAC パーミッションモデル を使用します。

Red Hat AMQ コンソールを使用するには、**OpenShift** ユーザーがアドレス空間と **アドレス** リソースへのアクセスを許可するロールが必要です。たとえば、編集アクセスの場合は、関連付けられたロールオブジェクトに **create**、**update**、および **delete** パーミッションを付与する必要があり、表示のみのアクセスの場合は、**list** パーミッションを付与する必要があります。

AMQ Online のサンプルロールの詳細は、AMQ Online のサンプルロール を参照してください。

4.2. RED HAT AMQ コンソールへのアクセス

前提条件

- OpenShift Container Platform 3.x では以下のコマンドを実行して Red Hat AMQ コンソールのホスト名を取得しておく。

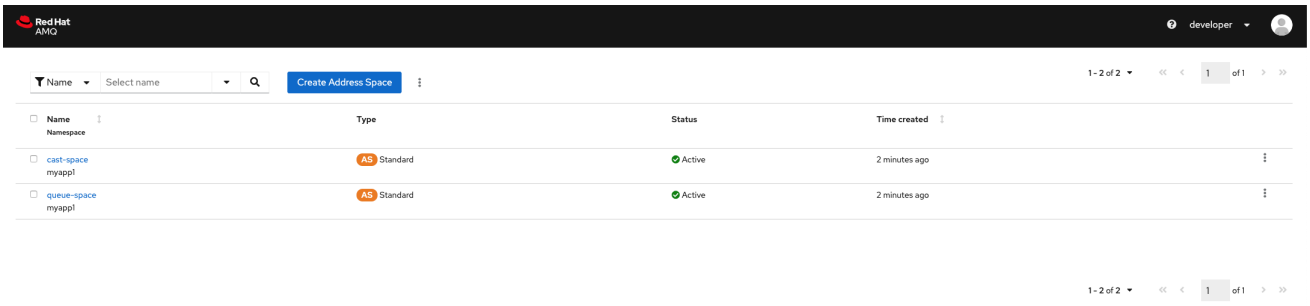
```
oc get routes console -o jsonpath={.spec.host}
```

- OpenShift Container Platform 4.x では以下のコマンドを実行して Red Hat AMQ コンソールのホスト名を取得しておく。

```
oc get consolelink -l app=enmasse -o jsonpath={.spec.href}
```

手順

1. Web ブラウザーで **https://console-host-name** に移動します。**console-host-name** は Red Hat AMQ コンソールのホスト名に置き換えます。
2. OpenShift ユーザー認証情報でログインします。Red Hat AMQ コンソールが開き、管理できるすべてのアドレス空間が一覧表示されます。アドレス空間の作成については Red Hat AMQ コンソールを使用したアドレス空間の作成 を参照してください。



4.3. RED HAT AMQ コンソールを使用したメッセージおよびアプリケーション接続統計の表示

前提条件

- Red Hat AMQ コンソールにログインしておく。

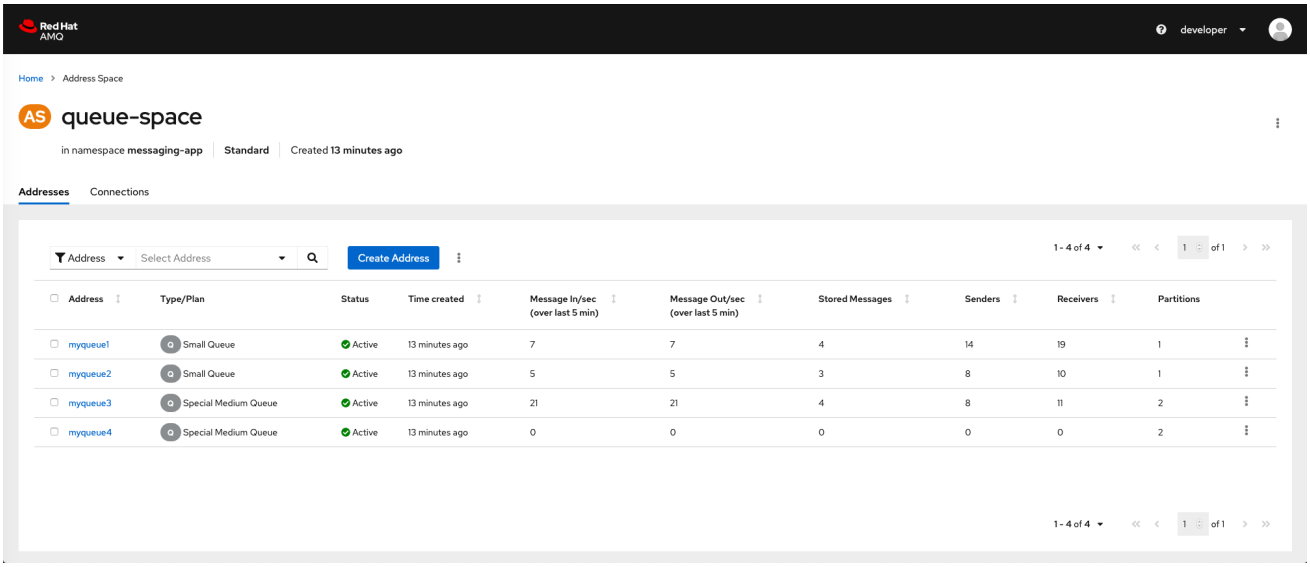


表4.1 メッセージ統計参照表

表示方法	アドレスページでの操作
アドレスの種類	2 列目の **タイプ/プラン** の最初のアイコンを参照してください。
アドレスプラン	2 列目の **タイプ/プラン** のアイコンに続く文字列を参照してください。
アドレスのステータス	3 列目の **ステータス** を参照してください。
1 秒あたりの受信メッセージ数 (過去 5 分間で計算)	**受信メッセージ数/秒** を参照してください。
1 秒あたりの送信メッセージ数 (過去 5 分間で計算)	**送信メッセージ数/秒** を参照してください
キューとトピックのアドレスタイプのみ: ブローカまたは複数のブローカに保存されているメッセージの数	**保存されたメッセージ**
接続されている送信者の数	**送信者** を参照してください。
割り当てられているレシーバーの数	**レシーバー** を参照してください。
標準アドレス空間のみ:1 秒あたりのメッセージ配信数	目的のアドレスをクリックすると、そのアドレスのリンクページが表示されます。Delivery Rate 欄を参照してください。

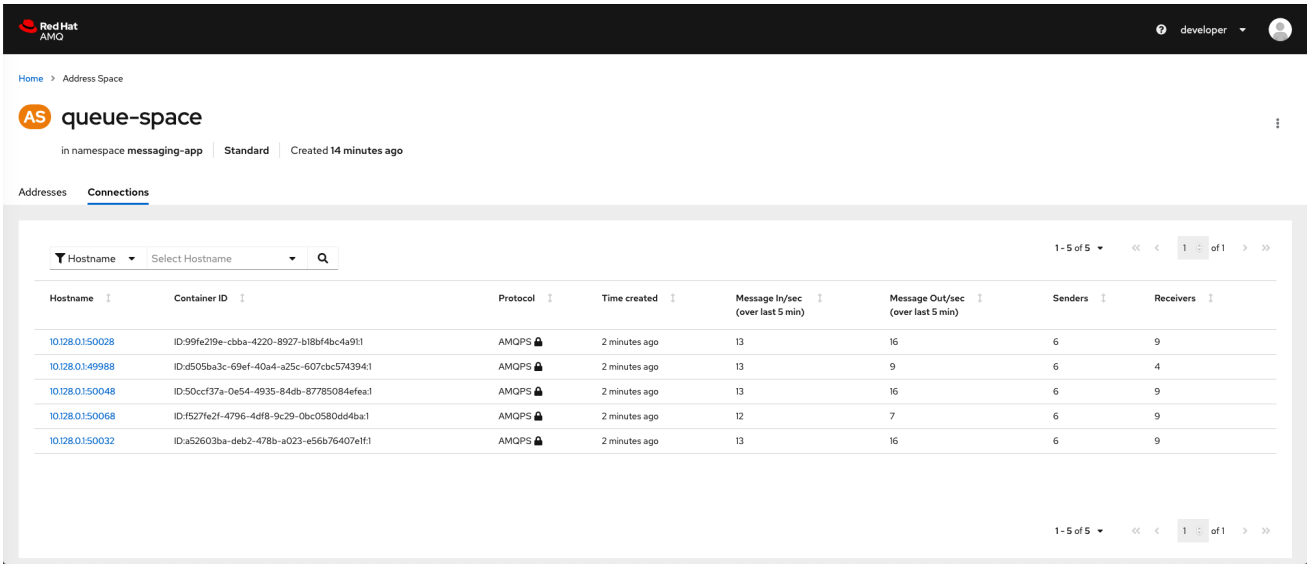
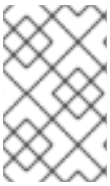


表4.2 アプリケーション接続統計参照表

表示方法	接続ページでの操作
1秒あたりの受信メッセージ数 (過去 5 分間で計算)	**受信メッセージ数/秒** を参照してください。
標準アドレス空間のみ:1秒あたりの送信メッセージ数 (過去 5 分間で計算)	**送信メッセージ数/秒** を参照してください
配信メッセージの総数	目的のホスト名をクリックして、送信者と受信者のリストを表示します。**Deliveries** 欄を参照してください。



注記

ブローカーアドレス空間の場合のみ、Connections ページの送信者数は **0** または **1** です。1つまたは複数の送信者が存在するとすぐに、実際の送信者数を反映するのではなく、**1** が表示されます。

4.4. RED HAT AMQ コンソールを使用したエンドポイント情報の表示

Red Hat AMQ コンソールを使用して、特定のアドレス空間に設定されたエンドポイントに関する情報を表示できます。この情報は、メッセージングアプリケーションを AMQ Online に接続するために必要です。

前提条件

- Red Hat AMQ コンソールにログインしておく。詳細は Red Hat AMQ コンソールへのアクセス を参照してください。

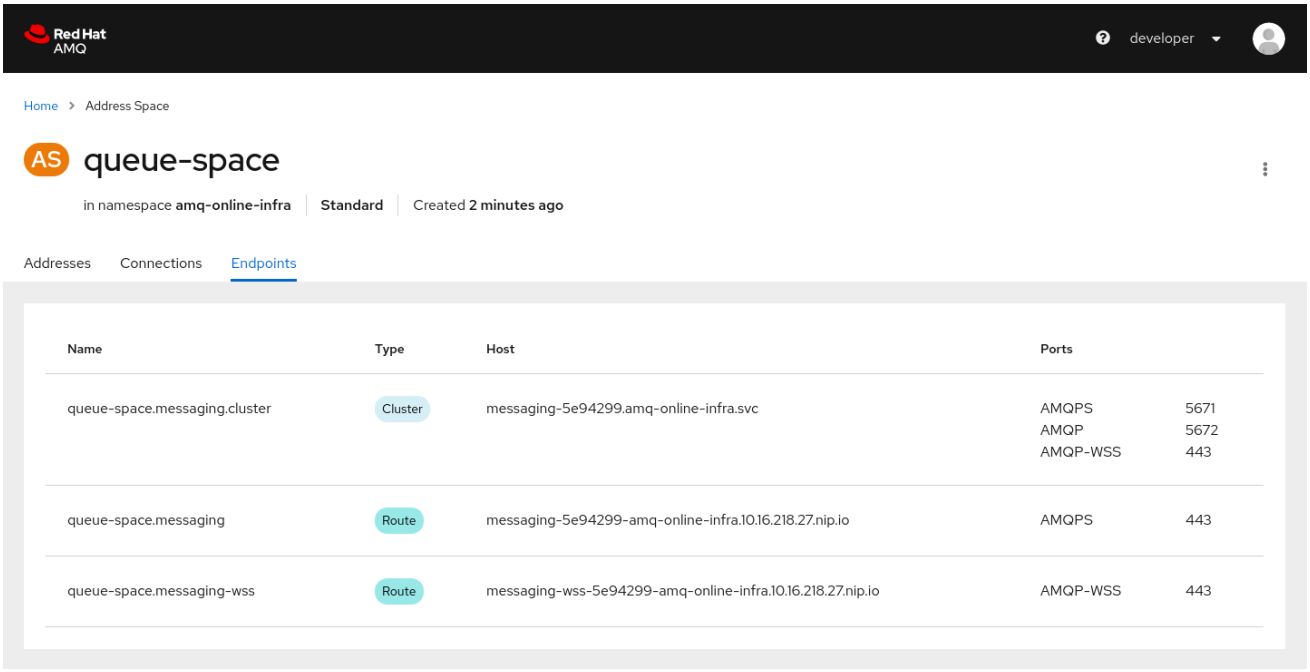


表4.3 メッセージングエンドポイント情報参照表

列	説明
Name	エンドポイントの名前を表示します。
Type	エンドポイントのタイプを表示します。有効な値は次のとおりです。 **cluster** クラスター IP アドレスを使用して OpenShift クラスターでアクセス可能なエンドポイント。 **route** クラスター外で使用可能な OpenShift ルート。 **loadbalancer** 外部ロードバランサーと統合する **LoadBalancer** サービス。 詳細は、次の OpenShift ドキュメントを参照してください。 ● ルート設定 ● ロードバランサーを使用した ingress クラスターの設定
ホスト	エンドポイントのホスト名を表示します。

列	説明
Ports	エンドポイントのポートプロトコル名とポート番号を表示します。有効なポート名は次のとおりです。 **AMQP** Advance Messaging Queuing Protocol **AMQPS** TLS 経由の Advance Messaging Queuing Protocol **AMQP-WS** AMQP-WebSocket プロトコル **AMQP-WSS** TLS 経由の AMQP-WebSocket プロトコル

4.5. キューとサブスクリプションのパージ

Red Hat AMQ コンソールを使用して、格納されているメッセージの **queue** または **subscription** アドレスタイプからすべてのメッセージを消去できます。

前提条件

- 保存されたメッセージを含むキューまたはサブスクリプションが必要です。

手順

1. Red Hat AMQ コンソールにログインします。詳細は Red Hat AMQ コンソールへのアクセス を参照してください。
2. **Addresses** ページに移動します。
3. パージするキューまたはサブスクリプションの横にあるチェックボックスをオンにします。
4. ページの上部にある縦の省略記号アイコンを右クリックし、**Purge** を選択します。キューまたはサブスクリプションが消去され、選択したキューまたはサブスクリプションの **保存** メッセージ数がゼロになります。

4.6. RED HAT AMQ コンソールを使用した接続の切断

アプリケーションが予期せずメッセージの処理を停止した場合に、Red Hat AMQ コンソールを使用してアプリケーションのメッセージング接続を強制的に切断できます。これは、アプリケーションをサービスに戻す便利な方法です。

接続を切断すると、アプリケーションの AMQ Online への接続が切断されることに注意してください。アプリケーションが切断するように設定されている場合には、アプリケーションは自動的に再接続する必要があります。

手順

1. Red Hat AMQ コンソールにログインします。詳細は Red Hat AMQ コンソールへのアクセス を参照してください。

2. **Connections** ページに移動します。
3. 単一の接続を閉じるには以下を実行します。
    a. 切断する接続を見つけます。
    b. 右端の列で、縦の省略記号アイコンをクリックし、**Close** を選択します。
    c. プロンプトが表示されたら、**Confirm** をクリックして、接続が切断されたことを確認します。接続は閉じられます。
4. 1回の操作で複数の接続を切断するには以下を実行します。
    a. 切断する接続の横にあるチェックボックスをオンにします。
    b. ページの上部にある縦の省略記号アイコンを右クリックし、**Close Selected** を選択します。
    c. プロンプトが表示されたら、**Confirm** をクリックして、接続が切断されたことを確認します。接続が切断されました。

第5章 ユーザーモデル

メッセージングクライアントは MessagingUser を使用して接続します。MessagingUser は、使用できるアドレスとそれらのアドレスで実行できる操作を制御する許可ポリシーを指定します。

ユーザーは **MessagingUser** リソースとして設定されます。ユーザーは、作成、削除、読み取り、更新、および一覧表示できます。

次の例は、**user-example1.yaml** ファイルを示しています。

```
apiVersion: user.enmasse.io/v1beta1
kind: MessagingUser
metadata:
  name: myspace.user1
spec:
  username: user1
  authentication:
    type: password
    password: cGFzc3dvcmQ= # Base64 encoded
  authorization:
    - addresses: ["myqueue", "queue1", "queue2", "topic*"]
      operations: ["send", "recv"]
    - addresses: ["anycast1"]
      operations: ["send"]
```

次のフィールドは必須です。

- **metadata.name**
- **metadata.namespace**
- **spec.authentication**
- **spec.authorization**

spec.authentication オブジェクトはユーザーの認証方法を定義し、**spec.authorization** はそのユーザーの承認ポリシーを定義します。

5.1. 認証

認証タイプでサポートされている値は **password** と **serviceaccount** です。**password** 認証タイプを使用する場合は、接続時にメッセージングクライアントが使用するユーザー名とパスワードを指定します。**serviceaccount** 認証タイプでは、特別な文字列 **@@serviceaccount@@** をユーザー名として使用し、OpenShift サービスアカウントトークンをパスワードとして使用します。

5.1.1. パスワード認証の種類

password タイプの場合には、追加フィールドの **パスワード** を、そのユーザーのパスワードの base64 エンコード値に設定する必要があります。リソースの読み取り時にパスワードは出力されません。

パスワードは、コマンドラインで base64 エンコードできます。たとえば、**my-password** をエンコードするには以下を実行します。

```
$ echo -n my-password | base64
bXktcGFzc3dvcmQ=
```

-

5.1.2. サービスアカウント認証タイプ

serviceaccount タイプの場合、**username** フィールドには、認証に使用される OpenShift サービスアカウント名が含まれている必要があります。メッセージングクライアントに接続するときは、文字列 **@@serviceaccount@@** をユーザー名として使用し、サービスアカウントトークンをパスワードとして使用します。アプリケーションで使用される AMQP クライアントは、SASL メカニズムタイプ **PLAIN** を使用するように設定する必要があります。

5.2. 承認

さらに、操作とアドレスを使用して承認ポリシーを定義できます。有効な操作は、**send**、**recv**、**view**、および **manage** です。

manage および **view** 操作は、アドレス空間内のすべてのアドレスに適用されます。

standard アドレス空間では、アドレスの末尾にアスタリスクワイルドカードを使用できます。アドレス **top*** は、アドレス **topic** および **topic/sub** と一致します。

brokered アドレス空間では、プラス記号とアスタリスクのワイルドカードをアドレスの末尾に使用して、スラッシュ区切り文字の後の 1 文字 (プラス記号) またはすべて文字 (アスタリスク) を検索できます。したがって、アドレス **topic/+** は **topic/sub** に一致しますが、**topic/s/sub** には一致しません。アドレス **topic/*** は、**topic/sub** および **topic/s/sub** と一致します。

5.3. ユーザーの管理

AMQ Online ユーザー管理は、**standard** の認証サービスを使用する場合にのみサポートされます。OpenShift では、OpenShift コマンドラインツールを使用してユーザーを管理できます。

前提条件

- アドレス空間 を作成しておく。

5.3.1. コマンドラインでのユーザーの作成

AMQ Online では、標準のコマンドラインツールを使用してユーザーを作成できます。

前提条件

- アドレス空間 を作成しておく。

手順

1. ユーザー定義ファイルのパスワードを正しく base64 エンコードするには、次のコマンドを実行します。

```
echo -n password | base64 #cGFzc3dvcmQ=
```

注記

このコマンドを実行するときは、必ず **-n** パラメーターを使用してください。このパラメーターを指定しないと、パスワードが正しくコード化されず、ログインの問題が発生します。

2. ユーザー定義をファイルに保存します。

```
apiVersion: user.enmasse.io/v1beta1
kind: MessagingUser
metadata:
  name: myspace.user1
spec:
  username: user1
  authentication:
    type: password
    password: cGFzc3dvcmQ= # Base64 encoded
  authorization:
    - addresses: ["myqueue", "queue1", "queue2", "topic*"]
      operations: ["send", "recv"]
    - addresses: ["anycast1"]
      operations: ["send"]
```

3. ユーザーと関連するユーザー権限を作成します。

```
oc create -f user-example1.yaml
```

4. ユーザーが作成されたことを確認します。

```
oc get messagingusers
```

5.3.2. コマンドラインを使用したユーザーの削除

ユーザーは、標準のコマンドラインツールを使用して削除できます。

前提条件

- アドレス空間を作成しておく。
- ユーザーを作成しておく。

手順

1. 現在のユーザーを一覧表示します。

```
oc get messagingusers
```

2. 目的のユーザーを削除します。

```
oc delete messaginguser myspace.user1
```

5.3.3. コマンドラインを使用したユーザー権限の管理

コマンドラインを使用して、既存のユーザーの権限を編集できます。

前提条件

- ユーザーを作成しておく。詳細は コマンドラインを使用したユーザーの作成 を参照してください。

手順

1. 権限を編集するユーザーを取得します。

```
oc get messaginguser myspace.user1 -o yaml > user-example1.yaml
```

2. 必要なアクセス許可を変更し、ファイルを保存します。
3. コマンドラインから次のコマンドを実行して、変更を適用します。

```
oc apply -f user-example1.yaml
```

新しいユーザー権限が適用されます。

第6章 アプリケーションの AMQ ONLINE への接続

次のクライアント例のいずれかを使用して、アプリケーションを AMQ Online に接続できます。

- AMQ Online Python
- AMQ Online JMS
- AMQ Online JavaScript
- AMQ Online C++
- AMQ Online .NET

OpenShift クラスターの外部からメッセージングサービスに接続するには、SNI を設定して TLS を使用し、アドレス空間の完全修飾ホスト名を指定する必要があります。使用するポートは 443 です。

サポートされるメッセージングプロトコルは、使用されるアドレス空間のタイプによって異なります。アドレス空間の種類の詳細は、アドレス空間 を参照してください。

6.1. 自己署名 CA 証明書の取得

AddressSpace エンドポイント設定で **selfsigned** 証明書プロバイダータイプを選択した場合に、メッセージングクライアントアプリケーションに接続するときに、**AddressSpace** サーバー証明書に署名した生成 CA が必要です。次の手順を使用して、**AddressSpace** から証明書を取得できます。

警告

実稼働環境で自己署名証明書を使用することはお勧めしません。

手順

1. メッセージングテナントとしてログインします。

```
oc login -u developer
```

2. **AddressSpace** から CA 証明書を取得します。
これにより、CA 証明書を含むファイルが PEM 形式で提供されます。

```
oc get addressspace myspace -n namespace -o jsonpath='{.status.caCert}{"\n"}' | base64 --decode > ca.crt
```

3. PKCS12 または JKS 形式のトラストストアが必要な場合は、次のコマンドを使用して生成します。
PKS の場合:

```
keytool -import -trustcacerts -alias root -file ca.crt -storetype pkcs12 -keystore ca.pkcs12 -storepass password -noprompt
```

JKS の場合:

```
keytool -import -trustcacerts -alias root -file ca.crt -storetype jks -keystore ca.jsk -storepass password -noprompt
```

6.2. クライアントの例

6.2.1. AMQ Online Python の例

次の AMQ Online Python の例を使用して、アプリケーションを AMQ Online に接続できます。この例では、**myqueue** という名前の **queue** タイプのアドレスを作成したと想定しています。

```
from __future__ import print_function, unicode_literals
from proton import Message
from proton.handlers import MessagingHandler
from proton.reactor import Container

class HelloWorld(MessagingHandler):
    def __init__(self, server, address):
        super(HelloWorld, self).__init__()
        self.server = server
        self.address = address

    def on_start(self, event):
        conn = event.container.connect(self.server)
        event.container.create_receiver(conn, self.address)
        event.container.create_sender(conn, self.address)

    def on_sendable(self, event):
        event.sender.send(Message(body="Hello World!"))
        event.sender.close()

    def on_message(self, event):
        print(event.message.body)
        event.connection.close()

Container(HelloWorld("amqps://_messaging-route-hostname_:443", "myqueue")).run()
```

6.2.1.1. 階層トピックでサブスクライバーを作成する際の既知の問題

AMQ Online で階層トピックにサブスクライバーを作成すると、ブローカーが代わりに競合するコンシューマーとしてサブスクライバーを作成するという既知の問題が存在します (トピックではなくキューのようにアドレスを処理します)。

この問題の回避策として、ソースで **トピック** ケイパビリティーを設定します。

手順

1. **simple_recv.py** ファイルで、**from proton.reactor import Container** を変更して **ReceiverOption** を追加します。

```
class CapabilityOptions(ReceiverOption):
    def apply(self, receiver):
        receiver.source.capabilities.put_object(symbol("topic"))
```

1. 次の行を変更して、**options=CapabilityOptions ()** を追加します。

```
def on_start(self, event):
        event.container.create_receiver(conn, self.address, options=CapabilityOptions())
```

6.2.2. AMQ Online JMS の例

次の AMQ Online JMS の例を使用して、アプリケーションを AMQ Online に接続できます。この例では、**myqueue** という名前の **queue** タイプのアドレスを作成したと想定しています。

```
package org.apache.qpid.jms.example;

import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.DeliveryMode;
import javax.jms.Destination;
import javax.jms.ExceptionListener;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.jms.MessageProducer;
import javax.jms.Session;
import javax.jms.TextMessage;
import javax.naming.Context;
import javax.naming.InitialContext;

public class HelloWorld {
    public static void main(String[] args) throws Exception {
        try {
            // The configuration for the Qpid InitialContextFactory has been supplied in
            // a jndi.properties file in the classpath, which results in it being picked
            // up automatically by the InitialContext constructor.
            Context context = new InitialContext();

            ConnectionFactory factory = (ConnectionFactory) context.lookup("myFactoryLookup");
            Destination queue = (Destination) context.lookup("myQueueLookup");

            Connection connection = factory.createConnection(System.getProperty("USER"),
System.getProperty("PASSWORD"));
            connection.setExceptionListener(new MyExceptionListener());
            connection.start();

            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

            MessageProducer messageProducer = session.createProducer(queue);
            MessageConsumer messageConsumer = session.createConsumer(queue);

            TextMessage message = session.createTextMessage("Hello world!");
            messageProducer.send(message, DeliveryMode.NON_PERSISTENT,
Message.DEFAULT_PRIORITY, Message.DEFAULT_TIME_TO_LIVE);
```

```
            TextMessage receivedMessage = (TextMessage) messageConsumer.receive(2000L);

            if (receivedMessage != null) {
                System.out.println(receivedMessage.getText());
            } else {
                System.out.println("No message received within the given timeout!");
            }

            connection.close();
        } catch (Exception exp) {
            System.out.println("Caught exception, exiting.");
            exp.printStackTrace(System.out);
            System.exit(1);
        }
    }

    private static class MyExceptionListener implements ExceptionListener {
        @Override
        public void onException(JMSException exception) {
            System.out.println("Connection ExceptionListener fired, exiting.");
            exception.printStackTrace(System.out);
            System.exit(1);
        }
    }
}
```

jndi.properties を使用:

```
connectionfactory.myFactoryLookup = amqps://messaging-route-hostname:443?
transport.trustAll=true&transport.verifyHost=false
queue.myQueueLookup = myqueue
```

6.2.3. AMQ Online JavaScript の例

次の AMQ Online JavaScript の例を使用して、アプリケーションを AMQ Online に接続できます。この例では、**myqueue** という名前の **queue** タイプのアドレスを作成したと想定しています。

```
var container = require('rhea');
container.on('connection_open', function (context) {
    context.connection.open_receiver('myqueue');
    context.connection.open_sender('myqueue');
});
container.on('message', function (context) {
    console.log(context.message.body);
    context.connection.close();
});
container.on('sendable', function (context) {
    context.sender.send({body:'Hello World!'});
    context.sender.detach();
});
container.connect({username: 'username', password: 'password', port:443, host:'messaging-route-
hostname', transport:'tls', rejectUnauthorized:false});
```

6.2.3.1. WebSocket を使用した AMQ Online JavaScript の例

```
var container = require('rhea');
var WebSocket = require('ws');

container.on('connection_open', function (context) {
    context.connection.open_receiver('myqueue');
    context.connection.open_sender('myqueue');
});
container.on('message', function (context) {
    console.log(context.message.body);
    context.connection.close();
});
container.on('sendable', function (context) {
    context.sender.send({body:'Hello World!'});
    context.sender.detach();
});

var ws = container.websocket_connect(WebSocket);
container.connect({username: 'username', password: 'password', connection_details:
ws("wss://messaging-route-hostname:443", ["binary"], {rejectUnauthorized: false})});
```

6.2.4. AMQ Online C++ の例

C++ クライアントには、Python と同じオプションを持つ同等の **simple_recv** と **simple_send** の例があります。ただし、C++ ライブラリーは URL に対して同じレベルの処理を実行しません。特に、TLS の使用を意味する **amqps://** は使用できないので、例を次のように変更する必要があります。

```
#include <proton/connection.hpp>
#include <proton/container.hpp>
#include <proton/default_container.hpp>
#include <proton/delivery.hpp>
#include <proton/message.hpp>
#include <proton/messaging_handler.hpp>
#include <proton/ssl.hpp>
#include <proton/thread_safe.hpp>
#include <proton/tracker.hpp>
#include <proton/url.hpp>

#include <iostream>

#include "fake_cpp11.hpp"

class hello_world : public proton::messaging_handler {
  private:
    proton::url url;

  public:
    hello_world(const std::string& u) : url(u) {}

    void on_container_start(proton::container& c) OVERRIDE {
        proton::connection_options co;
        co.ssl_client_options(proton::ssl_client_options());
        c.client_connection_options(co);
        c.connect(url);
    }
```

```
    void on_connection_open(proton::connection& c) OVERRIDE {
        c.open_receiver(url.path());
        c.open_sender(url.path());
    }

    void on_sendable(proton::sender &s) OVERRIDE {
        proton::message m("Hello World!");
        s.send(m);
        s.close();
    }

    void on_message(proton::delivery &d, proton::message &m) OVERRIDE {
        std::cout << m.body() << std::endl;
        d.connection().close();
    }
};

int main(int argc, char **argv) {
    try {
        std::string url = argc > 1 ? argv[1] : "messaging-route-hostname:443/myqueue";

        hello_world hw(url);
        proton::default_container(hw).run();

        return 0;
    } catch (const std::exception& e) {
        std::cerr << e.what() << std::endl;
    }

    return 1;
}
```

6.2.4.1. 階層トピックでサブスクライバーを作成する際の既知の問題

AMQ Online で階層トピックにサブスクライバーを作成すると、ブローカーが代わりに競合するコンシューマーとしてサブスクライバーを作成するという既知の問題が存在します (トピックではなくキューのようにアドレスを処理します)。

回避策として、ソースで **トピック** ケイパビリティーを設定する必要があります。

手順

- **topic_receive.cpp** ファイルで、次の例のようにコードを編集します。

```
void on_container_start(proton::container& cont) override {
        proton::connection conn = cont.connect(conn_url_);
        proton::receiver_options opts {};
        proton::source_options sopts {};

        sopts.capabilities(std::vector<proton::symbol> { "topic" });
        opts.source(sopts);

        conn.open_receiver(address_, opts);
    }
```

6.2.5. AMQ Online .NET の例

次の AMQ Online .NET の例を使用して、アプリケーションを AMQ Online に接続できます。この例では、**myqueue** という名前の **queue** タイプのアドレスを作成したと想定しています。

```
using System;
using Amqp;

namespace Test
{
   public class Program
   {
      public static void Main(string[] args)
      {
         String url = (args.Length > 0) ? args[0] : "amqps://messaging-route-hostname:443";
         String address = (args.Length > 1) ? args[1] : "myqueue";

         Connection.DisableServerCertValidation = true;
         Connection connection = new Connection(new Address(url));
         Session session = new Session(connection);
         SenderLink sender = new SenderLink(session, "test-sender", address);

         Message messageSent = new Message("Test Message");
         sender.Send(messageSent);

         ReceiverLink receiver = new ReceiverLink(session, "test-receiver", address);
         Message messageReceived = receiver.Receive(TimeSpan.FromSeconds(2));
         Console.WriteLine(messageReceived.Body);
         receiver.Accept(messageReceived);

         sender.Close();
         receiver.Close();
         session.Close();
         connection.Close();
      }
   }
}
```

付録A サブスクリプションの使用

AMQ Online は、ソフトウェアサブスクリプションを通じて提供されます。サブスクリプションを管理するには、Red Hat カスタマーポータルでアカウントにアクセスします。

アカウントへのアクセス

1. access.redhat.com に移動します。
2. アカウントがない場合は、作成します。
3. アカウントにログインします。

サブスクリプションのアクティベート

1. access.redhat.com に移動します。
2. **サブスクリプション** に移動します。
3. **Activate a subscription** に移動し、16 桁のアクティベーション番号を入力します。

zip および tar ファイルのダウンロード

zip または tar ファイルにアクセスするには、Red Hat カスタマーポータルを使用して、ダウンロードする関連ファイルを検索します。RPM パッケージを使用している場合は、この手順は必要ありません。

1. ブラウザーを開き、access.redhat.com/downloads で Red Hat カスタマーポータルの **Product Downloads** ページにログインします。
2. **JBOSS INTEGRATION AND AUTOMATION** カテゴリーの **Red Hat AMQ Online** エントリーを見つけます。
3. 目的の AMQ Online 製品を選択します。Software Downloads ページが開きます。
4. コンポーネントの **ダウンロード** リンクをクリックします。

パッケージ用のシステムの登録

RPM パッケージを Red Hat Enterprise Linux にインストールするには、システムが登録されている必要があります。zip または tar ファイルを使用している場合、この手順は必要ありません。

1. access.redhat.com に移動します。
2. **Registration Assistant** に移動します。
3. ご使用の OS バージョンを選択し、次のページに進みます。
4. システムターミナルで listed コマンドを使用して、登録を完了します。

詳細は、How to Register and Subscribe a System to the Red Hat Customer Portal を参照してください。

付録B メッセージングテナント向けの AMQ ONLINE リソース

次の表では、メッセージングテナントロールに関連する AMQ Online リソースについて説明します。

表B.1 AMQ Online メッセージングテナントリソーステーブル

リソース	説明
addresses	アドレスを指定します。
addressspaces	アドレス空間を指定します。
messagingusers	使用できるアドレスと、それらのアドレスに対して実行できる操作を制御する承認ポリシーを指定します。

改訂日時：2023-01-28 12:19:11 +1000