Red Hat AMQ 7.4

Installing and Managing AMQ Online on OpenShift Container Platform

For use with AMQ Online 1.2

Last Updated: 2019-08-13

Red Hat AMQ 7.4 Installing and Managing AMQ Online on OpenShift Container Platform

For use with AMQ Online 1.2

Legal Notice

Abstract

This guide describes how to install and manage AMQ Online.

Table of Contents

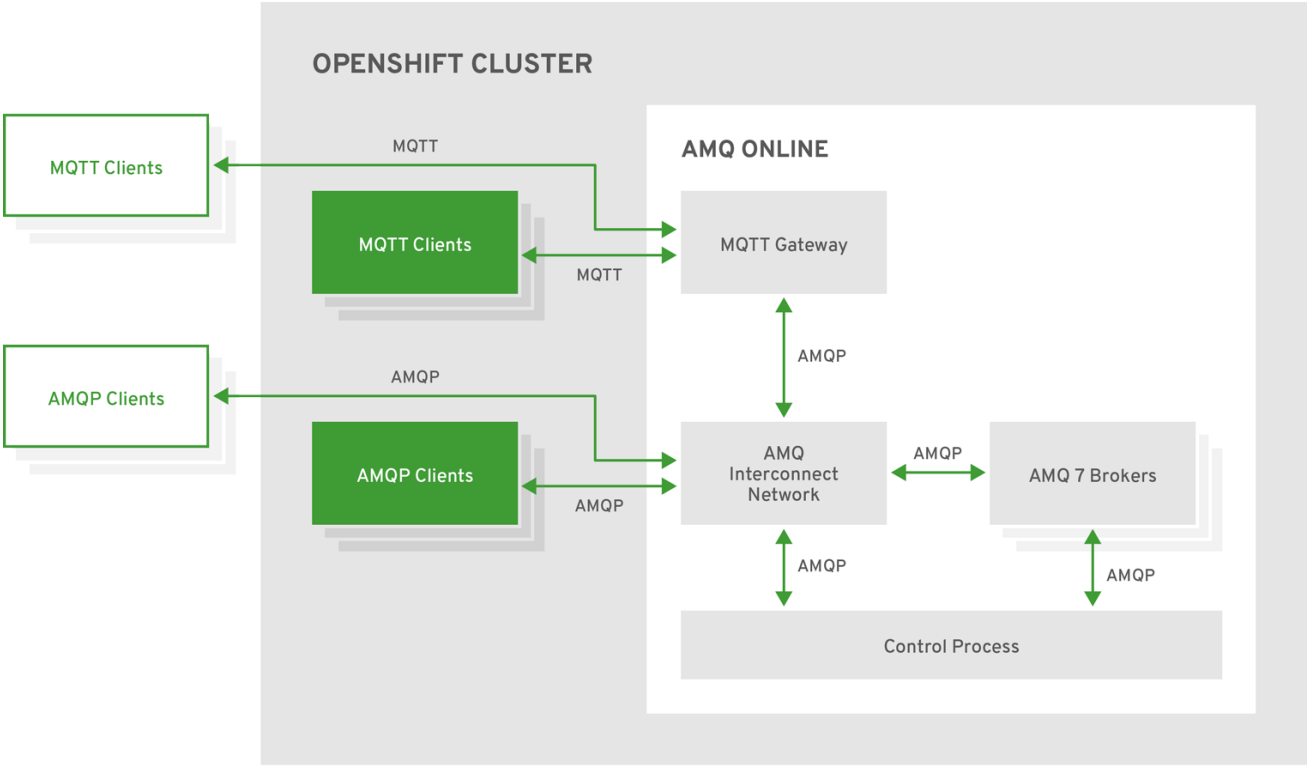
CHAPTER 1. INTRODUCTION

1.1. AMQ ONLINE OVERVIEW

Red Hat AMQ Online is an OpenShift-based mechanism for delivering messaging as a managed service. With Red Hat AMQ Online, administrators can configure a cloud-native, multi-tenant messaging service either in the cloud or on premise. Developers can provision messaging using the Red Hat AMQ Console. Multiple development teams can provision the brokers and queues from the Console, without requiring each team to install, configure, deploy, maintain, or patch any software.

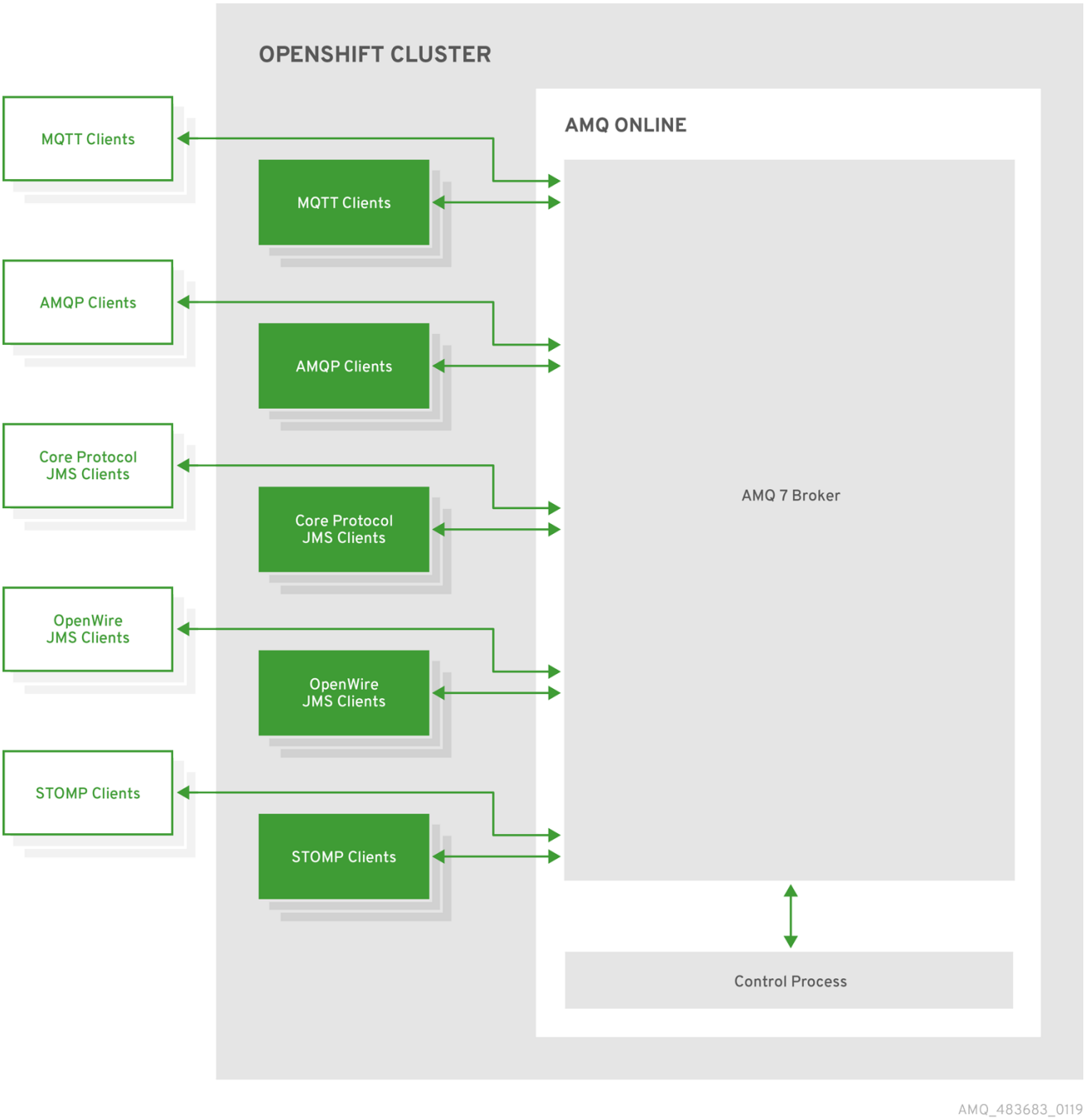
AMQ Online can provision different types of messaging depending on your use case. A user can request messaging resources by creating an address space. AMQ Online currently supports two address space types, standard and brokered, each with different semantics. The following diagrams illustrate the high-level architecture of each address space type:

Figure 1.1. Standard address space



AMQ_483683_0119

Figure 1.2. Brokered address space



1.2. SUPPORTED FEATURES

The following table shows the supported features for AMQ Online 1.2:

Table 1.1. Supported features reference table

Feature		Brokered address space	Standard address space
Address type	Queue	Yes	Yes
	Topic	Yes	Yes

Feature		Brokered address space	Standard address space
	Multicast	No	Yes
	Anycast	No	Yes
	Subscription	No	Yes
Messaging protocol	AMQP	Yes	Yes
	MQTT	Yes	Technology preview only
	CORE	Yes	No
	OpenWire	Yes	No
	STOMP	Yes	No
Transports	TCP	Yes	Yes
	WebSocket	Yes	Yes
Durable subscriptions	JMS durable subscriptions	Yes	No
	"Named" durable subscriptions	No	Yes
JMS	Transaction support	Yes	No
	Selectors on queues	Yes	No
	Message ordering guarantees (including prioritization)	Yes	No
Scalability	Scalable distributed queues and topics	No	Yes

1.3. AMQ ONLINE USER ROLES

AMQ Online users can be defined broadly in terms of two user roles: service administrator and messaging tenant. Depending on the size of your organization, these roles might be performed by the same person or different people.

The service administrator performs the initial installation and any subsequent upgrades. The service administrator might also deploy and manage the messaging infrastructure, such as monitoring the

routers, brokers, and administration components; and creating the address space plans and address plans. *Installing and Managing AMQ Online on OpenShift Container Platform* provides information about how to set up and manage AMQ Online as well as configure the infrastructure and plans as a service administrator.

The messaging tenant can request messaging resources, using both cloud-native APIs and tools. The messaging tenant can also manage the users and permissions of a particular address space within the messaging system as well as create address spaces and addresses. For more information about how to manage address spaces, addresses, and users, see *Using AMQ Online on OpenShift Container Platform*.

1.4. SUPPORTED CONFIGURATIONS

For more information about AMQ Online supported configurations see Red Hat AMQ 7 Supported Configurations.

1.5. DOCUMENT CONVENTIONS

1.5.1. Variable text

This document contains code blocks with variables that you must replace with values specific to your installation. In this document, such text is styled as italic monospace.

For example, in the following code block, replace ***my-namespace*** with the namespace used in your installation:

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

CHAPTER 2. INSTALLING AMQ ONLINE

AMQ Online can be installed by applying the YAML files using the OpenShift Container Platform command-line interface, or by running the Ansible playbook.

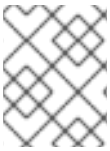
Prerequisites

- To install AMQ Online, the OpenShift Container Platform command-line interface (CLI) is required.
  - For more information about how to install the CLI on OpenShift 3.x, see the OpenShift Container Platform 3.11 documentation.
  - For more information about how to install the CLI on OpenShift 4.1, see the OpenShift Container Platform 4.1 documentation.
- An OpenShift cluster is required.
- A user on the OpenShift cluster with **cluster-admin** permissions is required to set up the required cluster roles and API services.

2.1. DOWNLOADING AMQ ONLINE

Procedure

- Download and extract the **amq-online-install.zip** file from the AMQ Online download site .



NOTE

Although container images for AMQ Online are available in the Red Hat Container Catalog, we recommend that you use the YAML files provided instead.

2.2. INSTALLING AMQ ONLINE USING A YAML BUNDLE

The simplest way to install AMQ Online is to use the predefined YAML bundles.

Procedure

1. Log in as a user with **cluster-admin** privileges:

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

2. (Optional) If you want to deploy to a project other than **amq-online-infra** you must run the following command and substitute **amq-online-infra** in subsequent steps:

```
sed -i 's/amq-online-infra/my-project/' install/bundles/amq-online/*.yaml
```

3. Create the project where you want to deploy AMQ Online:

```
oc new-project amq-online-infra
```

4. Change the directory to the location of the downloaded release files.

5. Deploy using the **amq-online** bundle:

```
oc apply -f install/bundles/amq-online
```

6. (Optional) Install the example plans and infrastructure configuration:

```
oc apply -f install/components/example-plans
```

7. (Optional) Install the example roles:

```
oc apply -f install/components/example-roles
```

8. (Optional) Install the **standard** authentication service:

```
oc apply -f install/components/example-authservices/standard-authservice.yaml
```

2.3. INSTALLING AMQ ONLINE USING ANSIBLE

Installing AMQ Online using Ansible requires creating an inventory file with the variables for configuring the system. Example inventory files can be found in the **ansible/inventory** folder.

The following example inventory file enables a minimal installation of AMQ Online:

```
[enmasse]
localhost ansible_connection=local

[enmasse:vars]
namespace=enmasse-infra
enable_rbac=False
api_server=True
service_catalog=False
register_api_server=True
keycloak_admin_password=admin
authentication_services=["standard"]
monitoring_namespace=enmasse-monitoring
monitoring_operator=False
monitoring=False
```

The following Ansible configuration settings are supported:

Table 2.1. Ansible configuration settings

Name	Description	Default value	Required
namespace	Specifies the project where AMQ Online is installed.	Not applicable	yes
enable_rbac	Specifies whether to enable RBAC authentication of REST APIs	True	no

Name	Description	Default value	Required
service_catalog	Specifies whether to enable integration with the Service Catalog	False	no
authentication_services	Specifies the list of authentication services to deploy. Supported values are **none** and **standard**.	**none**	no
keycloak_admin_password	Specifies the admin password to use for the **standard** authentication service Red Hat Single Sign-On instance	Not applicable	yes (if **standard** authentication service is enabled)
api_server	Specifies whether to enable the REST API server	True	no
register_api_server	Specifies whether to register the API server with OpenShift master	False	no
secure_api_server	Specifies whether to enable mutual TLS for the API server	False	no
install_example_plans	Specifies whether to install example plans and infrastructure configurations	True	no
monitoring_namespace	Specifies the project where AMQ Online monitoring is installed.	Not applicable	yes
monitoring_operator	Specifies whether to install the monitoring infrastructure	Not applicable	no
monitoring	Specifies whether to install the service monitors, Prometheus rules, and Grafana dashboards for monitoring AMQ Online	Not applicable	no

Procedure

1. Create an inventory file.
2. Run the Ansible playbook:

```
ansible-playbook -i inventory-file ansible/playbooks/openshift/deploy_all.yml
```

CHAPTER 3. CONFIGURING AMQ ONLINE

3.1. SERVICE CONFIGURATION RESOURCES AND DEFINITION

The service operator configures AMQ Online by defining resources constituting the "service configuration". This configuration contains instances of the following resource types:

- **AuthenticationService** - Describes an authentication service instance used to authenticate messaging clients.
- **AddressSpacePlan** - Describes the messaging resources available for address spaces using this plan, such as the available address plans and the amount of router and broker resources that can be used.
- **AddressPlan** - Describes the messaging resources consumed by a particular address using this plan, such as what fraction of routers and brokers an address will use and other properties that should be set for multiple addresses.
- **StandardInfraConfig** - Describes the router and broker configuration for the **standard** address space type such as memory limits, storage capacity, affinity, and more.
- **BrokeredInfraConfig** - Describes the broker configuration for the **brokered** address space type such as memory limits, storage capacity, affinity, and more.

When created, these resources define the configuration that is available to the messaging tenants.

The following diagram illustrates the relationship between the different service configuration resources (green) and how they are referenced by the messaging tenant resources (blue).

3.2. MINIMAL SERVICE CONFIGURATION

Configuring AMQ Online for production takes some time and consideration. The following procedure will get you started with a minimal service configuration. For a more complete example, navigate to the **install/components/example-plans** folder of the AMQ Online distribution. All of the commands must be run in the namespace where AMQ Online is installed.

Procedure

1. Save the example configuration:

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
```

```
  name: default
spec: {}
---
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: standard-small-queue
spec:
  addressType: queue
  resources:
    router: 0.01
    broker: 0.1
---
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: standard-small
spec:
  addressSpaceType: standard
  infraConfigRef: default
  addressPlans:
  - standard-small-queue
  resourceLimits:
    router: 2.0
    broker: 3.0
    aggregate: 4.0
---
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: none-authservice
spec:
  type: none
```

2. Apply the example configuration:

```
oc apply -f service-config.yaml
```

3.3. ADDRESS SPACE PLANS

Address space plans are used to configure quotas and control the resources consumed by address spaces. Address space plans are configured by the AMQ Online service operator and are selected by the messaging tenant when creating an address space.

AMQ Online includes a default set of plans that are sufficient for most use cases.

Plans are configured as custom resources. The following example shows a plan for the standard address space:

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: restrictive-plan
  labels:
    app: enmasse
```

```
spec:
  displayName: Restrictive Plan
  displayOrder: 0
  infraConfigRef: default 1
  shortDescription: A plan with restrictive quotas
  longDescription: A plan with restrictive quotas for the standard address space
  addressSpaceType: standard 2
  addressPlans: 3
  - small-queue
  - small-anycast
  resourceLimits: 4
    router: 2.0
    broker: 2.0
    aggregate: 2.0
```

1. A reference to the **StandardInfraConfig** (for the **standard** address space type) or the **BrokeredInfraConfig** (for the **brokered** address space type) describing the infrastructure deployed for address spaces using this plan.
2. The address space type this plan applies to, either **standard** or **brokered**.
3. A list of address plans available to address spaces using this plan.
4. The maximum number of routers (**router**) and brokers (**broker**) for address spaces using this plan. For the **brokered** address space type, only the **broker** field is required.

The other fields are used by the Red Hat AMQ Console UI. Note the field **spec.infraConfigRef**, which points to an infrastructure configuration that must exist when an address space using this plan is created. For more information about infrastructure configurations, see Infrastructure configuration.

3.4. CREATING ADDRESS SPACE PLANS

Procedure

1. Log in as a service admin:

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

2. Select the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Create an address space plan definition:

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: restrictive-plan
  labels:
    app: enmasse
spec:
  displayName: Restrictive Plan
  displayOrder: 0
```

```
infraConfigRef: default
shortDescription: A plan with restrictive quotas
longDescription: A plan with restrictive quotas for the standard address space
addressSpaceType: standard
addressPlans:
- small-queue
- small-anycast
resourceLimits:
  router: 2.0
  broker: 2.0
  aggregate: 2.0
```

4. Create the address space plan:

```
oc create -f restrictive-plan.yaml
```

5. Verify that schema has been updated and contains the plan:

```
oc get addressspaceschema standard -o yaml
```

3.5. ADDRESS PLANS

Address plans specify the expected resource usage of a given address. The sum of the resource usage for all resource types determines the amount of infrastructure provisioned for an address space. A single router and broker pod has a maximum usage of one. If a new address requires additional resources and the resource consumption is within the address space plan limits, a new pod will be created automatically to handle the increased load.

Address plans are configured by the AMQ Online service operator and are selected when creating an address.

AMQ Online includes a default set of address plans that are sufficient for most use cases.

In the Address space plans section, the address space plan references two address plans: **small-queue** and **small-anycast**. These address plans are stored as custom resources and are defined as follows:

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: small-queue
  labels:
    app: enmasse
spec:
  displayName: Small queue plan
  displayOrder: 0
  shortDescription: A plan for small queues
  longDescription: A plan for small queues that consume little resources
  addressType: queue (1)
  resources: (2)
    router: 0.2
    broker: 0.3
  partitions: 1 (3)
```

(1) The address type this plan applies to.

2 The resources consumed by addresses using this plan. The **router** field is optional for address plans referenced by a **brokered** address space plan.

3 The number of partitions that should be created for queues using this plan. Only available in the **standard** address space.

The other fields are used by the Red Hat AMQ Console UI.

A single router can support five instances of addresses and broker can support three instances of addresses with this plan. If the number of addresses with this plan increases to four, another broker is created. If it increases further to six, another router is created as well.

In the **standard** address space, address plans for the **queue** address type may contain a field **partitions**, which allows a queue to be sharded accross multiple brokers for HA and improved performance. Specifying an amount of **broker** resource above 1 will automatically cause a queue to be partitioned.

NOTE

A sharded queue no longer guarantees message ordering.

Although the example address space plan in Address space plans allows two routers and two brokers to be deployed, it only allows two pods to be deployed in total. This means that the address space is restricted to three addresses with the **small-queue** plan.

The **small-anycast** plan does not consume any broker resources, and can provision two routers at the expense of not being able to create any brokers:

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: small-anycast
  labels:
    app: enmasse
spec:
  addressType: anycast
  resources:
    router: 0.2
```

With this plan, up to 10 addresses can be created.

3.6. CREATING ADDRESS PLANS

Procedure

1. Log in as a service admin:

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

2. Select the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Create an address plan definition:

```
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: small-anycast
  labels:
    app: enmasse
spec:
  addressType: anycast
  resources:
    router: 0.2
```

4. Create the address plan:

```
oc create -f small-anycast-plan.yaml
```

5. Verify that schema has been updated and contains the plan:

```
oc get addressspaceschema standard -o yaml
```

3.7. INFRASTRUCTURE CONFIGURATION

AMQ Online creates infrastructure components such as routers, brokers, and consoles. These components can be configured while the system is running, and AMQ Online automatically updates the components with the new settings. The AMQ Online service operator can edit the AMQ Online default infrastructure configuration or create new configurations.

Infrastructure configurations can be referred to from one or more address space plans. For more information about address space plans, see Address space plans.

Infrastructure configuration can be managed for both **brokered** and **standard** infrastructure using **BrokeredInfraConfig** and **StandardInfraConfig** resources.

3.7.1. Brokered infrastructure configuration

BrokeredInfraConfig resources are used to configure infrastructure deployed by **brokered** address spaces. The brokered infrastructure configuration is referenced by address space plans in their **spec.infraConfigRef** field. For more information, see Address space plans.

For detailed information about the available brokered infrastructure configuration fields, see the Brokered infrastructure configuration fields table.

```
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: brokered-infra-config-example
spec:
  version: "0.26"
  admin:
    resources:
      memory: 256Mi
    podTemplate:
      metadata:
        labels:
          key: value
```

```
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
    podTemplate:
      spec:
        priorityClassName: messaging
```

The **version** field specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version. If omitted, the version will be assumed to be the same as the controllers reading the config.

The **admin** object specifies the settings you can configure for the **admin** components.

The **broker** object specifies the settings you can configure for the **broker** components. Changing the **.broker.resources.storage** setting does not configure the existing broker storage size.

For both **admin** and **broker** you can configure **podTemplate** settings like **metadata.labels**, **spec.priorityClassName**, **spec.tolerations** and **spec.affinity**.

For more information see Pod priority, Taints and tolerations, and Affinity and anti-affinity.

3.7.2. Standard infrastructure configuration

StandardInfraConfig resources are used to configure infrastructure deployed by **standard** address spaces. The standard infrastructure configuration is referenced by address space plans in their **spec.infraConfigRef** field. For more information, see Address space plans.

For detailed information about the available standard infrastructure configuration fields, see the Standard infrastructure configuration fields table.

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: myconfig
spec:
  version: "0.26"
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
  router:
    resources:
      memory: 256Mi
    linkCapacity: 1000
    minReplicas: 1
    policy:
      maxConnections: 1000
      maxConnectionsPerHost: 1
      maxConnectionsPerUser: 10
      maxSessionsPerConnection: 10
```

```
    maxSendersPerConnection: 5
    maxReceiversPerConnection: 5
  podTemplate:
    spec:
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 1
            preference:
            matchExpressions:
            - key: e2e-az-EastWest
              operator: In
              values:
              - e2e-az-East
              - e2e-az-West
```

The **version** field specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version. If omitted, the version will be assumed to be the same as the controllers reading the config.

The **admin** object specifies the settings you can configure for the **admin** components.

The **broker** object specifies the settings you can configure for the **broker** components. Changing the **.broker.resources.storage** setting does not configure the existing broker storage size.

The **router** object specifies the settings you can configure for the **router** components.

For **admin**, **broker** and **router** you can configure **podTemplate** settings like **metadata.labels**, **spec.priorityClassName**, **spec.tolerations** and **spec.affinity**.

See Pod priority, Taints and tolerations, and Affinity and anti-affinity for more information.

3.8. CREATING AND EDITING INFRASTRUCTURE CONFIGURATIONS

You can create a new infrastructure configuration or edit an existing one. For more information, see Infrastructure configuration.

Procedure

1. Log in as a service operator:

```
oc login -u developer
```

2. Change to the project where AMQ Online is installed:

```
oc project _amq-online-infra_
```

3. Edit the existing infrastructure configuration, or create a new infrastructure configuration using the following example:

```
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: myconfig
spec:
```

```
  version: "0.26"
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      memory: 2Gi
      storage: 100Gi
    addressFullPolicy: PAGE
  router:
    resources:
      memory: 256Mi
    linkCapacity: 1000
    minReplicas: 1
```

4. Apply the configuration changes:

```
oc apply -f standard-infra-config-example.yaml
```

5. Monitor the pods while they are restarted:

```
oc get pods -w
```

The configuration changes are applied within several minutes.

3.9. AUTHENTICATION SERVICES

Authentication services are used to configure the authentication and authorization endpoints available to messaging clients. The authentication services are configured by the AMQ Online service operator, and are specified when creating an address space.

Authentication services are configured as Custom Resources. An authentication service has a type, which can be **standard**, **external**, or **none**.

3.9.1. Standard authentication service

The **standard** authentication service type allows the tenant administrator to manage users and their related permissions through the **MessagingUser** Custom Resource. This is achieved by using a Red Hat Single Sign-On instance to store user credentials and access policies. For typical use cases only one **standard** authentication service needs to be defined.

3.9.1.1. Standard authentication service example

The following example shows an authentication service of type **standard**:

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: standard
spec:
  type: standard (1)
  standard:
    credentialsSecret: (2)
```

```
    name: my-admin-credentials
  certificateSecret (3)
    name: my-authservice-certificate
  resources: (4)
    requests:
      memory: 2Gi
    limits:
      memory: 2Gi
  storage: (5)
    type: persistent-claim
    size: 5Gi
  datasource: (6)
    type: postgresql
    host: example.com
    port: 5432
    database: authdb
```

1. Valid values for **type** are **none**, **standard**, or **external**.
2. (Optional) The secret must contain the **admin.username** field for the user and the **admin.password** field for the password of the Red Hat Single Sign-On admin user. If not specified, a random password will be generated and stored in a secret.
3. (Optional on OpenShift) A custom certificate can be specified. On OpenShift, a certificate is automatically created if not specified.
4. (Optional) Resource limits for the Red Hat Single Sign-On instance can be specified.
5. (Optional) The storage type can be specified as **ephemeral** or **persistent-claim**. For **persistent-claim**, you should also configure the size of the claim. The default type is **ephemeral**.
6. (Optional) Specifies the data source to be used by Red Hat Single Sign-On. The default option is the embedded **h2** data source. For production usage, the **postgresql** data source is recommended.

3.9.1.2. Deploying the standard authentication service

To implement the **standard** authentication service, you deploy it.

Procedure

1. Log in as a service admin:

```
oc login -u admin
```

2. Change to the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Create an **AuthenticationService** definition:

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
```

```
metadata:
  name: standard-authservice
spec:
  type: standard
```

4. Deploy the authentication service:

```
oc create -f standard-authservice.yaml
```

3.9.2. External authentication service

With the **external** authentication service you can configure an external provider of authentication and authorization policies through an AMQP SASL handshake. This configuration can be used to implement a bridge for your existing identity management system.

Depending on your use case, you might define several **external** authentication services.

3.9.2.1. External authentication service example

The following example shows an authentication service of type **external**:

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: my-external-1
spec:
  type: external
  realm: myrealm (1)
  external:
    host: example.com (2)
    port: 5671 (3)
    caCertSecret: (4)
      name: my-ca-cert
    clientCertSecret: (5)
      name: my-client-cert
```

1. (Optional) The **realm** is passed in the authentication request. If not specified, an identifier in the form of *namespace-addressspace* is used as the realm.
2. The host name of the external authentication server.
3. The port number of the external authentication server.
4. (Optional) The CA certificate to trust when connecting to the authentication server.
5. (Optional) The certificate that the client presents when connecting to the authentication server.

The external authentication server must implement the API described in the External authentication server API.

3.9.2.2. External authentication service example allowing overrides

The following example shows an authentication service of type **external** that allows overrides to the host name, port number, and realm by the messaging tenant:

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: my-external-2
spec:
  type: external
  realm: myrealm (1)
  external:
    host: example.org (2)
    port: 5671 (3)
    caCertSecret: (4)
      name: my-ca-cert
    clientCertSecret: (5)
      name: my-client-cert
    allowOverride: true (6)
```

1. (Optional) The **realm** is passed in the authentication request. If not specified, an identifier in the form of *namespace-addressspace* is used as the realm.
2. The host name of the external authentication server.
3. The port number of the external authentication server.
4. (Optional) The CA certificate to trust when connecting to the authentication server.
5. (Optional) The certificate that the client presents when connecting to the authentication server.
6. (Optional) Specifies whether address space overrides are allowed to the host name, port number, and realm. Valid values are **true** or **false**. If not specified, the default value is **false**.

The external authentication server must implement the API described in the External authentication server API.

3.9.2.3. External authentication server API

An external authentication server must implement an AMQP SASL handshake, read the connection properties of the client, and respond with the expected connection properties containing the authentication and authorization information. The authentication server is queried by the address space components, such as the router and broker, whenever a new connection is established to the messaging endpoints.

3.9.2.3.1. Authentication

The requested identity of the client can be read from the SASL handshake **username**. The implementation can then authenticate the user.

The authenticated identity is returned in the **authenticated-identity** map with the following key/values. While this example uses JSON, it must be set as an AMQP map on the connection property.

```
{
   "authenticated-identity": {
```

```
        "sub": "myid",
        "preferred_username": "myuser"
    }
}
```

3.9.2.3.2. Authorization

Authorization is a capability that can be requested by the client using the **ADDRESS-AUTHZ** connection capability. If this is set on the connection, the server responds with this capability in the offered capabilities, and add the authorization information to the connection properties.

The authorization information is stored within a map that correlates the address to a list of operations allowed on that address. The following connection property information contains the policies for the addresses **myqueue** and **mytopic**:

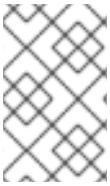
```
{
    "address-authz": {
        "myqueue": [
          "send",
          "recv"
        ],
        "mytopic": [
          "send"
        ]
    }
}
```

The allowed operations are:

- **send** - User can send to the address.
- **recv** - User can receive from the address.

3.9.3. None authentication service

The **none** authentication service type allows any client using any user name and password to send and receive messages to any address.



NOTE

It is not recommended to use the **none** authentication service in production environments. It is intended only to be used in non-production environments, such as internal test or development environments.

3.9.3.1. Deploying the none authentication service

To implement the **none** authentication service, you deploy it.

Procedure

1. Log in as a service admin:

```
oc login -u admin
```

2. Change to the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Create an **AuthenticationService** definition:

```
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: none-authservice
spec:
  type: none
```

4. Deploy the authentication service:

```
oc create -f none-authservice.yaml
```

3.10. AMQ ONLINE EXAMPLE ROLES

AMQ Online provides the following example roles that you can use directly or use as models to create your own roles.

For more information about service administrator resources, see the AMQ Online service administrator resources table.

For more information about messaging tenant resources, see the AMQ Online messaging tenant resources table.

Table 3.1. AMQ Online example roles table

Role	Description
enmasse.io:tenant-view	Specifies **get** and **list** permissions for **addresses**, **addressspaces**, **addressspaceschemas**, and **messagingusers**
enmasse.io:tenant-edit	Specifies **create**, **get**, **update**, **delete**, **list**, **watch**, and **patch** permissions for **addresses**, **addressspaces**, and **messagingusers**; **get** and **list** permissions for **addressspaceschemas**
service-admin cluster role	Specifies **create**, **get**, **update**, **delete**, **list**, **watch**, and **patch** permissions for **addressplans**, **addressspaceplans**, **brokeredinfraconfigs**, and **standardinfraconfigs**

CHAPTER 4. UPGRADING AMQ ONLINE

AMQ Online supports upgrades between minor versions using cloud-native tools. When upgrading, applying the configuration change automatically triggers the upgrade process to begin.

It is recommended to use the same method to upgrade that was used to initially install AMQ Online.

Upgrading AMQ Online is accomplished by applying the YAML files for the new version.

4.1. PRESERVATION OF MESSAGES WHEN UPGRADING FROM AMQ ONLINE 1.0.X AND 1.1.0 ONLY

The impact of ENTMQMAAS-1170 is that messages stored within brokers of AMQ Online 1.0.x and 1.1.0 are lost whenever the broker Pods are restarted.

When upgrading to the current version of AMQ Online, this procedure allows the preservation of the messages stored in the currently running instance of AMQ Online.

This procedure cannot preserve messages present on topic subscriptions. These messages will be lost. If the messages are valuable to your application, ensure that the subscribing applications have consumed all pending messages before beginning this procedure. In addition, if upgrading from AMQ Online 1.0.x, you must remove any subscription addresses.

Prerequisites

- You must shut down or quiesce all messaging applications using the AMQ Online instance so that there is no ongoing messaging traffic.
- If upgrading from AMQ Online 1.0.x, you must remove any subscription addresses before beginning this procedure.

Procedure

1. Log in as a user with **cluster-admin** privileges:

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

2. Switch to the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. If upgrading from AMQ Online 1.0.x, remove any topic subscription addresses.
   a. Identify the subscription addresses:

```
oc get address --selector=addressType=subscription --all-namespaces
```

   b. Delete the subscription address from each project where they exist:

```
oc delete address --selector=addressType=subscription --namespace <namespace>
```

4. Run the migration script located in the AMQ Online installation bundle:

```
./migrate_artemis_journal.sh
```

The script prompts you to confirm you want to proceed, and then copies the journal to a persistent location. The output from the script is similar to this example:

```
Remediation for https://issues.jboss.org/browse/ENTMQMAAS-1170
This script will copy the current Artemis journal for broker pods running in this AMQ Online 1.0/1.1.0 infra namespace to the correct persistent location.  It should be run before upgrade to AMQ Online 1.1.1.
It is important that applications using the AMQ Online are shutdown (or quiesced) before you proceed so that there is no messaging traffic through the system.
Are you sure? Y
=========
Processing broker pod broker-qxv5nefate-kmye-0
Detected Artemis version 2.6.3
Successfully processed broker pod broker-qxv5nefate-kmye-0
=========
Processing broker pod broker.7q0b04roxv-d4ddfb45f-zbjlq
Detected Artemis version 2.6.3
Successfully processed broker pod broker.7q0b04roxv-d4ddfb45f-zbjlq
All broker pods processed
```

5. Perform the appropriate upgrading procedure, given your original method of installation:
   - Upgrading AMQ Online using a YAML bundle .
   - Upgrading AMQ Online using Ansible

4.2. UPGRADING AMQ ONLINE USING A YAML BUNDLE

Prerequisites

- A new release of AMQ Online. For more information, see Downloading AMQ Online.

Procedure

1. Log in as a service operator:

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

2. Select the project where AMQ Online is installed:

```
oc project amq-online-infra
```

3. Apply the new release bundle:

```
oc apply -f install/bundles/amq-online
```

4. Monitor pods while they are restarted:

```
oc get pods -w
```

The pods restart and become active within several minutes.

4.3. UPGRADING AMQ ONLINE USING ANSIBLE

Prerequisites

- A new release of AMQ Online. For more information, see Downloading AMQ Online.

Procedure

1. Log in as a service operator:

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

2. Run the Ansible playbook from the new release:

```
ansible-playbook -i inventory-file ansible/playbooks/openshift/deploy_all.yml
```

3. Monitor pods while they are restarted:

```
oc get pods -w
```

The pods restart and become active within several minutes.

4.4. RESTARTING ROUTER PODS RUNNING A PREVIOUS VERSION OF THE IMAGE

When upgrading, sometimes a router pod is not automatically restarted by OpenShift. If this issue occurs, it is important to manually restart the affected pod to ensure correct operation of the system.

Prerequisites

- Completed the upgrading steps to a new release of AMQ Online. For more information on how to upgrade, see:
  - Upgrading AMQ Online using a YAML bundle
  - Upgrading AMQ Online using Ansible

Procedure

1. Find the router pod running a previous version of the image:

```
oc get pod -o go-template='{{range .items}}{{.metadata.name}}{{"\n"}}{{range
.spec.containers}}{{"\t"}}{{.image}}{{"\n"}}{{end}}{{"\n"}}{{end}}' --selector=capability=router
```

2. Delete the router pod that is running a previous version of the image:

```
oc delete pod <name>
```

The pod is automatically restarted.

CHAPTER 5. MONITORING AMQ ONLINE

You can monitor AMQ Online by deploying built-in monitoring tools or using your pre-existing monitoring infrastructure by deploying the required service monitors and Prometheus rules.

5.1. (OPTIONAL) DEPLOYING THE APPLICATION MONITORING OPERATOR

To monitor AMQ Online, an operator that acts on the monitoring Custom Resource Definitions must be deployed. You may skip this step if you have such an operator installed on your OpenShift cluster.

Procedure

1. Log in as a user with **cluster-admin** privileges:

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

2. (Optional) If you want to deploy to a namespace other than **amq-online-monitoring** you must run the following command and substitute **amq-online-monitoring** in subsequent steps:

```
sed -i 's/amq-online-monitoring/my-namespace/' install/bundles/amq-online/*.yaml
```

3. Create the amq-online-monitoring namespace:

```
oc new-project amq-online-monitoring
```

4. Deploy the **monitoring-operator** component:

```
oc apply -f install/components/monitoring-operator
```

5.2. (OPTIONAL) DEPLOYING THE KUBE-STATE-METRICS AGENT

You can monitor AMQ Online pods using the **kube-state-metrics** agent.

Procedure

1. Log in as a user with **cluster-admin** privileges:

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

2. Select the **amq-online-infra** project:

```
oc project amq-online-infra
```

3. Deploy the **kube-state-metrics** component:

```
oc apply -f install/components/kube-state-metrics
```

5.3. DEPLOYING MONITORING USING A YAML BUNDLE

The simplest way to deploy monitoring is to use a predefined YAML bundle.

Prerequisites

- The Application Monitoring Operator or an operator managing the same resources must be installed.

Procedure

1. Label the *amq-online-infra* namespace:

```
oc label namespace amq-online-infra monitoring-key=middleware
```

2. Select the **amq-online-infra** project:

```
oc project amq-online-infra
```

3. Deploy the **monitoring** bundle:

```
oc apply -f install/bundles/monitoring
```

5.4. DEPLOYING MONITORING USING ANSIBLE

Monitoring can also be deployed during the AMQ Online installation using Ansible using the required configuration settings.

5.5. CONFIGURING ALERT NOTIFICATIONS

To configure alert notifications, such as emails, you must change the default configuration of Alertmanager.

Prerequisites

- Create an Alertmanager configuration file following the Alertmanager documentation. An example configuration file for email notifications is shown:

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: enmasse
  name: alertmanager-config
data:
  alertmanager.yml: |
    global:
      resolve_timeout: 5m
      smtp_smarthost: localhost
      smtp_from: alerts@localhost
      smtp_auth_username: admin
      smtp_auth_password: password
    route:
      group_by: ['alertname']
      group_wait: 60s
```

```
    group_interval: 60s
    repeat_interval: 1h
    receiver: 'sysadmins'
  receivers:
  - name: 'sysadmins'
    email_configs:
    - to: sysadmin@localhost
  inhibit_rules:
    - source_match:
        severity: 'critical'
      target_match:
        severity: 'warning'
      equal: ['alertname']
```

- Your Alertmanager configuration file must be named **alertmanager.yaml** so it can be read by the Prometheus Operator.

Procedure

1. Delete the secret containing the default configuration:

```
oc delete secret alertmanager-application-monitoring
```

2. Create a secret containing your new configuration:

```
oc create secret generic alertmanager-application-monitoring --from-file=alertmanager.yaml
```

5.6. USING QDSTAT

You can use **qdstat** to monitor the AMQ Online service.

5.6.1. Viewing router connections using qdstat

You can view the router connections using **qdstat**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. On the command line, run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 -c

Connections
  id   host              container                       role    dir  security
authentication              tenant

========================================================================
========================================================================
===
```

```
  3   172.17.0.9:34998     admin-78794c68c8-9jdd6              normal  in   TLSv1.2(ECDHE-
RSA-AES128-GCM-SHA256)  CN=admin,O=io.enmasse(x.509)
  12  172.30.188.174:5671  27803a14-42d2-6148-9491-a6c1e69e875a  normal  out
TLSv1.2(ECDHE-RSA-AES128-GCM-SHA256)  x.509
  567  127.0.0.1:43546      b240c652-82df-48dd-b54e-3b8bbaef16c6  normal  in   no-security
PLAIN
```

5.6.2. Viewing router addresses using qdstat

You can view the router addresses using **qdstat**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. Run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 -a

Router Addresses
  class     addr                 phs  distrib      in-proc  local  remote  cntnr  in     out    thru  to-proc
from-proc

=================================================================================
=============================================
  local     $_management_internal       closest       1       0      0       0      0      0      0     588
588
  link-in   $lwt                    linkBalanced  0       0      0       0      0      0      0     0       0
  link-out  $lwt                    linkBalanced  0       0      0       0      0      0      0     0       0
  mobile    $management             0    closest       1       0      0       0      601    0      0     601     0
  local     $management                 closest       1       0      0       0      2,925  0      0     2,925   0
  local     qdhello                 flood         1       0      0       0      0      0      0     0       5,856
  local     qdrouter                flood         1       0      0       0      0      0      0     0       0
  topo      qdrouter                flood         1       0      0       0      0      0      0     0       196
  local     qdrouter.ma                multicast     1       0      0       0      0      0      0     0       0
  topo      qdrouter.ma                multicast     1       0      0       0      0      0      0     0       0
  local     temp.VTXOKyyWsq7OEei        balanced      0       1      0       0      0      0      0     0
0
  local     temp.k2RGQNPe6sDMvz4        balanced      0       1      0       0      0      3,511  0     0
3,511
  local     temp.xg+y8l_Tr4Y94LA        balanced      0       1      0       0      0      5      0     0
5
```

5.6.3. Viewing router links using qdstat

You can view the router links using **qdstat**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. On the command line, run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 -l

Router Links
  type      dir  conn id  id  peer  class   addr                 phs  cap  undel  unsett  del   presett
psdrop  acc   rej  rel  mod  admin    oper

=============================================================================
=============================================================================

  endpoint  in   3       8                                    250  0      0       3829  0       0       3829  0
0    0    enabled  up
  endpoint  out  3       9         local   temp.k2RGQNPe6sDMvz4       250  0      0       3829  3829
0       0     0    0    0    enabled  up
  endpoint  in   12      10                                   250  0      0       5     0       0       5     0    0
0    enabled  up
  endpoint  out  12      11        local   temp.xg+y8l_Tr4Y94LA       250  0      0       5     5       0
0     0    0    0    enabled  up
  endpoint  in   645     26        mobile  $management          0    50   0      0       1     0       0
1     0    0    0    enabled  up
  endpoint  out  645     27         local   temp.0BrHJ1O+fi6whyg       50   0      0       0     0       0
0     0    0    0    enabled  up
```

5.6.4. Viewing link routes using qdstat

You can view the link routes using **qdstat**.

Procedure

1. On the command line, run the following command to obtain the **podname** value needed in the following step:

```
oc get pods
```

2. On the command line, run the following command:

```
oc exec -n namespace -it qdrouterd-podname -- qdstat -b 127.0.0.1:7777 --linkroutes

Link Routes
  address  dir  distrib       status
  ====================================
  $lwt     in   linkBalanced  inactive
  $lwt     out  linkBalanced  inactive
```

CHAPTER 6. UNINSTALLING AMQ ONLINE

You must uninstall AMQ Online using the same method that you used to install AMQ Online.

6.1. UNINSTALLING AMQ ONLINE USING THE YAML BUNDLE

This method uninstalls AMQ Online that was installed using the YAML bundle.

Procedure

1. Log in as a user with **cluster-admin** privileges:

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

2. Delete the cluster-level resources:

```
oc delete clusterrolebindings -l app=enmasse
oc delete crd -l app=enmasse
oc delete clusterroles -l app=enmasse
oc delete apiservices -l app=enmasse
oc delete oauthclients -l app=enmasse
```

3. (Optional) Delete the service catalog integration:

```
oc delete clusterservicebrokers -l app=enmasse
```

4. Delete the project where AMQ Online is deployed:

```
oc delete project amq-online-infra
```

6.2. UNINSTALLING AMQ ONLINE USING ANSIBLE

Uninstalling AMQ Online using Ansible requires using the same inventory file that was used for installing AMQ Online.

NOTE

The playbook deletes the **amq-online-infra** project.

Procedure

1. Run the Ansible playbook, where ***inventory-file*** specifies the inventory file used at installation:

```
ansible-playbook -i inventory-file ansible/playbooks/openshift/uninstall.yml
```

APPENDIX A. AMQ ONLINE RESOURCES FOR SERVICE ADMINISTRATORS

The following table describes the AMQ Online resources that pertain to the service administrator role.

Table A.1. AMQ Online service administrator resources table

Resource	Description
addressplans	Specifies the address plan.
addressspaceplans	Specifies the address space plan.
addressspaceschemas	Defines the service characteristics available to an **addresspace**. An **addressspace** refers to one **addressspaceschema**. **standard** and **brokered** are predefined **addressspaceschemas**.
brokeredinfraconfigs	Specifies the infrastructure configuration for brokered address spaces. For more information see Brokered infrastructure configuration fields table.
standardinfraconfigs	Specifies the infrastructure configuration for standard address spaces. For more information see Standard infrastructure configuration fields table.

APPENDIX B. BROKERED INFRASTRUCTURE CONFIGURATION FIELDS

This table shows the fields available for the brokered infrastructure configuration and a brief description.

Table B.1. Brokered infrastructure configuration fields table

Field	Description
version	Specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version.
admin.resources.memory	Specifies the amount of memory allocated to the admin pod.
admin.podTemplate.metadata.labels	Specifies the labels added to the admin pod.
admin.podTemplate.spec.priorityClassName	Specifies the priority class to use for the admin pod so you can prioritize admin pods over other pods in the OpenShift cluster.
admin.podTemplate.spec.affinity	Specifies the affinity settings for the admin pod so you can specify where on particular nodes a pod runs, or if it cannot run together with other instances.
admin.podTemplate.spec.tolerations	Specifies the toleration settings for the admin pod, which allows this pod to run on certain nodes that other pods cannot run on.
broker.addressFullPolicy	Specifies action taken when a queue is full: **BLOCK**, **FAIL**, **PAGE**, **DROP**. The default value is **PAGE**. For more information see the AMQ Broker documentation.
broker.globalMaxSize	Specifies the maximum amount of memory used for queues in the broker.
broker.resources.memory	Specifies the amount of memory allocated to the broker.
broker.resources.storage	Specifies the amount of storage requested for the broker.
broker.podTemplate.metadata.labels	Specifies the labels added to the broker pod.
broker.podTemplate.spec.priorityClassName	Specifies the priority class to use for the broker pod so you can prioritize broker pods over other pods in the OpenShift cluster.
broker.podTemplate.spec.affinity	Specifies the affinity settings for the broker pod so you can specify where on particular nodes a pod runs, or if it cannot run together with other instances.
broker.podTemplate.spec.tolerations	Specifies the toleration settings for the broker pod, which allows this pod to run on certain nodes that other pods cannot run on.

broker.storageClassName	Specifies what storage class to use for the persistent volume for the broker.
broker.updatePersistentVolumeClaim	If the persistent volume supports resizing, setting this value to **true** allows the broker storage to be resized.

APPENDIX C. STANDARD INFRASTRUCTURE CONFIGURATION FIELDS

This table shows the fields available for the standard infrastructure configuration and a brief description.

Table C.1. Standard infrastructure configuration fields table

Field	Description
version	Specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version.
admin.resources.memory	Specifies the amount of memory allocated to the admin pod.
admin.podTemplate.metadata.labels	Specifies the labels added to the admin pod.
admin.podTemplate.spec.priorityClassName	Specifies the priority class to use for the admin pod so you can prioritize admin pods over other pods in the OpenShift cluster.
admin.podTemplate.spec.affinity	Specifies the affinity settings for the admin pod so you can specify where on particular nodes a pod runs, or if it cannot run together with other instances.
admin.podTemplate.spec.tolerations	Specifies the toleration settings for the admin pod, which allow this pod to run on certain nodes on which other pods cannot run.
broker.addressFullPolicy	Specifies action taken when a queue is full: **BLOCK**, **FAIL**, **PAGE**, **DROP**. The default value is **PAGE**. For more information see the AMQ Broker documentation.
broker.globalMaxSize	Specifies the maximum amount of memory used for queues in the broker.
broker.resources.memory	Specifies the amount of memory allocated to the broker.
broker.resources.storage	Specifies the amount of storage requested for the broker.
broker.podTemplate.metadata.labels	Specifies the labels added to the broker pod.
broker.podTemplate.spec.priorityClassName	Specifies the priority class to use for the broker pod so you can prioritize broker pods over other pods in the OpenShift cluster.
broker.podTemplate.spec.affinity	Specifies the affinity settings for the broker pod so you can specify where on particular nodes a pod runs, or if it cannot run together with other instances.
broker.podTemplate.spec.tolerations	Specifies the toleration settings for the broker pod, which allow this pod to run on certain nodes on which other pods cannot run.

broker.connectorIdleTimeout	Specifies the AMQP idle timeout to use for connection to router.
broker.connectorWorkerThreads	Specifies the number of worker threads of the connection to the router.
broker.storageClassName	Specifies what storage class to use for the persistent volume for the broker.
broker.updatePersistentVolumeClaim	If the persistent volume supports resizing, setting this value to **true** allows the broker storage to be resized.
router.resources.memory	Specifies the amount of memory allocated to the router.
router.linkCapacity	Specifies the default number of credits issued on AMQP links for the router.
router.handshakeTimeout	Specifies the amount of time in seconds to wait for the secure handshake to be initiated.
router.minReplicas	Specifies the minimum number of router pods to run; a minimum of two are required for high availability (HA) configuration.
router.podTemplate.metadata.labels	Specifies the labels added to the router pod.
router.podTemplate.spec.priorityClassName	Specifies the priority class to use for the router pod so you can prioritize router pods over other pods in the OpenShift cluster.
router.podTemplate.spec.affinity	Specifies the affinity settings for the router pod so you can specify where on particular nodes a pod runs, or if it cannot run together with other instances.
router.podTemplate.spec.tolerations	Specifies the toleration settings for the router pod, which allow this pod to run on certain nodes on which other pods cannot run.
router.idleTimeout	Specifies the AMQP idle timeout to use for all router listeners.
router.workerThreads	Specifies the number of worker threads to use for the router.
router.policy.maxConnections	Specifies the maximum number of router connections allowed.
router.policy.maxConnectionsPerUser	Specifies the maximum number of router connections allowed per user.
router.policy.maxConnectionsPerHost	Specifies the maximum number of router connections allowed per host.
router.policy.maxSessionsPerConnection	Specifies the maximum number of sessions allowed per router connection.

router.policy.maxSendersPerConnection	Specifies the maximum number of senders allowed per router connection.
router.policy.maxReceiversPerConnection	Specifies the maximum number of receivers allowed per router connection.

APPENDIX D. REST API REFERENCE

D.1. ENMASSE REST API

D.1.1. Overview

This is the EnMasse API specification.

D.1.1.1. Version information

Version : 1.0.0

D.1.1.2. URI scheme

Schemes : HTTPS

D.1.1.3. Tags

- addresses : Operating on Addresses.
- addressplans : Operating on AddressPlans.
- addressspaceplans : Operating on AddressSpacePlans.
- addressspaces : Operate on AddressSpaces
- brokeredinfraconfigs : Operating on BrokeredInfraConfigs.
- messagingusers : Operating on MessagingUsers.
- standardinfraconfigs : Operating on StandardInfraConfigs.

D.1.1.4. External Docs

Description : Find out more about EnMasse
URL : http://enmasse.io

D.1.2. Paths

D.1.2.1. POST /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans

D.1.2.1.1. Description

create an AddressSpacePlan

D.1.2.1.2. Parameters

Type	Name	Description	Schema

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.admin.v1beta2.AddressSpacePlan

D.1.2.1.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlan
201	Created	io.enmasse.admin.v1beta2.AddressSpacePlan
401	Unauthorized	No Content

D.1.2.1.4. Consumes

- **application/json**

D.1.2.1.5. Produces

- **application/json**

D.1.2.1.6. Tags

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.2. GET /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans

D.1.2.2.1. Description

list objects of kind AddressSpacePlan

D.1.2.2.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

D.1.2.2.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlanList
401	Unauthorized	No Content

D.1.2.2.4. Produces

- **application/json**

D.1.2.2.5. Tags

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.3. GET /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}

D.1.2.3.1. Description

read the specified AddressSpacePlan

D.1.2.3.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpacePlan to read.	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.3.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlan
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.3.4. Consumes

- **application/json**

D.1.2.3.5. Produces

- **application/json**

D.1.2.3.6. Tags

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.4. PUT /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}

D.1.2.4.1. Description

replace the specified AddressSpacePlan

D.1.2.4.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpacePlan to replace.	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.admin.v1beta2.AddressSpacePlan

D.1.2.4.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.admin.v1beta2.AddressSpacePlan
201	Created	io.enmasse.admin.v1beta2.AddressSpacePlan
401	Unauthorized	No Content

D.1.2.4.4. Produces

- **application/json**

D.1.2.4.5. Tags

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.5. DELETE /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}

D.1.2.5.1. Description

delete an AddressSpacePlan

D.1.2.5.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpacePlan to delete.	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.5.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.5.4. Produces

- **application/json**

D.1.2.5.5. Tags

- addressspaceplan
- admin
- enmasse_v1beta2

D.1.2.6. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses

D.1.2.6.1. Description

create an Address

D.1.2.6.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.v1beta1.Address

D.1.2.6.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address

HTTP Code	Description	Schema
401	Unauthorized	No Content

D.1.2.6.4. Consumes

- **application/json**

D.1.2.6.5. Produces

- **application/json**

D.1.2.6.6. Tags

- addresses
- enmasse_v1beta1

D.1.2.7. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses

D.1.2.7.1. Description

list objects of kind Address

D.1.2.7.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

D.1.2.7.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressList
401	Unauthorized	No Content

D.1.2.7.4. Produces

- **application/json**

D.1.2.7.5. Tags

- addresses
- enmasse_v1beta1

D.1.2.8. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.8.1. Description

read the specified Address

D.1.2.8.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of Address to read	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.8.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.8.4. Consumes

- **application/json**

D.1.2.8.5. Produces

- **application/json**

D.1.2.8.6. Tags

- addresses
- enmasse_v1beta1

D.1.2.9. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.9.1. Description

replace the specified Address

D.1.2.9.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of Address to replace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.v1beta1.Address

D.1.2.9.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address
401	Unauthorized	No Content

D.1.2.9.4. Produces

- **application/json**

D.1.2.9.5. Tags

- addresses
- enmasse_v1beta1

D.1.2.10. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.10.1. Description

delete an Address

D.1.2.10.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of Address to delete	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.10.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.10.4. Produces

- **application/json**

D.1.2.10.5. Tags

- addresses
- enmasse_v1beta1

D.1.2.11. PATCH /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}

D.1.2.11.1. Description

patches (RFC6902) the specified Address

D.1.2.11.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of Address to replace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		JsonPatchRequest

D.1.2.11.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
401	Unauthorized	No Content

D.1.2.11.4. Consumes

- **application/json-patch+json**

D.1.2.11.5. Produces

- **application/json**

D.1.2.11.6. Tags

- addresses
- enmasse_v1beta1

D.1.2.12. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces

D.1.2.12.1. Description

create an AddressSpace

D.1.2.12.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.v1beta1.AddressSpace

D.1.2.12.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpace

HTTP Code	Description	Schema
201	Created	io.enmasse.v1beta1.AddressSpace
401	Unauthorized	No Content

D.1.2.12.4. Consumes

- **application/json**

D.1.2.12.5. Produces

- **application/json**

D.1.2.12.6. Tags

- addressspaces
- enmasse_v1beta1

D.1.2.13. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces

D.1.2.13.1. Description

list objects of kind AddressSpace

D.1.2.13.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

D.1.2.13.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpaceList
401	Unauthorized	No Content

D.1.2.13.4. Produces

- **application/json**

D.1.2.13.5. Tags

- addressspaces
- enmasse_v1beta1

D.1.2.14. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addre

D.1.2.14.1. Description

create Addresses in an AddressSpace

D.1.2.14.2. Parameters

Type	Name	Description	Schema
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*	AddressList object	io.enmasse.v1beta1.AddressList

D.1.2.14.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.14.4. Consumes

- **application/json**

D.1.2.14.5. Produces

- **application/json**

D.1.2.14.6. Tags

- addressspace_addresses
- enmasse_v1beta1

D.1.2.15. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addre

D.1.2.15.1. Description

list objects of kind Address in AddressSpace

D.1.2.15.2. Parameters

Type	Name	Description	Schema
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.15.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressList
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.15.4. Produces

- **application/json**

D.1.2.15.5. Tags

- addressspace_addresses
- enmasse_v1beta1

D.1.2.16. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addre

D.1.2.16.1. Description

read the specified Address in AddressSpace

D.1.2.16.2. Parameters

Type	Name	Description	Schema
Path	**address** *required*	Name of Address	string
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.16.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.16.4. Produces

- **application/json**

D.1.2.16.5. Tags

- addressspace_addresses
- enmasse_v1beta1

D.1.2.17. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addre

D.1.2.17.1. Description

replace Address in an AddressSpace

D.1.2.17.2. Parameters

Type	Name	Description	Schema
Path	**address** *required*	Name of address	string
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*	Address object	io.enmasse.v1beta1.Address

D.1.2.17.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.Address
201	Created	io.enmasse.v1beta1.Address
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.17.4. Consumes

- **application/json**

D.1.2.17.5. Produces

- **application/json**

D.1.2.17.6. Tags

- addressspace_addresses
- enmasse_v1beta1

D.1.2.18. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{addressSpace}/addre

D.1.2.18.1. Description

delete an Address in AddressSpace

D.1.2.18.2. Parameters

Type	Name	Description	Schema
Path	**address** *required*	Name of Address	string
Path	**addressSpace** *required*	Name of AddressSpace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.18.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.18.4. Produces

- **application/json**

D.1.2.18.5. Tags

- addressspace_addresses
- enmasse_v1beta1

D.1.2.19. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.19.1. Description

read the specified AddressSpace

D.1.2.19.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpace to read	string

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.19.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpace
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.19.4. Consumes

- **application/json**

D.1.2.19.5. Produces

- **application/json**

D.1.2.19.6. Tags

- addressspaces
- enmasse_v1beta1

D.1.2.20. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.20.1. Description

replace the specified AddressSpace

D.1.2.20.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpace to replace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

Type	Name	Description	Schema
Body	**body** *required*		io.enmasse.v1beta1.AddressSpace

D.1.2.20.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpace
201	Created	io.enmasse.v1beta1.AddressSpace
401	Unauthorized	No Content

D.1.2.20.4. Produces

- **application/json**

D.1.2.20.5. Tags

- addressspaces
- enmasse_v1beta1

D.1.2.21. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.21.1. Description

delete an AddressSpace

D.1.2.21.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpace to delete	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.21.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.21.4. Produces

- **application/json**

D.1.2.21.5. Tags

- addressspaces
- enmasse_v1beta1

D.1.2.22. PATCH /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.22.1. Description

patches (RFC6902) the specified AddressSpace

D.1.2.22.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of AddressSpace to replace	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		JsonPatchRequest

D.1.2.22.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.v1beta1.AddressSpace
401	Unauthorized	No Content

D.1.2.22.4. Consumes

- **application/json-patch+json**

D.1.2.22.5. Produces

- **application/json**

D.1.2.22.6. Tags

- addressspaces
- enmasse_v1beta1

D.1.2.23. POST /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

D.1.2.23.1. Description

create a MessagingUser

D.1.2.23.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.user.v1beta1.MessagingUser

D.1.2.23.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUser
201	Created	io.enmasse.user.v1beta1.MessagingUser
401	Unauthorized	No Content

D.1.2.23.4. Consumes

- **application/json**

D.1.2.23.5. Produces

- **application/json**

D.1.2.23.6. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.24. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

D.1.2.24.1. Description

list objects of kind MessagingUser

D.1.2.24.2. Parameters

Type	Name	Description	Schema
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Query	**labelSelector** *optional*	A selector to restrict the list of returned objects by their labels. Defaults to everything.	string

D.1.2.24.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUserList
401	Unauthorized	No Content

D.1.2.24.4. Produces

- **application/json**

D.1.2.24.5. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.25. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.25.1. Description

read the specified MessagingUser

D.1.2.25.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of MessagingUser to read. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.25.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUser
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.25.4. Consumes

- **application/json**

D.1.2.25.5. Produces

- **application/json**

D.1.2.25.6. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.26. PUT /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.26.1. Description

replace the specified MessagingUser

D.1.2.26.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of MessagingUser to replace. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		io.enmasse.user.v1beta1.MessagingUser

D.1.2.26.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUser
201	Created	io.enmasse.user.v1beta1.MessagingUser
401	Unauthorized	No Content

D.1.2.26.4. Produces

- **application/json**

D.1.2.26.5. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.27. DELETE /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.27.1. Description

delete a MessagingUser

D.1.2.27.2. Parameters

Type	Name	Description	Schema
Path	**name** *required*	Name of MessagingUser to delete. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string

D.1.2.27.3. Responses

HTTP Code	Description	Schema
200	OK	Status
401	Unauthorized	No Content
404	Not found	No Content

D.1.2.27.4. Produces

- **application/json**

D.1.2.27.5. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.28. PATCH /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.28.1. Description

patches (RFC6902) the specified MessagingUser

D.1.2.28.2. Parameters

Type	Name	Description	Schema

Type	Name	Description	Schema
Path	**name** *required*	Name of MessagingUser to replace. Must include addressSpace and dot separator in the name (that is, 'myspace.user1'	string
Path	**namespace** *required*	object name and auth scope, such as for teams and projects	string
Body	**body** *required*		JsonPatchRequest

D.1.2.28.3. Responses

HTTP Code	Description	Schema
200	OK	io.enmasse.user.v1beta1.MessagingUser
401	Unauthorized	No Content

D.1.2.28.4. Consumes

- **application/json-patch+json**

D.1.2.28.5. Produces

- **application/json**

D.1.2.28.6. Tags

- auth
- enmasse_v1beta1
- user

D.1.3. Definitions

D.1.3.1. JsonPatchRequest

Name	Schema
document *required*	object

Name	Schema
patch *required*	< Patch > array

D.1.3.2. ObjectMeta

ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.

Name	Schema
name *required*	string
namespace *optional*	string

D.1.3.3. Patch

Name	Description	Schema
from *optional*	Required for operations copy, replace	string
op *required*		enum (add, remove, replace, move, copy, test)
path *required*	Slash separated format	string
value *optional*	Required for operations add, replace, test	string

D.1.3.4. Status

Status is a return value for calls that do not return other objects.

Name	Description	Schema
code *optional*	Suggested HTTP return code for this status, 0 if not set.	integer (int32)

D.1.3.5. io.enmasse.admin.v1beta1.BrokeredInfraConfig

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
kind *required*	enum (BrokeredInfraConfig)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec

D.1.3.6. io.enmasse.admin.v1beta1.BrokeredInfraConfigList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
items *required*	< io.enmasse.admin.v1beta1.BrokeredInfraConfig > array
kind *required*	enum (BrokeredInfraConfigList)

D.1.3.7. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec

Name	Schema
admin *optional*	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin
broker *optional*	io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker
networkPolicy *optional*	networkPolicy
version *optional*	string

networkPolicy

Name	Schema
egress *optional*	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress *optional*	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

D.1.3.8. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin

Name	Schema
podTemplate *optional*	io.enmasse.admin.v1beta1.InfraConfigPodSpec
resources *optional*	resources

resources

Name	Schema
memory *optional*	string

D.1.3.9. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker

Name	Schema
addressFullPolicy *optional*	enum (PAGE, BLOCK, FAIL)
podTemplate *optional*	io.enmasse.admin.v1beta1.InfraConfigPodSpec
resources *optional*	resources
storageClassName *optional*	string
updatePersistentVolumeClaim *optional*	boolean

resources

Name	Schema
memory *optional*	string
storage *optional*	string

D.1.3.10. io.enmasse.admin.v1beta1.InfraConfigPodSpec

Name	Schema
metadata *optional*	metadata
spec *optional*	spec

metadata

Name	Schema
labels *optional*	object

spec

Name	Schema
affinity *optional*	object
containers *optional*	< containers > array
priorityClassName *optional*	string
tolerations *optional*	< object > array

containers

Name	Schema
resources *optional*	object

D.1.3.11. io.enmasse.admin.v1beta1.StandardInfraConfig

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
kind *required*	enum (StandardInfraConfig)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.admin.v1beta1.StandardInfraConfigSpec

D.1.3.12. io.enmasse.admin.v1beta1.StandardInfraConfigList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta1)
items *required*	< io.enmasse.admin.v1beta1.StandardInfraConfig > array
kind *required*	enum (StandardInfraConfigList)

D.1.3.13. io.enmasse.admin.v1beta1.StandardInfraConfigSpec

Name	Schema
admin *optional*	io.enmasse.admin.v1beta1.StandardInfraConfigSpecAdmin
broker *optional*	io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker

Name	Schema
networkPolicy *optional*	networkPolicy
router *optional*	io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter
version *optional*	string

networkPolicy

Name	Schema
egress *optional*	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress *optional*	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

D.1.3.14. io.enmasse.admin.v1beta1.StandardInfraConfigSpecAdmin

Name	Schema
podTemplate *optional*	io.enmasse.admin.v1beta1.InfraConfigPodSpec
resources *optional*	resources

resources

Name	Schema
memory *optional*	string

D.1.3.15. io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker

Name	Schema
addressFullPolicy *optional*	enum (PAGE, BLOCK, FAIL)
connectorIdleTimeout *optional*	integer
connectorWorkerThreads *optional*	integer
podTemplate *optional*	io.enmasse.admin.v1beta1.InfraConfigPodSpec
resources *optional*	resources
storageClassName *optional*	string
updatePersistentVolumeClaim *optional*	boolean

resources

Name	Schema
memory *optional*	string
storage *optional*	string

D.1.3.16. io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter

Name	Schema
idleTimeout *optional*	integer
initialHandshakeTimeout *optional*	integer
linkCapacity *optional*	integer

Name	Schema
minReplicas *optional*	integer
podTemplate *optional*	io.enmasse.admin.v1beta1.InfraConfigPodSpec
policy *optional*	policy
resources *optional*	resources
workerThreads *optional*	integer

policy

Name	Schema
maxConnections *optional*	integer
maxConnectionsPerHost *optional*	integer
maxConnectionsPerUser *optional*	integer
maxReceiversPerConnection *optional*	integer
maxSendersPerConnection *optional*	integer
maxSessionsPerConnection *optional*	integer

resources

Name	Schema
memory *optional*	string

D.1.3.17. io.enmasse.admin.v1beta2.AddressPlan

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta2)
kind *required*	enum (AddressPlan)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.admin.v1beta2.AddressPlanSpec

D.1.3.18. io.enmasse.admin.v1beta2.AddressPlanList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta2)
items *required*	< io.enmasse.admin.v1beta2.AddressPlan > array
kind *required*	enum (AddressPlanList)

D.1.3.19. io.enmasse.admin.v1beta2.AddressPlanSpec

Name	Schema
addressType *required*	string
displayName *required*	string
displayOrder *optional*	integer
longDescription *optional*	string
partitions *optional*	integer

Name	Schema
resources *required*	resources
shortDescription *optional*	string

resources

Name	Schema
broker *optional*	number
router *optional*	number

D.1.3.20. io.enmasse.admin.v1beta2.AddressSpacePlan

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta2)
kind *required*	enum (AddressSpacePlan)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.admin.v1beta2.AddressSpacePlanSpec

D.1.3.21. io.enmasse.admin.v1beta2.AddressSpacePlanList

Name	Schema
apiVersion *required*	enum (admin.enmasse.io/v1beta2)
items *required*	< io.enmasse.admin.v1beta2.AddressSpacePlan > array
kind *required*	enum (AddressSpacePlanList)

D.1.3.22. io.enmasse.admin.v1beta2.AddressSpacePlanSpec

Name	Schema
addressPlans *required*	< string > array
addressSpaceType *required*	string
displayName *required*	string
displayOrder *optional*	integer
infraConfigRef *required*	string
longDescription *optional*	string
resourceLimits *required*	resourceLimits
shortDescription *optional*	string

resourceLimits

Name	Schema
aggregate *optional*	number
broker *optional*	number
router *optional*	number

D.1.3.23. io.enmasse.user.v1beta1.MessagingUser

Name	Schema
apiVersion *required*	enum (user.enmasse.io/v1beta1)

Name	Schema
kind *required*	enum (MessagingUser)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.user.v1beta1.UserSpec

D.1.3.24. io.enmasse.user.v1beta1.MessagingUserList

Name	Schema
apiVersion *required*	enum (user.enmasse.io/v1beta1)
items *required*	< io.enmasse.user.v1beta1.MessagingUser > array
kind *required*	enum (MessagingUserList)

D.1.3.25. io.enmasse.user.v1beta1.UserSpec

Name	Schema
authentication *optional*	authentication
authorization *optional*	< authorization > array
username *required*	string

authentication

Name	Description	Schema
federatedUserid *optional*	User id of the user to federate when 'federated' type is specified.	string

Name	Description	Schema
federatedUsername *optional*	User name of the user to federate when 'federated' type is specified.	string
password *optional*	Base64 encoded value of password when 'password' type is specified.	string
provider *optional*	Name of provider to use for federated identity when 'federated' type is specified.	string
type *required*		enum (password, serviceaccount)

authorization

Name	Schema
addresses *optional*	< string > array
operations *required*	< enum (send, receive, view, manage) > array

D.1.3.26. io.enmasse.v1beta1.Address

Name	Schema
apiVersion *required*	enum (enmasse.io/v1beta1)
kind *required*	enum (Address)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.v1beta1.AddressSpec
status *optional*	io.enmasse.v1beta1.AddressStatus

D.1.3.27. io.enmasse.v1beta1.AddressList

Name	Description	Schema
apiVersion *required*	Default : **"enmasse.io/v1beta1"**	enum (enmasse.io/v1beta1)
items *required*		< io.enmasse.v1beta1.Address > array
kind *required*		enum (AddressList)

D.1.3.28. io.enmasse.v1beta1.AddressSpace

Name	Schema
apiVersion *required*	enum (enmasse.io/v1beta1)
kind *required*	enum (AddressSpace)
metadata *required*	ObjectMeta
spec *required*	io.enmasse.v1beta1.AddressSpaceSpec
status *optional*	io.enmasse.v1beta1.AddressSpaceStatus

D.1.3.29. io.enmasse.v1beta1.AddressSpaceList

Name	Description	Schema
apiVersion *required*	Default : **"enmasse.io/v1beta1"**	enum (enmasse.io/v1beta1)
items *required*		< io.enmasse.v1beta1.AddressSpace > array
kind *required*		enum (AddressSpaceList)

D.1.3.30. io.enmasse.v1beta1.AddressSpaceSpec

Name	Schema
authenticationService *optional*	authenticationService
endpoints *optional*	< endpoints > array
networkPolicy *optional*	networkPolicy
plan *required*	string
type *required*	io.enmasse.v1beta1.AddressSpaceType

authenticationService

Name	Schema
name *optional*	string
overrides *optional*	overrides
type *optional*	string

overrides

Name	Schema
host *optional*	string
port *optional*	integer
realm *optional*	string

endpoints

Name	Schema
cert *optional*	cert
exports *optional*	< exports > array
expose *optional*	expose
name *optional*	string
service *optional*	string

cert

Name	Schema
provider *optional*	string
secretName *optional*	string
tlsCert *optional*	string
tlsKey *optional*	string

exports

Name	Schema
kind *optional*	enum (ConfigMap, Secret, Service)
name *optional*	string

expose

Name	Schema
annotations *optional*	object
loadBalancerPorts *optional*	< string > array
loadBalancerSourceRanges *optional*	< string > array
routeHost *optional*	string
routeServicePort *optional*	string
routeTlsTermination *optional*	string
type *optional*	enum (route, loadbalancer)

networkPolicy

Name	Schema
egress *optional*	< io.k8s.api.networking.v1.NetworkPolicyEgressRule > array
ingress *optional*	< io.k8s.api.networking.v1.NetworkPolicyIngressRule > array

D.1.3.31. io.enmasse.v1beta1.AddressSpaceStatus

Name	Schema
endpointStatuses *optional*	< endpointStatuses > array
isReady *optional*	boolean
messages *optional*	< string > array

endpointStatuses

Name	Schema
cert *optional*	string
externalHost *optional*	string
externalPorts *optional*	< externalPorts > array
name *optional*	string
serviceHost *optional*	string
servicePorts *optional*	< servicePorts > array

externalPorts

Name	Schema
name *optional*	string
port *optional*	integer

servicePorts

Name	Schema
name *optional*	string
port *optional*	integer

D.1.3.32. io.enmasse.v1beta1.AddressSpaceType

AddressSpaceType is the type of address space (standard, brokered). Each type supports different types of addresses and semantics for those types.

Type : enum (standard, brokered)

D.1.3.33. io.enmasse.v1beta1.AddressSpec

Name	Schema
address *required*	string
plan *required*	string
type *required*	io.enmasse.v1beta1.AddressType

D.1.3.34. io.enmasse.v1beta1.AddressStatus

Name	Schema
isReady *optional*	boolean
messages *optional*	< string > array
phase *optional*	enum (Pending, Configuring, Active, Failed, Terminating)

D.1.3.35. io.enmasse.v1beta1.AddressType

Type of address (queue, topic, ...). Each address type support different kinds of messaging semantics.

Type : enum (queue, topic, anycast, multicast)

D.1.3.36. io.k8s.api.networking.v1.IPBlock

IPBlock describes a particular CIDR (Ex. "192.168.1.1/24") that is allowed to the pods matched by a NetworkPolicySpec's podSelector. The except entry describes CIDRs that should not be included within this rule.

Name	Description	Schema
cidr *required*	CIDR is a string representing the IP Block Valid examples are "192.168.1.1/24"	string
except *optional*	Except is a slice of CIDRs that should not be included within an IP Block Valid examples are "192.168.1.1/24" Except values will be rejected if they are outside the CIDR range	< string > array

D.1.3.37. io.k8s.api.networking.v1.NetworkPolicyEgressRule

NetworkPolicyEgressRule describes a particular set of traffic that is allowed out of pods matched by a NetworkPolicySpec's podSelector. The traffic must match both ports and to. This type is beta-level in 1.8

Name	Description	Schema
ports *optional*	List of destination ports for outgoing traffic. Each item in this list is combined using a logical OR. If this field is empty or missing, this rule matches all ports (traffic not restricted by port). If this field is present and contains at least one item, then this rule allows traffic only if the traffic matches at least one port in the list.	< io.k8s.api.networking.v1.NetworkPolicyPort > array
to *optional*	List of destinations for outgoing traffic of pods selected for this rule. Items in this list are combined using a logical OR operation. If this field is empty or missing, this rule matches all destinations (traffic not restricted by destination). If this field is present and contains at least one item, this rule allows traffic only if the traffic matches at least one item in the to list.	< io.k8s.api.networking.v1.NetworkPolicyPeer > array

D.1.3.38. io.k8s.api.networking.v1.NetworkPolicyIngressRule

NetworkPolicyIngressRule describes a particular set of traffic that is allowed to the pods matched by a NetworkPolicySpec's podSelector. The traffic must match both ports and from.

Name	Description	Schema
from *optional*	List of sources which should be able to access the pods selected for this rule. Items in this list are combined using a logical OR operation. If this field is empty or missing, this rule matches all sources (traffic not restricted by source). If this field is present and contains at least on item, this rule allows traffic only if the traffic matches at least one item in the from list.	< io.k8s.api.networking.v1.NetworkPolicyPeer > array
ports *optional*	List of ports which should be made accessible on the pods selected for this rule. Each item in this list is combined using a logical OR. If this field is empty or missing, this rule matches all ports (traffic not restricted by port). If this field is present and contains at least one item, then this rule allows traffic only if the traffic matches at least one port in the list.	< io.k8s.api.networking.v1.NetworkPolicyPort > array

D.1.3.39. io.k8s.api.networking.v1.NetworkPolicyPeer

NetworkPolicyPeer describes a peer to allow traffic from. Only certain combinations of fields are allowed

Name	Description	Schema
ipBlock *optional*	IPBlock defines policy on a particular IPBlock. If this field is set then neither of the other fields can be.	io.k8s.api.networking.v1.IPBlock

Name	Description	Schema
namespaceSelector *optional*	Selects Namespaces using cluster-scoped labels. This field follows standard label selector semantics; if present but empty, it selects all namespaces. If PodSelector is also set, then the NetworkPolicyPeer as a whole selects the Pods matching PodSelector in the Namespaces selected by NamespaceSelector. Otherwise it selects all Pods in the Namespaces selected by NamespaceSelector.	io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector
podSelector *optional*	This is a label selector which selects Pods. This field follows standard label selector semantics; if present but empty, it selects all pods. If NamespaceSelector is also set, then the NetworkPolicyPeer as a whole selects the Pods matching PodSelector in the Namespaces selected by NamespaceSelector. Otherwise it selects the Pods matching PodSelector in the policy's own Namespace.	io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector

D.1.3.40. io.k8s.api.networking.v1.NetworkPolicyPort

NetworkPolicyPort describes a port to allow traffic on

Name	Description	Schema
port *optional*	The port on the given protocol. This can either be a numerical or named port on a pod. If this field is not provided, this matches all port names and numbers.	io.k8s.apimachinery.pkg.util.intstr.IntOrString
protocol *optional*	The protocol (TCP or UDP) which traffic must match. If not specified, this field defaults to TCP.	string

D.1.3.41. io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelector

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Name	Description	Schema
matchExpressions *optional*	matchExpressions is a list of label selector requirements. The requirements are ANDed.	< io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelectorRequirement > array

Name	Description	Schema
matchLabels *optional*	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.	< string, string > map

D.1.3.42. io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelectorRequirement

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Name	Description	Schema
key *required*	key is the label key that the selector applies to.	string
operator *required*	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.	string
values *optional*	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.	< string > array

D.1.3.43. io.k8s.apimachinery.pkg.util.intstr.IntOrString

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

Type : string (int-or-string)

APPENDIX E. USING YOUR SUBSCRIPTION

AMQ Online is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

Accessing your account

1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

Activating a subscription

1. Go to access.redhat.com.
2. Navigate to **My Subscriptions**.
3. Navigate to **Activate a subscription** and enter your 16-digit activation number.

Downloading zip and tar files

To access zip or tar files, use the Red Hat Customer Portal to find the relevant files for download. If you are using RPM packages, this step is not required.

1. Open a browser and log in to the Red Hat Customer Portal **Product Downloads** page at access.redhat.com/downloads.
2. Locate the **Red Hat AMQ Online** entries in the **JBOSS INTEGRATION AND AUTOMATION** category.
3. Select the desired AMQ Online product. The Software Downloads page opens.
4. Click the **Download** link for your component.

Registering your system for packages

To install RPM packages on Red Hat Enterprise Linux, your system must be registered. If you are using zip or tar files, this step is not required.

1. Go to access.redhat.com.
2. Navigate to **Registration Assistant**.
3. Select your OS version and continue to the next page.
4. Use the listed command in your system terminal to complete the registration.

To learn more see How to Register and Subscribe a System to the Red Hat Customer Portal .

Revised on 2019-08-13 16:37:25 UTC