Abstract

This guide describes how to install and manage AMQ Online.
# Table of Contents

## CHAPTER 1. INTRODUCTION
- 1.1. AMQ ONLINE OVERVIEW .............................................................. 9
- 1.2. SUPPORTED FEATURES ................................................................. 9
- 1.3. AMQ ONLINE USER ROLES ............................................................ 10
- 1.4. SUPPORTED CONFIGURATIONS .................................................. 11
- 1.5. DOCUMENT CONVENTIONS .......................................................... 11
- 1.5.1. Variable text ................................................................. 11

## CHAPTER 2. INSTALLING AMQ ONLINE
- 2.1. DOWNLOADING AMQ ONLINE .................................................... 12
- 2.2. INSTALLING AMQ ONLINE USING A YAML BUNDLE ..................... 12
- 2.3. INSTALLING AMQ ONLINE USING ANSIBLE ................................ 13
- 2.4. INSTALLING AND CONFIGURING AMQ ONLINE USING THE OPERATOR LIFECYCLE MANAGER ...................................................... 15
  - 2.4.1. Installing AMQ Online from the OperatorHub using the OpenShift console ......................................................... 15
  - 2.4.2. Configuring AMQ Online using the OpenShift console ............. 16
    - 2.4.2.1. Creating an authentication service custom resource using the OpenShift console .................................. 16
    - 2.4.2.2. Creating an infrastructure configuration custom resource using the OpenShift console ..................... 17
    - 2.4.2.3. Creating an address space plan custom resource using the OpenShift console ................................ 17
    - 2.4.2.4. Creating an address plan custom resource using the OpenShift console ................................................ 18

## CHAPTER 3. UPGRADING AMQ ONLINE
- 3.1. UPGRADING AMQ ONLINE USING A YAML BUNDLE ..................... 20
- 3.2. UPGRADING AMQ ONLINE USING ANSIBLE .................................. 20

## CHAPTER 4. UNINSTALLING AMQ ONLINE
- 4.1. UNINSTALLING AMQ ONLINE USING THE YAML BUNDLE .................. 22
- 4.2. UNINSTALLING AMQ ONLINE USING ANSIBLE ................................ 22
- 4.3. UNINSTALLING AMQ ONLINE USING THE OPERATOR LIFECYCLE MANAGER (OLM) ............................................................... 23
- 4.4. UNINSTALLING AMQ ONLINE USING THE OPENSHIFT CONSOLE ........ 23

## CHAPTER 5. CONFIGURING AMQ ONLINE
- 5.1. MINIMAL SERVICE CONFIGURATION ........................................... 25
- 5.2. ADDRESS SPACE PLANS ................................................................. 26
- 5.3. CREATING ADDRESS SPACE PLANS ............................................. 26
- 5.4. ADDRESS PLANS ........................................................................ 27
- 5.5. CREATING ADDRESS PLANS ........................................................ 29
- 5.6. INFRASTRUCTURE CONFIGURATION ............................................. 30
  - 5.6.1. Brokered infrastructure configuration .................................... 30
    - 5.6.1.1. Brokered infrastructure configuration example ............... 30
    - 5.6.1.2. Overriding the probe timing for brokered infrastructure configuration .................................................. 32
  - 5.6.2. Standard infrastructure configuration .................................. 32
    - 5.6.2.1. Standard infrastructure configuration example ............. 33
    - 5.6.2.2. Overriding the probe timing for standard infrastructure configuration ........................................... 34
- 5.7. CREATING AND EDITING INFRASTRUCTURE CONFIGURATIONS ....... 35
- 5.8. AUTHENTICATION SERVICES ......................................................... 36
  - 5.8.1. Standard authentication service .......................................... 36
    - 5.8.1.1. Standard authentication service example ....................... 37
    - 5.8.1.2. Deploying the standard authentication service .............. 37
    - 5.8.1.3. Deploying the standard authentication service for high availability (HA) ...................................................... 38
  - 5.8.2. External authentication service ............................................. 39
    - 5.8.2.1. External authentication service example ....................... 39
    - 5.8.2.2. External authentication service example allowing overrides .............................................................. 39
5.8.2.3. External authentication server API
5.8.2.3.1. Authentication
5.8.2.3.2. Authorization
5.8.3. None authentication service
5.8.3.1. Deploying the none authentication service
5.9. AMQ ONLINE EXAMPLE ROLES

CHAPTER 6. MONITORING AMQ ONLINE
6.1. ENABLING MONITORING ON OPENSIFT 4
6.2. (OPTIONAL) DEPLOYING THE APPLICATION MONITORING OPERATOR
6.3. (OPTIONAL) DEPLOYING THE KUBE-STATE-METRICS AGENT
6.4. ENABLING MONITORING
6.5. CONFIGURING ALERT NOTIFICATIONS
6.6. METRICS AND RULES
   6.6.1. Common metrics
   6.6.2. Address space controller metrics
      6.6.2.1. Summary
   6.6.3. Standard controller and agent metrics
      6.6.3.1. Summary
   6.6.4. Rules
      6.6.4.1. Records
         6.6.4.2. Alerts
6.7. ENABLING TENANT METRICS
6.8. USING QDSTAT
   6.8.1. Viewing router connections using qdstat
   6.8.2. Viewing router addresses using qdstat
   6.8.3. Viewing router links using qdstat
   6.8.4. Viewing link routes using qdstat

CHAPTER 7. OPERATION PROCEDURES FOR AMQ ONLINE
7.1. RESTARTING COMPONENTS TO ACQUIRE SECURITY FIXES
   7.1.1. Restarting Operators
   7.1.2. Restarting authentication services
   7.1.3. Restarting routers
   7.1.4. Restarting brokers
7.2. VIEWING ROUTER LOGS
7.3. VIEWING BROKER LOGS
7.4. ENABLING AN AMQP PROTOCOL TRACE FOR THE ROUTER
   7.4.1. Dynamically enabling the protocol trace for a single router
   7.4.2. Enabling the protocol trace using the StandardInfraConfig environment variable
7.5. ENABLING AN AMQP PROTOCOL TRACE FOR THE BROKER
7.6. EXAMINING THE STATE OF A BROKER USING THE AMQ BROKER MANAGEMENT INTERFACES
7.7. CHANGING THE BROKER LOGGING LEVEL AT RUNTIME

CHAPTER 8. AMQ ONLINE CONFIGURATION SIZING GUIDELINES
8.1. BROKER COMPONENT SIZING
   8.1.1. Example use case for a broker component configuration
      8.1.1.1. Example broker component configuration without paging
         8.1.1.2. Example broker component configuration with paging
   8.1.2. Broker scaling (standard address space only)
8.2. ROUTER COMPONENT SIZING
   8.2.1. Example use case for router component sizing
   8.2.2. High availability (HA)
   8.2.3. Router scaling
D.1.2.4.3. Responses
D.1.2.4.4. Produces
D.1.2.4.5. Tags
D.1.2.5. DELETE /apis/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}
  D.1.2.5.1. Description
  D.1.2.5.2. Parameters
  D.1.2.5.3. Responses
  D.1.2.5.4. Produces
  D.1.2.5.5. Tags
D.1.2.6. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses
  D.1.2.6.1. Description
  D.1.2.6.2. Parameters
  D.1.2.6.3. Responses
  D.1.2.6.4. Consumes
  D.1.2.6.5. Produces
  D.1.2.6.6. Tags
D.1.2.7. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses
  D.1.2.7.1. Description
  D.1.2.7.2. Parameters
  D.1.2.7.3. Responses
  D.1.2.7.4. Produces
  D.1.2.7.5. Tags
D.1.2.8. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}
  D.1.2.8.1. Description
  D.1.2.8.2. Parameters
  D.1.2.8.3. Responses
  D.1.2.8.4. Consumes
  D.1.2.8.5. Produces
  D.1.2.8.6. Tags
D.1.2.9. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}
  D.1.2.9.1. Description
  D.1.2.9.2. Parameters
  D.1.2.9.3. Responses
  D.1.2.9.4. Produces
  D.1.2.9.5. Tags
D.1.2.10. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}
  D.1.2.10.1. Description
  D.1.2.10.2. Parameters
  D.1.2.10.3. Responses
  D.1.2.10.4. Produces
  D.1.2.10.5. Tags
D.1.2.11. PATCH /apis/enmasse.io/v1beta1/namespaces/{namespace}/addresses/{name}
  D.1.2.11.1. Description
  D.1.2.11.2. Parameters
  D.1.2.11.3. Responses
  D.1.2.11.4. Consumes
  D.1.2.11.5. Produces
  D.1.2.11.6. Tags
D.1.2.12. POST /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressespaces
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1.1.2.12.4. Consumes</td>
<td>95</td>
</tr>
<tr>
<td>D.1.1.2.12.5. Produces</td>
<td>95</td>
</tr>
<tr>
<td>D.1.1.2.12.6. Tags</td>
<td>95</td>
</tr>
<tr>
<td>D.1.1.2.13. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces</td>
<td>95</td>
</tr>
<tr>
<td>D.1.1.2.13.1. Description</td>
<td>95</td>
</tr>
<tr>
<td>D.1.1.2.13.2. Parameters</td>
<td>95</td>
</tr>
<tr>
<td>D.1.1.2.13.3. Responses</td>
<td>95</td>
</tr>
<tr>
<td>D.1.1.2.13.4. Produces</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.13.5. Tags</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.14. GET /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.14.1. Description</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.14.2. Parameters</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.14.3. Responses</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.14.4. Consumes</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.14.5. Produces</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.14.6. Tags</td>
<td>96</td>
</tr>
<tr>
<td>D.1.1.2.15. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.15.1. Description</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.15.2. Parameters</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.15.3. Responses</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.15.4. Produces</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.15.5. Tags</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.16. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.16.1. Description</td>
<td>97</td>
</tr>
<tr>
<td>D.1.1.2.16.2. Parameters</td>
<td>98</td>
</tr>
<tr>
<td>D.1.1.2.16.3. Responses</td>
<td>98</td>
</tr>
<tr>
<td>D.1.1.2.16.4. Produces</td>
<td>98</td>
</tr>
<tr>
<td>D.1.1.2.16.5. Tags</td>
<td>98</td>
</tr>
<tr>
<td>D.1.1.2.17. PATCH /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}</td>
<td>98</td>
</tr>
<tr>
<td>D.1.1.2.17.1. Description</td>
<td>98</td>
</tr>
<tr>
<td>D.1.1.2.17.2. Parameters</td>
<td>98</td>
</tr>
<tr>
<td>D.1.1.2.17.3. Responses</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1.2.17.4. Consumes</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1.2.17.5. Produces</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1.2.17.6. Tags</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1.2.18. POST /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1.2.18.1. Description</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1.2.18.2. Parameters</td>
<td>99</td>
</tr>
<tr>
<td>D.1.1.2.18.3. Responses</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.18.4. Consumes</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.18.5. Produces</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.18.6. Tags</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.19. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.19.1. Description</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.19.2. Parameters</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.19.3. Responses</td>
<td>100</td>
</tr>
<tr>
<td>D.1.1.2.19.4. Produces</td>
<td>101</td>
</tr>
<tr>
<td>D.1.1.2.19.5. Tags</td>
<td>101</td>
</tr>
<tr>
<td>D.1.1.2.20. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}</td>
<td>101</td>
</tr>
<tr>
<td>D.1.1.2.20.1. Description</td>
<td>101</td>
</tr>
<tr>
<td>D.1.1.2.20.2. Parameters</td>
<td>101</td>
</tr>
<tr>
<td>D.1.1.2.20.3. Responses</td>
<td>101</td>
</tr>
<tr>
<td>D.1.1.2.20.4. Consumes</td>
<td>102</td>
</tr>
</tbody>
</table>
D.1.2.20.5. Produces
D.1.2.20.6. Tags

D.1.2.21. PUT /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}
D.1.2.21.1. Description
D.1.2.21.2. Parameters
D.1.2.21.3. Responses
D.1.2.21.4. Produces
D.1.2.21.5. Tags

D.1.2.22. DELETE /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}
D.1.2.22.1. Description
D.1.2.22.2. Parameters
D.1.2.22.3. Responses
D.1.2.22.4. Produces
D.1.2.22.5. Tags

D.1.2.23. PATCH /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}
D.1.2.23.1. Description
D.1.2.23.2. Parameters
D.1.2.23.3. Responses
D.1.2.23.4. Consumes
D.1.2.23.5. Produces
D.1.2.23.6. Tags

D.1.3. Definitions

D.1.3.1. JsonPatchRequest
D.1.3.2. ObjectMeta
D.1.3.3. Patch
D.1.3.4. Status

D.1.3.5. io.enmasse.admin.v1beta1.BrokeredInfraConfig
D.1.3.6. io.enmasse.admin.v1beta1.BrokeredInfraConfigList
D.1.3.7. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec
D.1.3.8. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin
D.1.3.9. io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker
D.1.3.10. io.enmasse.admin.v1beta1.InfraConfigPodSpec
D.1.3.11. io.enmasse.admin.v1beta1.StandardInfraConfig
D.1.3.12. io.enmasse.admin.v1beta1.StandardInfraConfigList
D.1.3.13. io.enmasse.admin.v1beta1.StandardInfraConfigSpec
D.1.3.15. io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker
D.1.3.16. io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter
D.1.3.17. io.enmasse.admin.v1beta2.AddressPlan
D.1.3.18. io.enmasse.admin.v1beta2.AddressPlanList
D.1.3.19. io.enmasse.admin.v1beta2.AddressPlanSpec
D.1.3.20. io.enmasse.admin.v1beta2.AddressSpacePlan
D.1.3.21. io.enmasse.admin.v1beta2.AddressSpacePlanList
D.1.3.22. io.enmasse.admin.v1beta2.AddressSpacePlanSpec
D.1.3.23. io.enmasse.user.v1beta1.MessagingUser
D.1.3.24. io.enmasse.user.v1beta1.MessagingUserList
D.1.3.25. io.enmasse.user.v1beta1.UserSpec
D.1.3.26. io.enmasse.v1beta1.Address
D.1.3.27. io.enmasse.v1beta1.AddressList
D.1.3.28. io.enmasse.v1beta1.AddressSpace
D.1.3.29. io.enmasse.v1beta1.AddressSpaceList
D.1.3.30. io.enmasse.v1beta1.AddressSpaceSpec
D.1.3.31. io.enmasse.v1beta1.AddressSpaceSpecConnector
APPENDIX E. USING YOUR SUBSCRIPTION

Accessing your account ................................................. 133
Activating a subscription .............................................. 133
Downloading zip and tar files ........................................ 133
Registering your system for packages ......................... 133
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

Figure 1.2. Brokered address space

1.2. SUPPORTED FEATURES

The following table shows the supported features for AMQ Online 1.6:

Table 1.1. Supported features reference table

<table>
<thead>
<tr>
<th>Feature</th>
<th>Brokered address space</th>
<th>Standard address space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address type</td>
<td>Queue</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Brokered address space</td>
<td>Standard address space</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Topic</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Multicast</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Anycast</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Subscription</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Messaging protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMQP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MQTT</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CORE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OpenWire</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>STOMP</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Transports</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WebSocket</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Durable subscriptions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMS durable subscriptions</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>&quot;Named&quot; durable subscriptions</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>JMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Selectors on queues</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Message ordering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>guarantees (including prioritization)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Scalability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scalable distributed queues and topics</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 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* 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-authts/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 3.11, see the OpenShift Container Platform 3.11 documentation.
  - For more information about how to install the CLI on OpenShift 4.x, see the OpenShift Container Platform 4.5 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
   ```

9. (Optional) Install the Service Catalog integration:

   ```
   oc apply -f install/components/service-broker
   oc apply -f install/components/cluster-service-broker
   ```

### 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=amq-online-infra
enable_rbac=False
api_server=True
service_catalog=False
register_api_server=True
keycloak_admin_password=admin
authentication_services=["standard"]
standard_authentication_service_postgresql=False
monitoring_namespace=enmasse-monitoring
monitoring_operator=False
monitoring=False
```

The following Ansible configuration settings are supported:

**Table 2.1. Ansible configuration settings**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default value</th>
<th>Required</th>
</tr>
</thead>
</table>

---

13
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>Specifies the project where AMQ Online is installed.</td>
<td>Not applicable</td>
<td>yes</td>
</tr>
<tr>
<td>enable_rbac</td>
<td>Specifies whether to enable RBAC authentication of REST APIs</td>
<td>True</td>
<td>no</td>
</tr>
<tr>
<td>service_catalog</td>
<td>Specifies whether to enable integration with the Service Catalog</td>
<td>False</td>
<td>no</td>
</tr>
<tr>
<td>authentication_services</td>
<td>Specifies the list of authentication services to deploy. Supported values are none and standard.</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>keycloak_admin_password</td>
<td>Specifies the admin password to use for the standard authentication service Red Hat Single Sign-On instance</td>
<td>Not applicable</td>
<td>yes (if standard authentication service is enabled)</td>
</tr>
<tr>
<td>api_server</td>
<td>Specifies whether to enable the REST API server</td>
<td>True</td>
<td>no</td>
</tr>
<tr>
<td>register_api_server</td>
<td>Specifies whether to register the API server with OpenShift master</td>
<td>False</td>
<td>no</td>
</tr>
<tr>
<td>secure_api_server</td>
<td>Specifies whether to enable mutual TLS for the API server</td>
<td>False</td>
<td>no</td>
</tr>
<tr>
<td>install_example_plans</td>
<td>Specifies whether to install example plans and infrastructure configurations</td>
<td>True</td>
<td>no</td>
</tr>
<tr>
<td>monitoring_namespace</td>
<td>Specifies the project where AMQ Online monitoring is installed.</td>
<td>Not applicable</td>
<td>yes</td>
</tr>
<tr>
<td>monitoring_operator</td>
<td>Specifies whether to install the monitoring infrastructure</td>
<td>Not applicable</td>
<td>no</td>
</tr>
</tbody>
</table>

**Procedure**

1. Create an inventory file.
2. Run the Ansible playbook:
2.4. INSTALLING AND CONFIGURING AMQ ONLINE USING THE OPERATOR LIFECYCLE MANAGER

You can use the Operator Lifecycle Manager to install and configure an instance of AMQ Online.

In OpenShift 4.x, the Operator Lifecycle Manager (OLM) helps users install, update, and manage the life cycle of all Operators and their associated services running across their clusters. It is part of the Operator Framework, an open source toolkit designed to manage Kubernetes native applications (Operators) in an effective, automated, and scalable way.

The OLM runs by default in OpenShift 4.x, which aids cluster administrators in installing, upgrading, and granting access to Operators running on their cluster. The OpenShift console provides management screens for cluster administrators to install Operators and grant specific projects access to use the catalog of Operators available on the cluster.

OperatorHub is the graphical interface that OpenShift cluster administrators use to discover, install, and upgrade Operators. With one click, these Operators can be pulled from OperatorHub, installed on the cluster, and managed by the OLM, ready for engineering teams to self-service manage the software in development, test, and production environments.

2.4.1. Installing AMQ Online from the OperatorHub using the OpenShift console

You can install the AMQ Online Operator on an OpenShift 4.x cluster by using OperatorHub in the OpenShift console.

Prerequisites

- Access to an OpenShift 4.x cluster and an account with `cluster-admin` permissions.

Procedure

1. In the OpenShift 4.x console, log in using an account with `cluster-admin` privileges.

2. To create the project where you want to deploy AMQ Online, click Home > Projects, and then click Create Project. The Create Project window opens.

3. In the Name field, type `amq-online-infra` and click Create. The `amq-online-infra` project is created.

4. Click Operators > OperatorHub.

5. In the Filter by keyword box, type AMQ Online to find the AMQ Online Operator.

6. Click the AMQ Online Operator. Information about the Operator is displayed.

7. Read the information about the Operator and click Install. The Create Operator Subscription page opens.

8. On the Create Operator Subscription page, for Installation Mode, click A specific namespace on the cluster, and then select the `amq-online-infra` namespace from the drop-down list.

9. Accept all of the remaining default selections and click Subscribe.
The amq-online page is displayed, where you can monitor the installation progress of the AMQ Online Operator subscription.

10. After the subscription upgrade status is shown as **Up to date**, click **Operators > Installed Operators** to verify that the AMQ Online ClusterServiceVersion (CSV) is displayed and its **Status** ultimately resolves to **InstallSucceeded** in the **amq-online-infra** namespace. For troubleshooting information, see the OpenShift documentation.

**Next steps**

- Configure AMQ Online using the OpenShift console

### 2.4.2. Configuring AMQ Online using the OpenShift console

After installing AMQ Online from the OperatorHub using the OpenShift console, create a new instance of a custom resource for the following items within the **amq-online-infra** project:

- an authentication service
- infrastructure configuration for an address space type (the example uses the standard address space type)
- an address space plan
- an address plan

After creating the new instances of the custom resources, next:

- create an address space
- create an address
- create a messaging user

The following procedures use the example data that is provided when using the OpenShift console.

#### 2.4.2.1. Creating an authentication service custom resource using the OpenShift console

You must create a custom resource for an authentication service to use AMQ Online. This example uses the standard authentication service.

**Procedure**

1. In the top right, click the **Plus** icon (+). The Import YAML window opens.
2. From the top left drop-down menu, select the **amq-online-infra** project.
3. Copy the following code:

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: standard-authservice
spec:
  type: standard
```
4. In the Import YAML window, paste the copied code and click Create. The AuthenticationService overview page is displayed.

5. Click Workloads > Pods. In the Readiness column, the Pod status is Ready when the custom resource has been deployed.

Next steps

- Create an infrastructure configuration custom resource using the OpenShift console

2.4.2.2. Creating an infrastructure configuration custom resource using the OpenShift console

You must create an infrastructure configuration custom resource to use AMQ Online. This example uses StandardInfraConfig for a standard address space.

Procedure

1. In the top right, click the Plus icon (+). The Import YAML window opens.

2. From the top left drop-down menu, select the amq-online-infra project.

3. Copy the following code:

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

4. In the Import YAML window, paste the copied code and click Create. The StandardInfraConfig overview page is displayed.

5. Click Operators > Installed Operators.

6. Click the AMQ Online Operator and click the Standard Infra Config tab to verify that its Status displays as Active.

Next steps

- Create an address space plan custom resource using the OpenShift console

2.4.2.3. Creating an address space plan custom resource using the OpenShift console

You must create an address space plan custom resource to use AMQ Online. This procedure uses the example data that is provided when using the OpenShift console.

Procedure

1. In the top right, click the Plus icon (+). The Import YAML window opens.

2. From the top left drop-down menu, select the amq-online-infra project.

3. Copy the following code:

```yaml
apiVersion: admin.enmasse.io/v1beta2
```
In the Import YAML window, paste the copied code and click Create. The AddressSpacePlan overview page is displayed.

Click Operators > Installed Operators

Click the AMQ Online Operator and click the Address Space Plan tab to verify that its Status displays as Active.

Next steps

- Create an address plan custom resource using the OpenShift console

2.4.2.4. Creating an address plan custom resource using the OpenShift console

You must create an address plan custom resource to use AMQ Online. This procedure uses the example data that is provided when using the OpenShift console.

Procedure

1. In the top right, click the Plus icon (+). The Import YAML window opens.

2. From the top left drop-down menu, select the amq-online-infra project.

3. Copy the following code:

```yaml
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: standard-small-queue
spec:
  addressType: queue
  resources:
    router: 0.01
    broker: 0.1

4. In the Import YAML window, paste the copied code and click Create. The AddressPlan overview page is displayed.

5. Click Operators > Installed Operators

6. Click the AMQ Online Operator and click the Address Plan tab to verify that its Status displays as Active.
Next steps

- create an address space
- create an address
- create a messaging user
CHAPTER 3. UPGRAADING 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.

Using the same method that was used to initially install AMQ Online to upgrade to a newer version of AMQ Online is recommended.

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

3.1. UPGRAADING 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:
   ```bash
   oc login -u system:admin
   ```
2. Select the project where AMQ Online is installed:
   ```bash
   oc project amq-online-infra
   ```
3. Apply the new release bundle:
   ```bash
   oc apply -f install/bundles/amq-online
   ```
4. Monitor pods while they are restarted:
   ```bash
   oc get pods -w
   ```
   The pods restart and become active within several minutes.
5. Delete api-server resources not needed after upgrade:
   ```bash
   oc delete sa api-server -n amq-online-infra
   oc delete clusterrolebinding enmasse.io:api-server-amq-online-infra
   oc delete clusterrole enmasse.io:api-server
   oc delete rolebinding api-server -n amq-online-infra
   oc delete role enmasse.io:api-server -n amq-online-infra
   ```

3.2. UPGRAADING 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:
   ```bash
   oc login -u system:admin
   ```

2. Run the Ansible playbook from the new release:
   ```bash
   ansible-playbook -i inventory-file ansible/playbooks/openshift/deploy_all.yml
   ```

3. Monitor pods while they are restarted:
   ```bash
   oc get pods -w
   ```
   The pods restart and become active within several minutes.

4. Delete `api-server` resources not needed after upgrade:
   ```bash
   oc delete sa api-server -n amq-online-infra
   oc delete clusterrolebinding enmasse.io:api-server-amq-online-infra
   oc delete clusterrole enmasse.io:api-server
   oc delete rolebinding api-server -n amq-online-infra
   oc delete role enmasse.io:api-server -n amq-online-infra
   ```
CHAPTER 4. UNINSTALLING AMQ ONLINE

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

4.1. 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 crd -l app=enmasse`
   `oc delete crd -l app=enmasse --timeout=600s`
   `oc delete clusterrolebindings -l app=enmasse`
   `oc delete clusterroles -l app=enmasse`
   `oc delete apiservices -l app=enmasse`
   `oc delete oauthclients -l app=enmasse`

3. (OpenShift 4) Delete the console integration:
   
   `oc delete consolelinks -l app=enmasse`

4. (Optional) Delete the service catalog integration:
   
   `oc delete clusterservicebrokers -l app=enmasse`

5. Delete the project where AMQ Online is deployed:
   
   `oc delete project amq-online-infra`

4.2. 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`
4.3. UNINSTALLING AMQ ONLINE USING THE OPERATOR LIFECYCLE MANAGER (OLM)

This method uninstalls AMQ Online that was installed using the Operator Lifecycle Manager (OLM).

Procedure

1. Log in as a user with `cluster-admin` privileges:
   
   ```bash
   oc login -u system:admin
   ```

2. Remove all `AddressSpace` instances:
   
   ```bash
   oc delete addressspaces -A --all --timeout=600s
   ```

3. Delete the subscription (replace `amq-online` with the name of the subscription used in the installation):
   
   ```bash
   oc delete subscription amq-online -n amq-online-infra
   ```

4. Remove the CSV for the Operator:
   
   ```bash
   oc delete csv -l app=enmasse -n amq-online-infra
   ```

5. Remove any remaining resources (replace `amq-online-infra` with the project where you installed AMQ Online):
   
   ```bash
   oc delete all -l app=enmasse -n amq-online-infra
   oc delete cm -l app=enmasse -n amq-online-infra
   oc delete secret -l app=enmasse amq-online-infra
   oc delete consolelinks -l app=enmasse
   oc delete oauthclients -l app=enmasse
   oc delete crd -l app=enmasse
   ```

6. (Optional: Skip this step if AMQ Online is installed in the `openshift-operators` namespace)
   Delete the namespace where AMQ Online was installed:
   
   ```bash
   oc delete namespace amq-online-infra
   ```

4.4. UNINSTALLING AMQ ONLINE USING THE OPENSHIFT CONSOLE

This method uninstalls AMQ Online that was installed using the Operator Lifecycle Manager (OLM) in the OpenShift Container Platform console.

Procedure

1. From the Project list, select the project where you installed AMQ Online.

2. Click `Catalog → Operator Management`. The Operator Management page opens.

3. Click the `Operator Subscriptions` tab.
4. Find the AMQ Online Operator you want to uninstall. In the far right column, click the vertical ellipsis icon and select **Remove Subscription**.

5. When prompted by the Remove Subscription window, select the **Also completely remove the AMQ Online Operator from the selected namespace** check box to remove all components related to the installation.
   - Click **Remove**. The AMQ Online Operator will stop running and no longer receive updates.

6. Remove any remaining resources by running the following command (replace **amq-online-infra** with the project where you installed AMQ Online):

   ```bash
   oc delete all -l app=enmasse -n amq-online-infra
   oc delete cm -l app=enmasse -n amq-online-infra
   oc delete secret -l app=enmasse amq-online-infra
   oc delete consolelinks -l app=enmasse
   oc delete oauthclients -l app=enmasse
   ```

7. (Optional: Skip this step if AMQ Online is installed in the **openshift-operators** namespace) Delete the namespace where AMQ Online was installed:

   ```bash
   oc delete namespace amq-online-infra
   ```
CHAPTER 5. CONFIGURING AMQ ONLINE

5.1. 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:

```yaml
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`
5.2. 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:

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

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.

5.3. 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:

    ```yaml
    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

### 5.4. 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:
The address type to which this plan applies.

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

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

(Optional) Restricts message time-to-live (TTL). Applies to address types queue and topic only.

(Optional) Provides message redelivery settings. Applies to address types queue and subscription only.

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 across multiple brokers for HA and improved performance. Specifying an amount of broker resource above 1 will automatically cause a queue to be partitioned.

The messageTtl field is used to restrict the effective absolute-expiry-time of any message put to a queue or topic. The maximum and minimum values are defined in milliseconds. The system adjusts the TTL value of an incoming message to a particular address based on these values:

- If a messages arrives at the address with a TTL value that is greater than the maximum value, the system changes the message TTL to the maximum value.
If a message arrives at the address with a TTL value that is less than the minimum value, the system changes the message TTL to the minimum value.

Messages that arrive without a TTL defined are considered to have a TTL value of infinity.

The messageRedelivery field is used to control the behavior of a queue or subscription when a receiver is unsuccessful in its processing a message and the system returns the message to the queue for redelivery. By default, the system redelivers messages indefinitely. However, the messageRedelivery field allows you to modify the behavior either by delaying redelivery or redirecting the message to a deadletter address.

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:

```yaml
apiVersion: admin.enmasse.io/v1beta2
tool: 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.

### 5.5. 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
tool: AddressPlan
metadata:
  name: small-anycast
```
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

5.6. 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.

5.6.1. Brokered infrastructure configuration

BrokeredInfraConfig resources are used to configure infrastructure deployed by brokered address spaces. Address space plans reference the brokered infrastructure configuration using the spec.infraConfigRef field. For more information about address space plans, see Address space plans.

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

5.6.1.1. Brokered infrastructure configuration example

The following example of a brokered infrastructure configuration file shows the various settings that can be specified.

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: brokered-infra-config-example
spec:
  version: "0.33"
  admin:
    resources:
      memory: 256Mi
    podTemplate:
      metadata:
```
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 is assumed to be the same version as the controllers reading the configuration.

Specifies the settings you can configure for the **admin** components.

Specifies the settings you can configure for the **broker** components. Note that changing the `.broker.resources.storage` setting does not configure the existing broker storage size.

For both **admin** and **broker** components you can configure the following **podTemplate** elements:

- `metadata.labels`
- `spec.priorityClassName`
- `spec.tolerations`
- `spec.affinity`
- `spec.containers.readinessProbe`
- `spec.containers.livenessProbe`
- `spec.containers.resources`
- `spec.containers.env`

All other **podTemplate** elements are ignored. For more information about these elements, see the OpenShift documentation in the following Related links section.

For more information about how to set a readiness probe timeout, see **Overriding the readiness probe timing for brokered infrastructure configuration**.

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

**Related links**

- For more information about the **podTemplate** settings, see the following OpenShift documentation:
  - **Pod priority**
  - **Taints and tolerations**
5.6.1.2. Overriding the probe timing for brokered infrastructure configuration

You can override the default values for the probe timing on broker resources. You might want to change the default values if, for example, it takes longer than expected for the broker storage to become available, or a server is slow.

The following example shows how to override certain default values of the readiness probe for broker resources.

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: brokered-infra-config
spec:
  broker:
    ...
  podTemplate:
    spec:
      containers:
        - name: broker
          readinessProbe:
            failureThreshold: 6
            initialDelaySeconds: 20

1. The `name` value must match the target container name. For a broker, the `podTemplate` name is `broker`.

2. Specifies the number of times that OpenShift tries when a Pod starts and the probe fails before either the Pod is marked Unready for a readiness probe, or restarting the container for a liveness probe. The default value is 3, and the minimum value is 1.

3. Specifies the number of seconds before performing the first probe after the container starts.

Related links

- OpenShift 3.11 documentation on overriding probe timeouts
- OpenShift 4.x documentation on overriding probe timeouts

5.6.2. Standard infrastructure configuration

`StandardInfraConfig` resources are used to configure infrastructure deployed by `standard` address spaces. Address space plans reference the standard infrastructure configuration using the `spec.infraConfigRef` field. For more information about address space plans, see `Address space plans`.

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

5.6.2.1. Standard infrastructure configuration example

The following example of a standard infrastructure configuration file shows the various settings that can be specified.

apiVersion: admin.enmasse.io/v1beta1
category: StandardInfraConfig
metadata:
  name: myconfig
spec:
  version: "0.33"
  admin:
    resources:
      memory: 256Mi
  broker:
    resources:
      cpu: 0.5
      memory: 2Gi
      storage: 100Gi
      addressFullPolicy: PAGE
  router:
    resources:
      cpu: 1
      memory: 256Mi
      linkCapacity: 1000
      minReplicas: 1
      policy:
        maxConnections: 1000
        maxConnectionsPerHost: 1
        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

1 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 is assumed to be the same version as the controllers reading the configuration.

2 Specifies the settings you can configure for the admin components.
Specifies the settings you can configure for the broker components. Changing the `.broker.resources.storage` setting does not configure the existing broker storage size.

Specifies the settings you can configure for the router components.

For admin, broker, and router components you can configure the following podTemplate elements:

- metadata.labels
- spec.priorityClassName
- spec.tolerations
- spec.affinity
- spec.containers.resources
- spec.containers.readinessProbe
- spec.containers.livenessProbe
- spec.containers.env

All other podTemplate elements are ignored. For more information about these elements, see the OpenShift documentation in the following Related links section.

For more information about how to set a readiness probe timeout, see Overriding the readiness probe timing for standard infrastructure configuration.

Related links

- For more information about the podTemplate settings, see the following OpenShift documentation:
  - Pod priority
  - Taints and tolerations
  - Affinity and anti-affinity
  - Application health
  - Compute resources
  - Environment variables

5.6.2.2. Overriding the probe timing for standard infrastructure configuration

You can override the default values for probe timing on broker and router resources. You might want to change the default values if, for example, it takes longer than expected for the broker storage to become available, or a server is slow.
The following example shows how to override certain default values of the readiness probe timeout for a broker resource and a liveness probe for a router resource.

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: standard-infra-config
spec:
  broker:
    ...
    podTemplate:
      spec:
        containers:
          - name: broker
            readinessProbe:
              failureThreshold: 6
              initialDelaySeconds: 20
  router:
    ...
    podTemplate:
      spec:
        containers:
          - name: router
            livenessProbe:
              failureThreshold: 6
              initialDelaySeconds: 20
```

1. The `name` value must match the target container name. For example, for a broker `podTemplate`, `name` is `broker`, and for a router `podTemplate`, it is `router`.

2. Specifies the number of times that OpenShift tries when a Pod starts and the probe fails before either the Pod is marked `Unready` for a readiness probe, or restarting the container for a liveness probe. The default value is 3, and the minimum value is 1.

3. Specifies the number of seconds before performing the first probe after the container starts.

Related links

- OpenShift 3.11 documentation about liveness and readiness probes (application health)
- OpenShift 4.x documentation about liveness and readiness probes (application health)

5.7. 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:

   ```bash
   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: "0.33"
     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.

5.8. 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**.

5.8.1. 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.
5.8.1.1. Standard authentication service example

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

```yaml
apiVersion: admin.enmasse.io/v1beta1
class: AuthenticationService
metadata:
  name: standard
spec:
  type: standard
  standard:
    credentialsSecret:
      name: my-admin-credentials
    certificateSecret:
      name: my-authservice-certificate
    resources:
      requests:
        memory: 2Gi
      limits:
        memory: 2Gi
    storage:
      type: persistent-claim
      size: 5Gi
    datasource:
      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.

5.8.1.2. Deploying the standard authentication service

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

**Procedure**

1. Log in as a service admin:
2. Change to the project where AMQ Online is installed:

   oc project amq-online-infra

3. Create an **AuthenticationService** definition:

   ```yaml
   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

5.8.1.3. Deploying the **standard** authentication service for high availability (HA)

For production deployment, the authentication services should be setup for high availability in order to reduce downtime during OpenShift updates or in the event of a node failure. To implement the **standard** authentication service in HA mode, you deploy it using a PostgreSQL database as the backend.

**Prerequisites**

- A PostgreSQL database.

**Procedure**

1. Log in as a service admin:

   oc login -u admin

2. Create a secret with the database credentials:

   oc create secret generic db-creds -n amq-online-infra --from-literal=database-user=admin --from-literal=database-password=secure-password

3. Create an **AuthenticationService** definition:

   ```yaml
   apiVersion: admin.enmasse.io/v1beta1
   kind: AuthenticationService
   metadata:
     name: standard-authservice
   spec:
     type: standard
     standard:
       replicas: 2
       datasource:
   ```
5.8.2. 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.

5.8.2.1. External authentication service example

The following example shows an authentication service of type external:

```yaml
apiVersion: admin.enmasse.io/v1beta1
group: AuthenticationService
metadata:
  name: my-external-1
spec:
  type: external
  realm: myrealm
  external:
    host: example.com
    port: 5671
    caCertSecret: my-ca-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.

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

5.8.2.2. 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:

```yaml
apiVersion: admin.enmasse.io/v1beta1
group: AuthenticationService
metadata:
  name: my-external-2
spec:
  type: external
  realm: myrealm
  external:
    host: example.com
    port: 5671
    caCertSecret: my-ca-cert
```
(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.

The host name of the external authentication server.

The port number of the external authentication server.

(Optional) The CA certificate to trust when connecting to the authentication server.

(Optional) Specifies whether address space overrides are allowed to the host name, port number, realm, and CA certificate. 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](#).

### 5.8.2.3. 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.

#### 5.8.2.3.1. 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.

```json
{
   "authenticated-identity": {
      "sub": "myid",
      "preferred_username": "myuser"
   }
}
```

#### 5.8.2.3.2. 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**:

```json
{
    "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.

### 5.8.3. 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.

### 5.8.3.1. Deploying the **none** authentication service

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

**Procedure**

1. Log in as a service admin:

   ```sh
   oc login -u admin
   ```

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

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

3. Create an **AuthenticationService** definition:

   ```yaml
   apiVersion: admin.enmasse.io/v1beta1
   ```
4. Deploy the authentication service:

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

## 5.9. 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 5.1. AMQ Online example roles table

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enmasse.io:tenant-view</td>
<td>Specifies <strong>get</strong> and <strong>list</strong> permissions for <strong>addresses</strong>, <strong>addressspaces</strong>, <strong>addressspaceschemas</strong>, and <strong>messagingusers</strong></td>
</tr>
<tr>
<td>enmasse.io:tenant-edit</td>
<td>Specifies <strong>create</strong>, <strong>get</strong>, <strong>update</strong>, <strong>delete</strong>, <strong>list</strong>, <strong>watch</strong>, and <strong>patch</strong> permissions for <strong>addresses</strong>, <strong>addressspaces</strong>, and <strong>messagingusers</strong>, <strong>get</strong> and <strong>list</strong> permissions for <strong>addressspaceschemas</strong></td>
</tr>
</tbody>
</table>
CHAPTER 6. MONITORING AMQ ONLINE

You can monitor AMQ Online by deploying built-in monitoring tools or using your pre-existing monitoring infrastructure.

6.1. ENABLING MONITORING ON OPENSSHIFT 4

In order to monitor AMQ Online on OpenShift 4 using the existing monitoring stack, user-workload monitoring must be enabled.

6.2. (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:

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

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

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

3. Create the enmasse-monitoring namespace:

   ```bash
   oc new-project enmasse-monitoring
   ```

4. Deploy the monitoring-operator resources:

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

5. Deploy the monitoring-operator component:

   ```bash
   oc apply -f install/components/monitoring-deployment
   ```

6.3. (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:

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

2. Select the amq-online-infra project:

   ```bash
   ```
3. Deploy the `kube-state-metrics` component:

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

6.4. ENABLING MONITORING

If you are not using a default installation configuration, the simplest way to deploy monitoring is to enable the monitoring environment variable on the `enmasse-operator` deployment.

**Prerequisites**

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

**Procedure**

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

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

2. Enable monitoring on the operator:

```bash
oc set env deployment -n amq-online-infra enmasse-operator ENABLE_MONITORING=true
```

6.5. 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:

```yaml
apiVersion: v1
class: 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
```

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:

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

2. Create a secret containing your new configuration:

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

### 6.6. METRICS AND RULES

#### 6.6.1. Common metrics

The following components export these common metrics:

- **enmasse-operator**
- **address-space-controller**
- **standard-controller**

**enmasse_version**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Provides the current version of each component in AMQ Online using the version label. The metric always returns a value of 1.</td>
</tr>
</tbody>
</table>

**Example**

```
enmasse_version{job="address-space-controller",version="1.0.1"} 1
enmasse_version{job="enmasse-operator",version="1.0.1"} 1
enmasse_version{job="standard-controller",version="1.0.1"} 1
```
6.6.2. Address space controller metrics

The following metrics for address-space-controller are available for AMQ Online.

6.6.2.1. Summary

For every metric exported of the type enmasse_address_space_status_ready there is a corresponding metric of type enmasse_address_space_status_not_ready. The values of each can never be the same.

For example:

```
enmasse_address_space_status_ready{name="my-address-space"} 1
enmasse_address_space_status_not_ready{name="my-address-space"} 0
```

The total number of address spaces equals the sum of all address spaces in the ready state plus the sum of all address spaces in the not ready state:

```
enmasse_address_spaces_total == (sum(enmasse_address_space_status_ready) + sum(enmasse_address_space_status_not_ready))
```

**enmasse_address_space_status_ready**

*Type*  
Boolean

*Description*  
Indicates each address space that is in a ready state.

*Example*

```
enmasse_address_space_status_ready{name="prod-space"} 1
enmasse_address_space_status_ready{name="dev-space"} 0
```

**enmasse_address_space_status_not_ready**

*Type*  
Boolean

*Description*  
Indicates each address space that is in a not ready state.

*Example*

```
enmasse_address_space_status_not_ready{name="prod-space"} 0
enmasse_address_space_status_not_ready{name="dev-space"} 1
```

**enmasse_address_spaces_total**

*Type*  
Gauge

*Description*  
Returns the total number of address spaces, regardless of whether they are in a ready or not ready state.
Example
enmasse_address_spaces_total 1

enmasse_address_space_connectors_total
Type
Gauge
Description
Returns the total number of address space connectors in each address space.
Example

enmasse_address_space_connectors_total{name="space-one"} 0
enmasse_address_space_connectors_total{name="space-two"} 2

6.6.3. Standard controller and agent metrics

The following standard-controller and agent metrics are available for Brokered address spaces only in AMQ Online.

6.6.3.1. Summary

The total number of addresses equals the sum of the total number of addresses in the ready state and the total number of addresses in the not ready state:

enmasse_addresses_total == enmasse_addresses_ready_total + enmasse_addresses_not_ready_total

The total number of addresses equals the total number of addresses in all phases:

enmasse_addresses_total == enmasse_addresses_active_total + enmasse_addresses_configuring_total + enmasse_addresses_failed_total + enmasse_addresses_pending_total + enmasse_addresses_terminating_total

enmasse_addresses_total
Description
Provides the total number of addresses, per address space, regardless of state.
Type
Gauge
Example

enmasse_addresses_total{addressspace="space-one"} 5
enmasse_addresses_total{addressspace="space-two"} 3

enmasse_addresses_ready_total
Type
Gauge
Description
Provides the total number of addresses currently in the ready state.

Example

```
enmasse_addresses_ready_total{addressspace="space-one"} 3
enmasse_addresses_ready_total{addressspace="space-two"} 2
```

**enmasse_addresses_not_ready_total**

**Type**

Gauge

**Description**

Provides the total number of addresses currently in the not ready state.

Example

```
enmasse_addresses_not_ready_total{addressspace="space-one"} 2
enmasse_addresses_not_ready_total{addressspace="space-two"} 1
```

**enmasse_addresses_active_total**

**Type**

Gauge

**Description**

Provides the total number of addresses currently in the active phase.

Example

```
enmasse_addresses_active_total{addressspace="space-one"} 2
```

**enmasse_addresses_configuring_total**

**Type**

Gauge

**Description**

Provides the total number of addresses currently in the configuring phase.

Example

```
enmasse_addresses_configuring_total{addressspace="space-one"} 2
```

**enmasse_addresses_failed_total**

**Type**

Gauge

**Description**

Provides the total number of addresses currently in the failed phase.

Example

```
enmasse_addresses_failed_total{addressspace="space-one"} 2
```

**enmasse_addresses_pending_total**

**Type**

Gauge
Description
Provides the total number of addresses currently in the pending phase.

Example
\texttt{enmasse_addresses_pending_total\{addressspace=\"space-one\"\} 2}

\texttt{enmasse_addresses_terminating_total}

Type
Gauge

Description
Provides the total number of addresses currently in the terminating phase.

Example
\texttt{enmasse_addresses_terminating_total\{addressspace=\"space-one\"\} 2}

\texttt{enmasse_standard_controller_loop_duration_seconds}

Type
Gauge

Description
Provides the execution time, in seconds, for the most recent standard controller reconcile loop.

Example
\texttt{enmasse_standard_controller_loop_duration_seconds 0.33}

\texttt{enmasse_standard_controller_router_check_failures_total}

Type
Counter

Description
Provides the total number of router check failures during reconciliation loop.

Example
\texttt{enmasse_standard_controller_router_check_failures_total\{addressspace=\"firstspace\"\} 0}
\texttt{enmasse_standard_controller_router_check_failures_total\{addressspace=\"myspace\"\} 0}

\texttt{enmasse_addresses_forwarders_ready_total}

Type
Gauge

Description
Provides the total number of address forwarders in the ready state.

Example
\texttt{enmasse_addresses_forwarders_ready_total\{addressspace=\"myspace\"\} 2}

\texttt{enmasse_addresses_forwarders_not_ready_total}

Type
Gauge

Description
Provides the total number of address forwarders in the not ready state.

Example

\[
\text{enmasse_addresses_forwarders_not_ready_total\{addressspace=\"myspace\"\} 0}
\]

**enmasse_addresses_forwarders_total**

*Type*  
Gauge

*Description*
Provides the total number of address forwarders, regardless of whether they are in a ready or not ready state.

Example

\[
\text{enmasse_addresses_forwarders_total\{addressspace=\"myspace\"\} 2}
\]

**enmasse_address_canary_health_failures_total**

*Type*  
Gauge

*Description*
Total number of health check failures due to failure to send and receive messages to probe addresses.

Example

\[
\text{enmasse_address_canary_health_failures_total\{addressspace=\"myspace\"\} 2}
\]

**enmasse_address_canary_health_check_failures_total**

*Type*  
Gauge

*Description*
Total number of attempted health check runs that failed due to controller errors.

Example

\[
\text{enmasse_address_canary_health_check_failures_total\{addressspace=\"myspace\"\} 1}
\]

### 6.6.4. Rules

This section details Prometheus rules installed using the PrometheusRule CRD with AMQ Online. Two types of Prometheus rules are available in AMQ Online:

- **Record**: Pre-computed expressions saved as a new set of time series.
- **Alert**: Expressions that trigger an alert when evaluated as true.

#### 6.6.4.1. Records

Records are a type of Prometheus rule that are pre-computed expressions saved as a new set of time series. The following records are available for AMQ Online.

**enmasse_address_spaces_ready_total**

*Description*
Aggregates the `enmasse_address_space_status_ready` in a single gauge-type metric that provides the total number of addresses in a **ready** state.

**Expression**

```
sum by(service, exported_namespace) (enmasse_address_space_status_ready)
```

**Example**

```
enmasse_address_spaces_ready_total{exported_namespace="prod_namespace",service="address-space-controller"} 1
```

### `enmasse_address_spaces_not_ready_total`

**Description**

Aggregates the `enmasse_address_space_status_not_ready` in a single gauge-type metric that provides the total number of addresses in a **not ready** state.

**Expression**

```
sum by(service, exported_namespace) (enmasse_address_space_status_not_ready)
```

**Example**

```
enmasse_address_spaces_not_ready_total{exported_namespace="prod_namespace",service="address-space-controller"} 1
```

### `enmasse_component_health`

**Description**

Provides a Boolean-style metric for each `address-space-controller` and `api-server` indicating if they are up and running.

**Expression**

```
up{job="address-space-controller"} or on(namespace) (1 - absent(up{job="address-space-controller"}))
up{job="api-server"} or on(namespace) (1 - absent(up{job="api-server"}))
```

**Example**

```
enmasse_component_health{job="address-space-controller"} 1
enmasse_component_health{job="api-server"} 1
```

#### 6.6.4.2. Alerts

Alerts are a type of Prometheus rule that are expressions that trigger an alert when evaluated as true. The following alerts are available for AMQ Online.

**ComponentHealth**
Description
Triggers when a component is not in a healthy state.

Expression
\[ \text{component\_health} = 0 \]

AddressSpaceHealth
Description
Triggers when one or more address spaces are not in a \textit{ready} state.

Expression
\[ \text{enmasse\_address\_spaces\_not\_ready\_total} > 0 \]

AddressHealth
Description
Triggers when one or more addresses are not in a \textit{ready} state.

Expressions
\[ \text{enmasse\_addresses\_not\_ready\_total} > 0 \]

6.7. ENABLING TENANT METRICS

Metrics from brokers and routers can be exposed to tenants without exposing system-admin metrics. To expose tenant metrics create a service monitor in any non-\texttt{amq-online-infra} namespace, ideally the namespace of the concerned address space(s).

Prerequisites
- The \texttt{servicemonitor} Custom Resource Definition provided by the Prometheus Operator must be installed.
- The tenant must have their own monitoring stack installed.

Procedure
- Create a \texttt{servicemonitor} resource with a the selector configured to match labels of \texttt{monitoring-key: enmasse-tenants} and the \texttt{amq-online-infra} as the namespace selector. An example service monitor is shown below:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: enmasse-tenants
  labels:
    app: enmasse
spec:
  selector:
    matchLabels:
      monitoring-key: enmasse-tenants
  endpoints:
  - port: health
```

namespaceSelector:
  matchNames:
  - amq-online-infra

- Ensure the tenant’s monitoring stack has read permissions for service monitors in the service monitor’s namespace but not in the amq-online-infra as this would expose service-admin metrics too.

### 6.8. USING QDSTAT

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

#### 6.8.1. 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:

   ```bash
   oc get pods
   ```

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

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

   **Output**

   ```
   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
   ```

#### 6.8.2. 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:

   ```bash
   oc get pods
   ```

2. Run the following command:

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

   **Output**

   ```
   Connections id host container role dir security
   ============================================================================
   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
   ```
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:

   ```bash
   oc get pods
   ```

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

   ```bash
   oc exec -n namespace -i qdrouterd-podname -- qdstat -b 127.0.0.1:7777 -l
   ```
6.8.4. 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:

   ```bash
   oc get pods
   ```

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

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

   **Link Routes**

<table>
<thead>
<tr>
<th>address</th>
<th>dir</th>
<th>distrib</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>$lwt</td>
<td>in</td>
<td>linkBalanced</td>
<td>inactive</td>
</tr>
<tr>
<td>$lwt</td>
<td>out</td>
<td>linkBalanced</td>
<td>inactive</td>
</tr>
</tbody>
</table>
CHAPTER 7. OPERATION PROCEDUREs FOR AMq ONliNE

7.1. RESTARTING COMPONENTS TO ACQUIRE SECURITY FIXES

Restarting AMQ Online components is required to get image updates for CVEs. The scripts are provided in the AMQ Online installation files within the script folder. To restart all components, run all scripts.

7.1.1. Restarting Operators

Operators can be restarted without affecting the messaging system.

Procedure

- Run the restart-operators.sh script:
  
  `./scripts/restart-operators.sh amq-online-infra`

7.1.2. Restarting authentication services

Authentication service restarts will temporarily affect new messaging connections. Existing connections will continue to work even if the authentication service is restarted.

Procedure

- Run the restart-authservices.sh script:
  
  `./scripts/restart-authservices.sh amq-online-infra`

7.1.3. Restarting routers

Messaging routers are only deployed in the standard address space type. The script assumes that at least two replicas of the router are running and performs a rolling restart. Messaging clients connected to the restarting router are disconnected and must reconnect to be served by a different router.

Procedure

- Run the restart-routers.sh script, which requires at least one router to be available:
  
  `./scripts/restart-routers.sh amq-online-infra 1`

7.1.4. Restarting brokers

For the brokered address space type, restarting the broker causes downtime temporarily to messaging clients while the broker is restarted. For the standard address space type, messaging clients are not disconnected from the messaging routers, but clients are not able to consume messages stored on the restarting broker.

Procedure

- Run the restart-brokers.sh script:
7.2. VIEWING ROUTER LOGS

For the standard address space type, you can view the router logs to troubleshoot issues with clients not connecting or issues with sending and receiving messages.

Procedure

1. List all router Pods and choose the Pod for the relevant address space:

   ```
   oc get pods -l name=qdrouterd -o go-template --template '{{range .items}}{{.metadata.name}}{{"\t"}}{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
   ```

2. Display the logs for the Pod:

   ```
   oc logs pod -c router
   ```

7.3. VIEWING BROKER LOGS

For the brokered or standard address space type, you can view the broker logs to troubleshoot issues with clients not connecting or issues with sending and receiving messages.

Procedure

1. List all broker Pods and choose the Pod for the relevant address space:

   ```
   oc get pods -l role=broker -o go-template --template '{{range .items}}{{.metadata.name}}{{"\t"}}{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
   ```

2. Display the logs for the Pod:

   ```
   oc logs pod
   ```

7.4. ENABLING AN AMQP PROTOCOL TRACE FOR THE ROUTER

For diagnostic purposes, you can enable an AMQP protocol trace for a router. This can be helpful when troubleshooting issues related to client connectivity or with sending and receiving messages. There are two methods for enabling a protocol trace for the router.

- You can dynamically enable/disable the protocol trace for a single router using a qdmange command. This method avoids the need to restart the router. The setting will be lost the next time the router restarts.

- Alternatively, you can apply configuration to the standardinfraconfig that enables the protocol trace for all routers of all address spaces using that standardinfraconfig. This method will cause all the routers to restart.
WARNING

Enabling the protocol trace increases the CPU overhead of the router(s) and may decrease messaging performance. It may also increase the disk space requirements associated with any log retention system. Therefore, it is recommended that you enable the protocol trace for as short a time as possible.

7.4.1. Dynamically enabling the protocol trace for a single router

Procedure

1. Log in as a service operator:
   
   \[ \text{oc login -u developer} \]

2. Change to the project where AMQ Online is installed:
   
   \[ \text{oc project amq-online-infra} \]

3. List all router Pods and choose the Pod for the relevant address space:
   
   \[ \text{oc get pods -l name=qdrouterd -o go-template --template '{{range .items}}{{.metadata.name}}{{"\t"}}{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'} \]

4. Enable the protocol trace for a single router:
   
   \[ \text{echo \{'"enable":"trace+"\'} | oc exec qdrouterd-podname --stdin=true --tty=false -- qdmanage update -b 127.0.0.1:7777 --type=log --name=log/PROTOCOL --stdin} \]

5. Display the logs for the Pod that will include the protocol trace:
   
   \[ \text{oc logs pod} \]

6. Disable the protocol trace:
   
   \[ \text{echo \{'"enable":"info\'} | oc exec qdrouterd-podname --stdin=true --tty=false -- qdmanage update -b 127.0.0.1:7777 --type=log --name=log/PROTOCOL --stdin} \]

7.4.2. Enabling the protocol trace using the StandardInfraConfig environment variable

Procedure

1. Log in as a service operator:
   
   \[ \text{oc login -u developer} \]

2. Change to the project where AMQ Online is installed:
3. Determine the **addressspaceplan** name for the address space concerned:

   ```
   oc get addressspace -n namespace address-space-name --output "jsonpath={.spec.plan}"
   ```

4. Determine the **standardinfraconfig** name for the **addressspaceplan** name:

   ```
   oc get addressspaceplan address-space-plan --output "jsonpath={.spec.infraConfigRef}"
   ```

5. Enable the protocol trace for all routers of all address spaces using that **standardinfraconfig**:

   ```
   oc patch standardinfraconfig standardinfraconfig-name --type=merge -p '{"spec":{"router":{"podTemplate":{"spec":{"containers":[{"env":[{"name":"PN_TRACE_FRM","value":"true"},{"name":"router"}]}}}}}}'
   ```

6. Display the logs for the Pod that will include the protocol trace:

   ```
   oc logs pod
   ```

7. Disable the protocol trace:

   ```
   oc patch standardinfraconfig standardinfraconfig-name --type=merge -p '{"spec":{"router":{"podTemplate":{"spec":{"containers":[{"env":[{"name":"PN_TRACE_FRM"}]}}}}}}'
   ```

---

### 7.5. ENABLING AN AMQP PROTOCOL TRACE FOR THE BROKER

For diagnostic purposes, you can enable an AMQP protocol trace for a broker. This can be helpful when troubleshooting issues with sending or receiving messages.

To enable the protocol trace, you apply configuration to the **standardinfraconfig** (for standard address spaces) or **brokeredinfraconfig** (for brokered address spaces) that enables the protocol trace for all brokers of all address spaces using that configuration. Applying this configuration will cause the brokers to restart.

**WARNING**

Enabling the protocol trace increases the CPU overhead of the broker(s) and may decrease messaging performance. It may also increase the disk space requirements associated with any log retention system. Therefore, it is recommended that you enable the protocol trace for as short a time as possible.

**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. Determine the `addressspaceplan` name for the address space concerned:

```bash
oc get addressspace -n namespace address-space-name --output "jsonpath={.spec.plan} {"n"}"
```

4. Determine the `standardinfraconfig` or `brokeredinfraconfig` name for the `addressspaceplan` name:

```bash
oc get addressspaceplan address-space-plan --output "jsonpath={.spec.infraConfigRef} {"n"}"
```

5. Enable the protocol trace for all brokers of all address spaces using that `standardinfraconfig` or `brokeredinfraconfig`:

```bash
oc patch infraconfig-resource infraconfig-name --type=merge -p '{"spec":{"broker": {"podTemplate":{"spec":{"containers":[{"env": [{"name":"PN_TRACE_FRM","value":"true"}],"name":"broker"}]}}}}}'}
```

6. Display the logs for the Pod that will include the protocol trace:

```bash
oc logs pod
```

7. Disable the protocol trace:

```bash
oc patch infraconfig-resource infraconfig-name --type=merge -p '{"spec":{"broker": {"podTemplate":{"spec":{"containers":[{"env": [{"name":"PN_TRACE_FRM"}],"name":"broker"}]}}}}}'}
```

### 7.6. EXAMINING THE STATE OF A BROKER USING THE AMQ BROKER MANAGEMENT INTERFACES

If a problem is suspected with a Broker associated with an address space, you can examine the state of the broker directly using its built-in management interfaces. AMQ Online exposes the AMQ Broker’s CLI and JMX (via Jolokia). It does not expose the AMQ Broker Console.

**Procedure**

1. Log in as a service admin:

   ```bash
   oc login -u admin
   ```

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

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

3. Retrieve the uuid for the address space:
4. Retrieve the broker support credentials (username and password) for the address space:

   oc get secret broker-support-<uuid> --template='{{.data.username}}' | base64 --decode
   oc get secret broker-support-<uuid> --template='{{.data.password}}' | base64 --decode

5. Identify the broker pod name:

   oc get pods -l infraUuid=<uuid>,role=broker

   In the standard address, there may be many brokers. To identify the broker(s) hosting a particular queue, use this command:

   oc get address address-resource-name -o jsonpath="{.status.brokerStatuses[*].containerId}"

6. Execute support commands on the broker’s pod:

   To execute an AMQ Broker CLI command, use a command similar to the following:

   oc exec broker-pod-name -- /opt/amq/bin/artemis address show --user username --password password

   To execute an AMQ Broker Jolokia JMX command, use a command similar to the following:

   'https://localhost:8161/console/jolokia/read/org.apache.activemq.artemis:broker="broker pod name"/AddressMemoryUsage'

   IMPORTANT

   The double quotes around the broker pod name within the URL are required. Make sure you protect them from your command shell using single quotes surrounding the whole URL, as shown in the above command. If they are not present, you will receive an authorization failure.

7.7. CHANGING THE BROKER LOGGING LEVEL AT RUNTIME

For diagnostic purposes, you can change the broker logging level at runtime, which can be helpful when troubleshooting issues with sending or receiving messages.

To do this, use the oc rsh command to connect to the broker Pod and adjust the logging.properties file used by the broker. The broker automatically reloads this file and immediately applies the changes without interrupting established messaging connections.

The changes made to logging.properties are transient. The system reverts the logging level to default levels the next time the broker Pod restarts.
**WARNING**

Increasing the logging verbosity increases the CPU overhead of the broker(s) and might decrease messaging performance. It might also increase the disk space requirements associated with any log retention system. Therefore, it is recommended that you increase the logging verbosity for as short a time as possible.

### Procedure

1. Log in as a service operator:
   
   ```bash
   oc login -u developer
   ```

2. Change to the project where AMQ Online is installed:
   
   ```bash
   oc project amq-online-infra
   ```

3. List all broker Pods and choose the Pod for the relevant address space:
   
   ```bash
   oc get pods -l role=broker -o go-template --template '{{range .items}}{{.metadata.name}}
   
   {{"\t"}}{{.metadata.annotations.addressSpace}}{{"\n"}}{{end}}'
   ```

4. To change the logging level for a single broker, use the `oc rsh` command to connect to the broker Pod and edit the `logging.properties` file. The broker uses the JBoss Logging framework. Adjust the logging levels of the packages corresponding to the area of interest. There are comments in the file to guide you.
   
   ```bash
   oc rsh pod
   cp /var/run/artemis/split-1/broker/etc/logging.properties /tmp/logging.properties
   vi /var/run/artemis/split-1/broker/etc/logging.properties
   exit
   ```

5. Display the logs for the Pod that will include the protocol trace:
   
   ```bash
   oc logs pod
   ```

6. To revert to normal logging levels, revert the contents of the `logging.properties` file.
   
   ```bash
   oc rsh pod
   cp /tmp/logging.properties /var/run/artemis/split-1/broker/etc/logging.properties
   exit
   ```
CHAPTER 8. AMQ ONLINE CONFIGURATION SIZING GUIDELINES

The following information provides guidelines on how to size AMQ Online installations. More specifically, these guidelines offer specific configuration recommendations for components and plans based on use cases, and the trade-offs involved when adjusting the configuration settings. Sizing AMQ Online involves configuration of:

- Brokers
- Routers (standard address space only)
- Operator(s)
- Plans

For example, each address space type has certain distinct features that need to be considered when creating the address plans.

For more information about address space types and their semantics, see [address spaces](#).

**NOTE**

Properly sizing AMQ Online components also requires taking into consideration the following points regarding your OpenShift cluster:

- The OpenShift cluster must have sufficient capacity to handle the requested resources. If the OpenShift nodes are configured with 4 GB of memory, you cannot configure brokers and routers with memory sizes larger than 4 GB.

- Since each address space creates a dedicated piece of infrastructure, you need to ensure that cluster capacity can meet demand as the number of address spaces increases.

- The use of affinity and tolerations might also restrict the nodes available for the messaging infrastructure to use.

### 8.1. BROKER COMPONENT SIZING

Brokers are configured using the `BrokeredInfraConfig` and `StandardInfraConfig` resources, depending on the type of address space. When sizing a broker, consider:

- The average message size
- The number of messages stored
- The number of queues and topics
- The address full policy

**NOTE**

In AMQ Online, you can only restrict the total amount of memory allocated for a broker. You cannot restrict the amount of memory used by individual addresses.
The broker persists all messages to disk. When the **BLOCK**, **FAIL**, or **DROP** address full policy is specified, the number of messages that can be persisted is limited to the amount of memory in the broker. By using the **PAGE** address full policy, more messages can be stored than can be held in memory, at the expense of a potential performance degradation from reading data from disk. Therefore, paging is useful in the case of large messages or a large backlog of messages in your system.

### 8.1.1. Example use case for a broker component configuration

Given 10 queues with a maximum of 1000 messages stored per queue and an average message size of 128 kB, the amount of storage space required to store messages is:

\[
10 \text{ queues} \times 1000 \text{ messages} \times (128 + (128 \text{ kB} \times 1024)) = 1.25 \text{ GB}
\]

In addition, the broker has a fixed storage footprint of about 50 MB.

The amount of memory required for the broker depends on which address full policy is specified. If the **PAGE** policy is used, the memory requirements can be reduced since the messages are stored separately from the journal (which always needs to fit in memory). If the **FAIL**, **BLOCK**, or **DROP** policies are specified, all messages must also be held in memory, even if they are persisted.

There is also constant memory cost associated with running the broker as well as the JVM. The memory available to store message is automatically derived from the memory set in the broker configuration and is set to be half the JVM memory, which in turn is set to half of the system memory.

#### NOTE

In the **standard** address space type, multiple broker instances might be created. The sizing of these broker instances also depends on the address plan configuration and how many addresses you expect each broker to be able to handle before another broker is spawned.

### 8.1.1.1. Example broker component configuration without paging

For broker configurations not using a **PAGE** policy, take into consideration an additional 5 percent bookkeeping overhead per address should be taken into account (\(1.05 \times 1.25 = 1.35 \text{ GB}\)):

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: cfg1
spec:
  broker:
    addressFullPolicy: FAIL
    globalMaxSize: 1.35Gb
    resources:
      memory: 8Gi
      storage: 2Gi

... 
```

### 8.1.1.2. Example broker component configuration with paging

When paging is enabled, the original formula can be modified to only account for a reference to the message as well as holding 1000 in-flight messages in memory:

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: cfg1
spec:
  broker:
    addressFullPolicy: PAGE
    globalMaxSize: 1.35Gb
    resources:
      memory: 8Gi
      storage: 2Gi
    ... 
```
(1000 messages * 1000 * 128 kB) + (10 queues * 128 kB * 1024) = 123.5 MB

So, the amount of memory specified for the broker can now be reduced, as seen in this configuration example:

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: BrokedrInfraConfig
metadata:
  name: cfg1
spec:
  broker:
    addressFullPolicy: PAGE
    globalMaxSize: 124Mb
    resources:
      memory: 1Gi
      storage: 2Gi
```

### 8.1.2. Broker scaling (standard address space only)

Brokers are deployed on demand, that is, when addresses of type `queue` or `topic` are created. The number of brokers deployed is restricted by the resource limits specified in the `AddressSpacePlan` configuration. The following `AddressSpacePlan` configuration example specifies a limit of four brokers in total per address space:

```yaml
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: cfg1
spec:
  resourceLimits:
    broker: 4.0
```

In terms of capacity, multiply the memory requirements for the broker by the limit.

The number of broker instances are scaled dynamically between one and the maximum limit specified based on the `AddressPlan` used for the different addresses. An `AddressPlan` specifies the fraction of a broker that is required by an address. The fraction specified in the plan is multiplied by the number of addresses referencing this plan, and then rounded up to produce the number of desired broker replicas.

**AddressPlan configuration example**

```yaml
apiVersion: admin.enmasse.io/v1beta2
kind: AddressPlan
metadata:
  name: plan1
spec:
  resources:
    broker: 0.01
```

If you create 110 addresses with `plan1` as the address plan, the number of broker replicas is \( \text{ceil}(110 \text{ addresses} \times 0.01 \text{ broker}) = 2 \text{ replicas} \).
The total number of brokers is capped by the address space plan resource limits.

8.2. ROUTER COMPONENT SIZING

Routers are configured in the **StandardInfraConfig** resource. In determining router sizing, consider:

- The number of addresses
- The number of connections and links
- Link capacity

The router does not persist any state and therefore does not require persistent storage.

Address configuration itself does not require a significant amount of router memory. However, queues and subscriptions require an additional two links between the router and broker per address.

The total number of links is then two times the number of queues/subscriptions plus the number of client links. Each link requires metadata and buffers in the router to handle routing messages for that link.

The router link capacity affects how many messages the router can handle per link. Setting the link capacity to a higher value might improve performance, but at the cost of potentially more memory being used to hold in-flight messages if senders are filling the links. If you have many connections and links, consider specifying a lower value to balance the memory usage.

In addition, the router has to parse the message headers, manage dispositions and settlements of messages, and other per-link activities. The per-link cost can be derived using a constant factor of the link capacity and message size. This factor varies depending on the message size. The following table provides an approximation of this factor for different message size ranges:

**Table 8.1. Link multiplication factor**

<table>
<thead>
<tr>
<th>Message size (bytes)</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-1000</td>
<td>18,000</td>
</tr>
<tr>
<td>1000-4000</td>
<td>22,000</td>
</tr>
<tr>
<td>4000-10,000</td>
<td>30,000</td>
</tr>
<tr>
<td>&gt;10,000</td>
<td>50,000</td>
</tr>
</tbody>
</table>

8.2.1. Example use case for router component sizing

Consider the following example use case:

- 500 anycast and 1000 queued addresses
- 10,000 connected clients (one link per client)
- Link capacity of 10
- An average message size of 512 bytes
Based on measurements, an estimated 7 kB overhead per anycast address is realistic, so:

\[
\text{500 anycast addresses} \times 7 \text{ kB overhead per address} = 3.5 \text{ MB}
\]

Memory usage of queues and topics is slightly higher than that of anycast addresses, with an estimated 32 kB overhead per address. In addition, each router-broker link can have up to linkCapacity message deliveries to keep track of. Also, we need to multiply the link capacity with the multiplication factor to account for the worst-case scenario:

\[
(1000 \text{ queued addresses} \times 32,768) + (2000 \times 18,000 \text{ link multiplication factor} \times 100 \text{ links}) = 374 \text{ MB}
\]

Memory usage of client connections/links:

\[
10,000 \text{ clients} \times 10 \text{ link capacity} \times 18,000 \text{ link multiplication factor} = 1717 \text{ MB}
\]

**NOTE**

The memory usage of client connections/links can be divided by the number of router instances.

If you have N routers, the total amount of router memory required for this configuration, including a constant base memory of 50 MB, is 50 + 3.5 + (374 + 1717)/N MB.

To ensure the maximum number of connections and links is not exceeded, a router policy can be applied as well. The following configuration example shows two routers with a router policy specified:

```yaml
apiVersion: admin.enmasse.io/v1beta1
groupKind: StandardInfraConfig
metadata:
  name: cfg1
spec:
  router:
    resources:
      memory: 1100Mi
    linkCapacity: 10
    policy:
      maxConnections: 5000
      maxSessionsPerConnection: 1
      maxSendersPerConnection: 1
      maxReceiversPerConnection: 1

2.2.2. High availability (HA)

To configure routers for high availability (HA), multiply the minimum number of required router replicas by the amount of memory per router to calculate the amount of expected memory usage. Although all connections and links are distributed across all routers, if one router fails, you must plan for those connections and links to be redistributed across the remaining routers.

2.2.3. Router scaling

Routers are scaled dynamically on demand within the limits specified for minReplicas in the StandardInfraConfig resource and the resourceLimits.router specified in the AddressSpacePlan.
restrict the number of routers to a maximum number of four, but require a minimum amount of two routers for HA purposes, the following configuration is needed:

```yaml
apiVersion: admin.enmasse.io/v1beta1
description: StandardInfraConfig
metadata:
  name: cfg1
spec:
  router:
    minReplicas: 2
...
```

---

```yaml
apiVersion: admin.enmasse.io/v1beta2
description: AddressSpacePlan
metadata:
  name: plan1
spec:
  infraConfigRef: cfg1
  resourceLimits:
    router: 4
...
```

In terms of capacity, multiply the memory requirements for the router by the resource limit. The router will then scale up to the resource limits specified in the `AddressSpacePlan` for the address space.

The number of router replicas is scaled dynamically between the minimum and maximum limits based on the `AddressPlan` used for the different addresses. An `AddressPlan` describes the fraction of a router that is required by an address. The fraction defined in the plan is multiplied by the number of addresses referencing this plan, and then rounded up to produce the number of desired router replicas.

**AddressPlan configuration example:**

```yaml
apiVersion: admin.enmasse.io/v1beta2
description: AddressPlan
metadata:
  name: plan1
spec:
  ...
  resources:
    router: 0.01
...
```

If you create 110 addresses with `plan1` as the address plan, the number of router replicas is `ceil(110 addresses * 0.01 router) = 2 replicas`.

If the number of replicas exceeds the address space plan limit, the addresses exceeding the maximum number remain in the `Pending` state and an error message describing the issue is displayed in the `Address` status section.

### 8.3. OPERATOR COMPONENT SIZING

The operator component is tasked with reading all address configuration and applying these configurations to the routers and brokers. It is important to size the operator component proportionally to the number of addresses.
In the **standard** address space, the **admin** Pod contains two processes, **agent** and **standard-controller**. These processes cannot be sized individually, but the memory usage of both is proportional to the number of addresses. In the **brokered** address space, only a single **agent** process exists.

**NOTE**

The operator processes are running on either a JVM or a Node.JS VM. Sizing the amount of memory for these processes at twice the amount of memory required for the address configuration itself is recommended.

### 8.3.1. Operator component configuration example

Each address adds about 20 kB overhead to the operator process. With 1500 addresses, an additional \( 1500 \times 2 \text{ kB} = 30 \text{ MB} \) is needed for the operator process.

In addition, these processes have a base memory requirement of 256 MB. So, the total operator memory needed is \( 256 \text{ MB} + 30 \text{ MB} = 286 \text{ MB} \). This value can be configured in both the **StandardInfraConfig** and **BrokeredInfraConfig** resources:

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: StandardInfraConfig
metadata:
  name: cfg1
spec:
  admin:
    resources:
      memory: 300Mi
...
```

### 8.4. PLAN SIZING

Plans enable dynamic scaling in the **standard** address space, as shown in the broker and router sizing sections. At the cluster level, the combination of plans and infrastructure configuration settings determines the maximum number of Pods that can be deployed on the cluster. Since AMQ Online does not support limiting the number of address spaces that can be created, it is a best practice to apply a policy to limit who is allowed to create address spaces. Such policy configuration can be handled through the standard OpenShift policies.

From a capacity-planning perspective, it is useful to calculate the maximum number of Pods and the maximum amount of memory that can be consumed for a given address space. To make this calculation using a script, see **Running the check-memory calculation script**.

### 8.4.1. Running the check-memory calculation script

You can use this script to calculate the maximum number of Pods and the maximum amount of memory that can be consumed for a given address space.

In this script, memory is assumed to be specified using the **Mi** unit, while storage is assumed to be specified using the **Gi** unit. Also, all three components, **admin**, **router**, and **broker**, must have limits specified for the script to work as intended.

**Procedure**

1. Save the following script as **check-memory.sh**:
2. Run the script using the following command:

```bash
bash calculate-memory.sh standard-small
```

If all components have limits defined in the assumed units, the script outputs the total resource limits for address spaces using this plan, as in the following example:

```
Pods: 3. Memory: 1280 MB. Storage: 2 GB
```

### 8.5. ADDRESS SIZING

Per address broker memory limits are calculated from the address plan configuration. AMQ Online determines the maximum size allowed for each queue by multiplying the broker configuration `globalMaxSize` (specified in the `standardinfraconfig` or `brokeredinfraconfig`) by the address plan’s broker resource limit. The behavior when the queue reaches its memory limit is governed by the address full policy. For more information on the address full policy, see [Broker component sizing](#).

For example, if the broker’s configuration specifies `globalMaxSize = 124 MB` and the address plan configuration specifies `addressplan.spec.resources.broker = 0.2`, the maximum size allowed for each queue is 25 MB ($124 \times 0.2 = 25 \text{ MB}$).
CHAPTER 9. UNDERSTANDING AMQ ONLINE RESOURCE CONFIGURATION

9.1. ADDRESS SPACE AND ADDRESS CONCEPTS IN AMQ ONLINE

Before you begin configuring resources for AMQ Online, you must first understand the concepts of an address space and an address in AMQ Online.

9.1.1. Address space

An address space is a group of addresses that can be accessed through a single connection (per protocol). This means that clients connected to the endpoints of an address space can send messages to or receive messages from any authorized address within that address space. An address space can support multiple protocols, as defined by the address space type.

**NOTE**

You cannot modify endpoints for an existing address space.

AMQ Online has two types of address spaces:

- **Standard**
- **Brokered**

9.1.2. Address

An address is part of an address space and represents a destination for sending and receiving messages. An address has a type, which defines the semantics of sending messages to and receiving messages from that address.

The types of addresses available in AMQ Online depend on the address space type.

9.2. SERVICE CONFIGURATION RESOURCES AND DEFINITION

The service administrator configures AMQ Online by creating Custom Resources that comprise the "service configuration." This service configuration contains instances of the following Custom Resource types:

<table>
<thead>
<tr>
<th>Custom Resource type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticationService</td>
<td>Specifies an authentication service instance used to authenticate messaging clients.</td>
</tr>
<tr>
<td>AddressSpacePlan</td>
<td>Specifies 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.</td>
</tr>
<tr>
<td>Custom Resource type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>AddressPlan</strong></td>
<td>Specifies the messaging resources consumed by a particular address using this plan, such as the fraction of routers and brokers an address can use and other properties that can be specified for multiple addresses.</td>
</tr>
<tr>
<td><strong>StandardInfraConfig</strong></td>
<td>For the standard address space type, specifies the router and broker configuration such as memory limits, storage capacity, affinity, and more.</td>
</tr>
<tr>
<td><strong>BrokedInfraConfig</strong></td>
<td>For the brokereaddress space type, specifies the broker configuration such as memory limits, storage capacity, affinity, and more.</td>
</tr>
</tbody>
</table>

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

The following diagram illustrates the relationship between the different service configuration resources and how they are referenced by the messaging tenant resources.
9.3. EXAMPLE USE CASE FOR CONFIGURING AMQ ONLINE

To help illustrate how the service configuration resources can be defined to satisfy a particular use case, the requirements of Company X for using AMQ Online are outlined. This use case is referenced throughout the following documentation describing the service configuration resource types in further detail.

Company X has the following requirements:

- Ability to accommodate multiple separate teams—for example, engineering and quality assurance (QA) work teams—that use messaging independently. To meet this requirement, multiple address spaces are needed.

- Since the applications for Company X are written to use JMS APIs and make extensive use of local transactions and they use a mixture of AMQP and OpenWire clients, using the brokered address space type is required.

- For engineering work, restricting the messaging infrastructure to support storage of no more than 1000 messages of approximately 1 KB per message, with up to 10 queues and topics is required.

  For QA work, restricting the messaging infrastructure to support storage of no more than 10,000 messages of approximately 100 KB, with up to 50 queues and topics is required.
For engineering work, the ability to restrict who can connect into the address space is required.

For engineering work, the engineering team does not need to create distinct users that need to be individually authenticated.

For QA work, the QA team must be able to create users for each instance.

Each of these requirements and how they can be met by configuring the appropriate resources is discussed in the following sections.

### 9.3.1. Restricting messaging infrastructure

Company X has the following requirements for using AMQ Online:

- For engineering work, restricting the messaging infrastructure to support storage of no more than 1000 messages of approximately 1 KB per message, with up to 10 queues and topics is required.
- For QA work, restricting the messaging infrastructure to support storage of no more than 10,000 messages of approximately 100 KB, with up to 50 queues and topics is required.

Meeting this requirement involves configuring the `BrokeredInfraConfig` resource. The following points need to be taken into consideration:

- Calculate the memory size for the broker: Given the requirements, specifying a relatively small memory size for engineering work is likely sufficient, while more memory is required for the QA work. For more information about broker sizing guidelines, see [Broker component sizing](#).

- Calculate the minimum amount of storage for the broker. For more information about broker sizing guidelines, see [Broker component sizing](#).

#### 9.3.1.1. Examples of brokered infrastructure configurations

The following brokered infrastructure configuration examples show broker component resource values that meet the requirements of Company X.

**Brokered infrastructure configuration example for engineering**

```yaml
apiVersion: admin.enmasse.io/v1beta1
class: BrokeredInfraConfig
metadata:
  name: engineering
spec:
  broker:
    resources:
      memory: 512Mi
      storage: 20Mi
```

**Brokered infrastructure configuration example for QA**

```yaml
apiVersion: admin.enmasse.io/v1beta1
class: BrokeredInfraConfig
metadata:
  name: qa
spec:
  broker:
```
### 9.3.2. Ability to restrict address space connections

Company X has the following requirement for using AMQ Online: For engineering work, the ability to restrict who can connect into the address space is required.

To meet this requirement you must set a network policy in the brokered infrastructure configuration. For more information about network policies, see

- OpenShift Container Platform 3.11 documentation about Enabling Network Policy.
- OpenShift Container Platform 4.2 documentation about Configuring network policy with OpenShift SDN.

**Brokered infrastructure configuration example showing network policy setting**

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: BrokeredInfraConfig
metadata:
  name: engineering
spec:
  networkPolicy:
    ingress:
      - from:
          namespaceSelector:
            matchLabels:
              org: engineering
  broker:
    resources:
      memory: 512Mi
      storage: 20Mi
```

In addition, the address space plan references the previous `BrokeredInfraConfig` Custom Resource.

**Address space plan example**

```yaml
apiVersion: admin.enmasse.io/v1beta2
kind: AddressSpacePlan
metadata:
  name: engineering
spec:
  infraConfigRef: engineering
  addressSpaceType: brokered
  addressPlans:
    - brokered-queue
    - brokered-topic
```

### 9.3.3. Authentication service resource examples
Company X has the following requirement for using AMQ Online: For engineering work, the engineering team does not need to create distinct users that need to be individually authenticated. To meet this requirement, you specify the `none` authentication service:

**None authentication service example**

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

For QA work, the QA team must be able to create users for each instance. Also, QA has a database they want to use for persisting the users. To meet this requirement, you must use the `standard` authentication service and specify a data source:

**Standard authentication service example**

```yaml
apiVersion: admin.enmasse.io/v1beta1
kind: AuthenticationService
metadata:
  name: qa
spec:
type: standard
standard:
  storage:
    type: persistent-claim
    size: 5Gi
  datasource:
    type: postgresql
    host: db.example.com
    port: 5432
    database: authdb
```
The following table describes the AMQ Online resources that pertain to the service administrator role.

Table A.1. AMQ Online service administrator resources table

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressplans</td>
<td>Specifies the address plan.</td>
</tr>
<tr>
<td>addressspaceplans</td>
<td>Specifies the address space plan.</td>
</tr>
<tr>
<td>addressspaceschemas</td>
<td>Defines the service characteristics available to an addressspace. An addressspace refers to one addressspaceschema. standard and brokered are predefined addressspaceschemas.</td>
</tr>
<tr>
<td>brokeredinfraconfigs</td>
<td>Specifies the infrastructure configuration for brokered address spaces. For more information see Brokered infrastructure configuration fields table.</td>
</tr>
<tr>
<td>standardinfraconfigs</td>
<td>Specifies the infrastructure configuration for standard address spaces. For more information see Standard infrastructure configuration fields table.</td>
</tr>
</tbody>
</table>
This table shows the fields available for the brokered infrastructure configuration and a brief description.

**Table B.1. Brokered infrastructure configuration fields table**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>version</strong></td>
<td>Specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version.</td>
</tr>
<tr>
<td><strong>admin.resources.memory</strong></td>
<td>Specifies the amount of memory allocated to the admin Pod.</td>
</tr>
<tr>
<td><strong>admin.podTemplate.metadata.labels</strong></td>
<td>Specifies the labels added to the admin Pod.</td>
</tr>
<tr>
<td><strong>admin.podTemplate.spec.affinity</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>admin.podTemplate.spec.priorityClassName</strong></td>
<td>Specifies the priority class to use for the admin Pod so you can prioritize admin Pods over other Pods in the OpenShift cluster.</td>
</tr>
<tr>
<td><strong>admin.podTemplate.spec.tolerations</strong></td>
<td>Specifies the toleration settings for the admin Pod, which allows this Pod to run on certain nodes that other Pods cannot run on.</td>
</tr>
<tr>
<td><strong>broker.addressFullPolicy</strong></td>
<td>Specifies the action taken when a queue is full: <strong>BLOCK, FAIL, PAGE, DROP</strong>. The default value is <strong>PAGE</strong>. For more information see the AMQ Broker documentation.</td>
</tr>
<tr>
<td><strong>broker.globalMaxSize</strong></td>
<td>Specifies the maximum amount of memory used for queues in the broker.</td>
</tr>
<tr>
<td><strong>broker.resources.memory</strong></td>
<td>Specifies the amount of memory allocated to the broker.</td>
</tr>
<tr>
<td><strong>broker.resources.storage</strong></td>
<td>Specifies the amount of storage requested for the broker.</td>
</tr>
<tr>
<td><strong>broker.podTemplate.metadata.labels</strong></td>
<td>Specifies the labels added to the broker Pod.</td>
</tr>
<tr>
<td><strong>broker.podTemplate.spec.affinity</strong></td>
<td>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.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.priorityClassName</code></td>
<td>Specifies the priority class to use for the broker Pod so you can prioritize broker Pods over other Pods in the OpenShift cluster.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.tolerations</code></td>
<td>Specifies the toleration settings for the broker Pod, which allows this Pod to run on certain nodes that other Pods cannot run on.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.securityContext</code></td>
<td>Specifies the security context for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.env</code></td>
<td>Specifies environment variables for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.livenessProbe.failureThreshold</code></td>
<td>Specifies the number of times that OpenShift tries when a broker Pod starts and the probe fails before restarting the container.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.livenessProbe.initialDelaySeconds</code></td>
<td>Specifies the probe delay value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.livenessProbe.timeoutSeconds</code></td>
<td>Specifies the probe timeout value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.readinessProbe.failureThreshold</code></td>
<td>Specifies the number of times that OpenShift tries when a broker Pod starts and the probe fails before the Pod is marked <strong>Unready</strong>.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.readinessProbe.initialDelaySeconds</code></td>
<td>Specifies the probe delay value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.readinessProbe.timeoutSeconds</code></td>
<td>Specifies the probe timeout value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.resources</code></td>
<td>Specifies broker Pod resource requests and limits for CPU and memory.</td>
</tr>
<tr>
<td><code>broker.storageClassName</code></td>
<td>Specifies what storage class to use for the persistent volume for the broker.</td>
</tr>
<tr>
<td><code>broker.updatePersistentVolumeClaim</code></td>
<td>If the persistent volume supports resizing, setting this value to <strong>true</strong> allows the broker storage to be resized.</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Specifies the AMQ Online version used. When upgrading, AMQ Online uses this field to determine whether to upgrade the infrastructure to the requested version.</td>
</tr>
<tr>
<td>admin.resources.memory</td>
<td>Specifies the amount of memory allocated to the admin Pod.</td>
</tr>
<tr>
<td>admin.podTemplate.metadata.labels</td>
<td>Specifies the labels added to the admin Pod.</td>
</tr>
<tr>
<td>admin.podTemplate.spec.affinity</td>
<td>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.</td>
</tr>
<tr>
<td>admin.podTemplate.spec.priorityClassName</td>
<td>Specifies the priority class to use for the admin Pod so you can prioritize admin pods over other Pods in the OpenShift cluster.</td>
</tr>
<tr>
<td>admin.podTemplate.spec.tolerations</td>
<td>Specifies the toleration settings for the admin Pod, which allow this Pod to run on certain nodes on which other Pods cannot run.</td>
</tr>
<tr>
<td>broker.addressFullPolicy</td>
<td>Specifies the action taken when a queue is full: BLOCK, FAIL, PAGE, DROP. The default value is PAGE. For more information see the AMQ Broker documentation.</td>
</tr>
<tr>
<td>broker.globalMaxSize</td>
<td>Specifies the maximum amount of memory used for queues in the broker.</td>
</tr>
<tr>
<td>broker.resources.memory</td>
<td>Specifies the amount of memory allocated to the broker.</td>
</tr>
<tr>
<td>broker.resources.storage</td>
<td>Specifies the amount of storage requested for the broker.</td>
</tr>
<tr>
<td>broker.podTemplate.metadata.labels</td>
<td>Specifies the labels added to the broker Pod.</td>
</tr>
<tr>
<td>broker.podTemplate.spec.affinity</td>
<td>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.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.priorityClassName</code></td>
<td>Specifies the priority class to use for the broker Pod so you can prioritize broker Pods over other Pods in the OpenShift cluster.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.tolerations</code></td>
<td>Specifies the toleration settings for the broker Pod, which allow this Pod to run on certain nodes on which other Pods cannot run.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.securityContext</code></td>
<td>Specifies the security context for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.env</code></td>
<td>Specifies environment variables for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.livenessProbe.failureThreshold</code></td>
<td>Specifies the number of times that OpenShift tries when a broker Pod starts and the probe fails before restarting the container.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.livenessProbe.initialDelaySeconds</code></td>
<td>Specifies the probe delay value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.livenessProbe.timeoutSeconds</code></td>
<td>Specifies the probe timeout value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.readinessProbe.failureThreshold</code></td>
<td>Specifies the number of times that OpenShift tries when a broker Pod starts and the probe fails before the Pod is marked Unready.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.readinessProbe.initialDelaySeconds</code></td>
<td>Specifies the probe delay value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.readinessProbe.timeoutSeconds</code></td>
<td>Specifies the probe timeout value in seconds for the broker Pod.</td>
</tr>
<tr>
<td><code>broker.podTemplate.spec.containers.resources</code></td>
<td>Specifies broker Pod resource requests and limits for CPU and memory.</td>
</tr>
<tr>
<td><code>broker.connectorIdleTimeout</code></td>
<td>Specifies the AMQP idle timeout to use for connection to router.</td>
</tr>
<tr>
<td><code>broker.connectorWorkerThreads</code></td>
<td>Specifies the number of worker threads of the connection to the router.</td>
</tr>
<tr>
<td><code>broker.storageClassName</code></td>
<td>Specifies what storage class to use for the persistent volume for the broker.</td>
</tr>
<tr>
<td><code>broker.updatePersistentVolumeClaim</code></td>
<td>If the persistent volume supports resizing, setting this value to <code>true</code> allows the broker storage to be resized.</td>
</tr>
<tr>
<td><code>broker.treatRejectAsUnmodifiedDeliveryFailed</code></td>
<td>Treat rejected delivery outcome as modified delivery failed. This causes the message to be re-sent to the consumer by default. The default value is <code>true</code>.</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>broker.useModifiedForTransientDeliveryErrors</td>
<td>Respond with modified for transient delivery errors to allow sender to retry. The default value is true.</td>
</tr>
<tr>
<td>broker.minLargeMessageSize</td>
<td>Specifies the minimum size of a message for it to be treated as a large message. A large message is always paged to disk with a reference in the journal. The default value is -1 (never page).</td>
</tr>
<tr>
<td>router.resources.memory</td>
<td>Specifies the amount of memory allocated to the router.</td>
</tr>
<tr>
<td>router.linkCapacity</td>
<td>Specifies the default number of credits issued on AMQP links for the router.</td>
</tr>
<tr>
<td>router.handshakeTimeout</td>
<td>Specifies the amount of time in seconds to wait for the secure handshake to be initiated.</td>
</tr>
<tr>
<td>router.minReplicas</td>
<td>Specifies the minimum number of router Pods to run; a minimum of two are required for high availability (HA) configuration.</td>
</tr>
<tr>
<td>router.podTemplate.metadata.labels</td>
<td>Specifies the labels added to the router Pod.</td>
</tr>
<tr>
<td>router.podTemplate.spec.affinity</td>
<td>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.</td>
</tr>
<tr>
<td>router.podTemplate.spec.priorityClassName</td>
<td>Specifies the priority class to use for the router Pod so you can prioritize router pods over other pods in the OpenShift cluster.</td>
</tr>
<tr>
<td>router.podTemplate.spec.tolerations</td>
<td>Specifies the toleration settings for the router Pod, which allow this Pod to run on certain nodes on which other Pods cannot run.</td>
</tr>
<tr>
<td>broker.podTemplate.spec.securityContext</td>
<td>Specifies the security context for the router Pod.</td>
</tr>
<tr>
<td>router.podTemplate.spec.containers.env</td>
<td>Specifies the environment variables for the router Pod.</td>
</tr>
<tr>
<td>router.podTemplate.spec.containers.livenessProbe.failureThreshold</td>
<td>Specifies the number of times that OpenShift tries when a router Pod starts and the probe fails before restarting the container.</td>
</tr>
<tr>
<td>router.podTemplate.spec.containers.livenessProbe.initialDelaySeconds</td>
<td>Specifies the probe delay value in seconds for the router Pod.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>router.podTemplate.spec.containers.livenessProbe.timeoutSeconds</code></td>
<td>Specifies the probe timeout value in seconds for the router Pod.</td>
</tr>
<tr>
<td><code>router.podTemplate.spec.containers.readinessProbe.failureThreshold</code></td>
<td>Specifies the number of times that OpenShift tries when a router Pod starts and the probe fails before the Pod is marked <strong>Unready</strong>.</td>
</tr>
<tr>
<td><code>router.podTemplate.spec.containers.readinessProbe.initialDelaySeconds</code></td>
<td>Specifies the probe delay value in seconds for the router Pod.</td>
</tr>
<tr>
<td><code>router.podTemplate.spec.containers.readinessProbe.timeoutSeconds</code></td>
<td>Specifies the probe timeout value in seconds for the router Pod.</td>
</tr>
<tr>
<td><code>router.podTemplate.spec.containers.resources</code></td>
<td>Specifies router Pod resource requests and limits for CPU and memory.</td>
</tr>
<tr>
<td><code>router.idleTimeout</code></td>
<td>Specifies the AMQP idle timeout to use for all router listeners.</td>
</tr>
<tr>
<td><code>router.workerThreads</code></td>
<td>Specifies the number of worker threads to use for the router.</td>
</tr>
<tr>
<td><code>router.policy.maxConnections</code></td>
<td>Specifies the maximum number of router connections allowed.</td>
</tr>
<tr>
<td><code>router.policy.maxConnectionsPerUser</code></td>
<td>Specifies the maximum number of router connections allowed per user.</td>
</tr>
<tr>
<td><code>router.policy.maxConnectionsPerHost</code></td>
<td>Specifies the maximum number of router connections allowed per host.</td>
</tr>
<tr>
<td><code>router.policy.maxSessionsPerConnection</code></td>
<td>Specifies the maximum number of sessions allowed per router connection.</td>
</tr>
<tr>
<td><code>router.policy.maxSendersPerConnection</code></td>
<td>Specifies the maximum number of senders allowed per router connection.</td>
</tr>
<tr>
<td><code>router.policy.maxReceiversPerConnection</code></td>
<td>Specifies the maximum number of receivers allowed per router connection.</td>
</tr>
</tbody>
</table>
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: 0.33-SNAPSHOT

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: https://enmasse.io/documentation/

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
</table>

Red Hat AMQ 2020.Q4 Installing and Managing AMQ Online on OpenShift
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>io.enmasse.admin.v1beta2.AddressSpacePlan</td>
</tr>
</tbody>
</table>

**D.1.2.1.3. Responses**

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.admin.v1beta2.AddressSpacePlan</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.admin.v1beta2.AddressSpacePlan</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

**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**

/api/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans

**D.1.2.2.1. Description**

list objects of kind AddressSpacePlan

**D.1.2.2.2. Parameters**
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>labelSelector</td>
<td>A selector to restrict the list of returned objects by their labels. Defaults to everything.</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>optional</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D.1.2.3. GET

/api/admin.enmasse.io/v1beta2/namespaces/{namespace}/addressspaceplans/{name}

D.1.2.3.1. Description

read the specified AddressSpacePlan

D.1.2.3.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of AddressSpacePlan to read.</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D.1.2.3.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.admin.v1beta2.AddressSpacePlan</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of AddressSpacePlan to replace.</td>
<td>string</td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>io.enmasse.admin.v1beta2.AddressSpacePlan</td>
</tr>
</tbody>
</table>
D.1.2.4.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.admin.v1beta2.AddressSpacePlan</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.admin.v1beta2.AddressSpacePlan</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of AddressSpacePlan to delete.</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D.1.2.5.3. Responses
<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>Status</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
</tbody>
</table>

D.1.2.6.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>labelSelector</td>
<td>A selector to restrict the list of returned objects by their labels. Defaults to everything.</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>optional</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D.1.2.7.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.AddressList</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of Address to read</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
</tbody>
</table>

D.1.2.8.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of Address to replace</td>
<td>string</td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
</tbody>
</table>

D.1.2.9.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of Address to delete</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><em>required</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><em>required</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**D.1.2.10.3. Responses**

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>Status</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of Address to replace</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><em>required</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><em>required</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>JsonPatchRequest</td>
</tr>
<tr>
<td></td>
<td><em>required</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D.1.2.11.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>io.enmasse.v1beta1.AddressSpace</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D.1.2.12.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.Address</td>
</tr>
</tbody>
</table>

Red Hat AMQ 2020.Q4 Installing and Managing AMQ Online on OpenShift
<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.v1beta1.AddressSpace</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query</td>
<td>labelSelector</td>
<td>A selector to restrict the list of returned objects by their labels. Defaults to everything.</td>
<td>string</td>
</tr>
</tbody>
</table>

D.1.2.13.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.AddressSpaceList</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>
D.1.2.13.4. Produces
- `application/json`

D.1.2.13.5. Tags
- `addressspaces`
- `enmasse_v1beta1`

D.1.2.14. GET `/apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}`

D.1.2.14.1. Description
read the specified AddressSpace

D.1.2.14.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of AddressSpace to read</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><em>required</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><em>required</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D.1.2.14.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td><code>io.enmasse.v1beta1.AddressSpace</code></td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.14.4. Consumes
- `application/json`

D.1.2.14.5. Produces
- `application/json`

D.1.2.14.6. Tags
- `addressspaces`
APPENDIX D. REST API REFERENCE

- enmasse_v1beta1

D.1.2.15. PUT /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.15.1. Description
replace the specified AddressSpace

D.1.2.15.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of AddressSpace to replace</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>io.enmasse.v1beta1.AddressSpace</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D.1.2.15.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.AddressSpace</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.v1beta1.AddressSpace</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.15.4. Produces
- application/json

D.1.2.15.5. Tags
- addressspaces
- enmasse_v1beta1

D.1.2.16. DELETE /apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.16.1. Description
delete an AddressSpace

D.1.2.16.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of AddressSpace to delete</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td>for teams and projects</td>
<td></td>
</tr>
</tbody>
</table>

D.1.2.16.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>Status</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.16.4. Produces

- application/json

D.1.2.16.5. Tags

- addressspaces
- enmasse_v1beta1

D.1.2.17. PATCH
/apis/enmasse.io/v1beta1/namespaces/{namespace}/addressspaces/{name}

D.1.2.17.1. Description

patches (RFC6902) the specified AddressSpace

D.1.2.17.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of AddressSpace to replace</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### D.1.2.17.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.v1beta1.AddressSpace</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

### D.1.2.17.4. Consumes

- application/json-patch+json

### D.1.2.17.5. Produces

- application/json

### D.1.2.17.6. Tags

- addressspaces
- enmasse_v1beta1

### D.1.2.18. POST /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

#### D.1.2.18.1. Description

create a MessagingUser

#### D.1.2.18.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>JsonPatchRequest</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D.1.2.18.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.user.v1beta1.MessagingUser</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.user.v1beta1.MessagingUser</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.18.4. Consumes

- application/json

D.1.2.18.5. Produces

- application/json

D.1.2.18.6. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.19. GET /apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers

D.1.2.19.1. Description

list objects of kind MessagingUser

D.1.2.19.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>namespace required</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td>Query</td>
<td>labelSelector optional</td>
<td>A selector to restrict the list of returned objects by their labels. Defaults to everything.</td>
<td>string</td>
</tr>
</tbody>
</table>

D.1.2.19.3. Responses
<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.user.v1beta1.MessagingUserList</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.19.4. Produces

- application/json

D.1.2.19.5. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.20. GET

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

D.1.2.20.1. Description

read the specified MessagingUser

D.1.2.20.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of MessagingUser to read. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').</td>
<td>string</td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
</tbody>
</table>

D.1.2.20.3. Responses

<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.user.v1beta1.MessagingUser</td>
</tr>
<tr>
<td>HTTP Code</td>
<td>Description</td>
<td>Schema</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.20.4. Consumes

- application/json

D.1.2.20.5. Produces

- application/json

D.1.2.20.6. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.21. PUT
/apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.21.1. Description
replace the specified MessagingUser

D.1.2.21.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of MessagingUser to replace. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').</td>
<td>string</td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>io.enmasse.user.v1beta1.MessagingUser</td>
</tr>
</tbody>
</table>

D.1.2.21.3. Responses
<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>io.enmasse.user.v1beta1.MessagingUser</td>
</tr>
<tr>
<td>201</td>
<td>Created</td>
<td>io.enmasse.user.v1beta1.MessagingUser</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.21.4. Produces

- application/json

D.1.2.21.5. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.22. DELETE
/apis/user.enmasse.io/v1beta1/namespaces/{namespace}/messagingusers/{name}

D.1.2.22.1. Description

delete a MessagingUser

D.1.2.22.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of MessagingUser to delete. Must include addressSpace and dot separator in the name (that is, 'myspace.user1').</td>
<td>string</td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
</tbody>
</table>

D.1.2.22.3. Responses
<table>
<thead>
<tr>
<th>HTTP Code</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td>Status</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not found</td>
<td>No Content</td>
</tr>
</tbody>
</table>

D.1.2.22.4. Produces

- application/json

D.1.2.22.5. Tags

- auth
- enmasse_v1beta1
- user

D.1.2.23. PATCH

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

D.1.2.23.1. Description

patches (RFC6902) the specified MessagingUser

D.1.2.23.2. Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>name</td>
<td>Name of MessagingUser to replace. Must include addressSpace and dot separator in the name (that is, 'myspace.user1')</td>
<td>string</td>
</tr>
<tr>
<td>Path</td>
<td>namespace</td>
<td>object name and auth scope, such as for teams and projects</td>
<td>string</td>
</tr>
<tr>
<td>Body</td>
<td>body</td>
<td></td>
<td>JsonPatchRequest</td>
</tr>
</tbody>
</table>

D.1.2.23.3. Responses
### D.1.2.23.4. Consumes

- `application/json-patch+json`

### D.1.2.23.5. Produces

- `application/json`

### D.1.2.23.6. Tags

- `auth`
- `enmasse_v1beta1`
- `user`

### D.1.3. Definitions

#### D.1.3.1. JsonPatchRequest

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>object</td>
</tr>
<tr>
<td>patch</td>
<td><code>&lt;Patch&gt;</code> array</td>
</tr>
</tbody>
</table>

#### D.1.3.2. ObjectMeta

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
</tbody>
</table>
# D.1.3.3. Patch

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>Required for operations copy, replace</td>
<td>string</td>
</tr>
<tr>
<td>op</td>
<td></td>
<td>enum (add, remove, replace, move, copy, test)</td>
</tr>
<tr>
<td>path</td>
<td>Slash separated format</td>
<td>string</td>
</tr>
<tr>
<td>value</td>
<td>Required for operations add, replace, test</td>
<td>string</td>
</tr>
</tbody>
</table>

## D.1.3.4. Status

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>Suggested HTTP return code for this status, 0 if not set.</td>
<td>integer (int32)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>enum (admin.enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>kind</td>
<td>enum (BrokeredInfraConfig)</td>
</tr>
<tr>
<td>metadata</td>
<td>ObjectMeta</td>
</tr>
<tr>
<td>spec</td>
<td>io.enmasse.admin.v1beta1.BrokeredInfraConfigSpec</td>
</tr>
</tbody>
</table>
### D.1.3.6. io.enmasse.admin.v1beta1.BrokeredInfraConfigList

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td><code>enum (admin.enmasse.io/v1beta1)</code></td>
</tr>
<tr>
<td>items</td>
<td><code>&lt; io.enmasse.admin.v1beta1.BrokeredInfraConfig &gt; array</code></td>
</tr>
<tr>
<td>kind</td>
<td><code>enum (BrokeredInfraConfigList)</code></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td><code>io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecAdmin</code></td>
</tr>
<tr>
<td>broker</td>
<td><code>io.enmasse.admin.v1beta1.BrokeredInfraConfigSpecBroker</code></td>
</tr>
<tr>
<td>networkPolicy</td>
<td><code>networkPolicy</code></td>
</tr>
<tr>
<td>version</td>
<td><code>string</code></td>
</tr>
</tbody>
</table>

**networkPolicy**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>egress</td>
<td><code>&lt; io.k8s.api.networking.v1.NetworkPolicyEgressRule &gt; array</code></td>
</tr>
<tr>
<td>ingress</td>
<td><code>&lt; io.k8s.api.networking.v1.NetworkPolicyIngressRule &gt; array</code></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>podTemplate</td>
<td><code>io.enmasse.admin.v1beta1.InfraConfigPodSpec</code></td>
</tr>
<tr>
<td>Name</td>
<td>Schema</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>resources</td>
<td>resources</td>
</tr>
<tr>
<td>memory</td>
<td>string</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressFullPolicy</td>
<td>enum (PAGE, BLOCK, FAIL)</td>
</tr>
<tr>
<td>podTemplate</td>
<td>io.enmasse.admin.v1beta1.InfraConfigPodSpec</td>
</tr>
<tr>
<td>resources</td>
<td>resources</td>
</tr>
<tr>
<td>storageClassName</td>
<td>string</td>
</tr>
<tr>
<td>updatePersistentVolumeClaim</td>
<td>boolean</td>
</tr>
</tbody>
</table>

resources

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory</td>
<td>string</td>
</tr>
<tr>
<td>storage</td>
<td>string</td>
</tr>
<tr>
<td>Name</td>
<td>Schema</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>metadata</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>metadata</td>
</tr>
<tr>
<td>spec</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>spec</td>
</tr>
</tbody>
</table>

metadata

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>labels</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>object</td>
</tr>
</tbody>
</table>

spec

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>affinity</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>object</td>
</tr>
<tr>
<td>containers</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>&lt; containers &gt; array</td>
</tr>
<tr>
<td>priorityClassName</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>string</td>
</tr>
<tr>
<td>securityContext</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>object</td>
</tr>
<tr>
<td>tolerations</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>&lt; object &gt; array</td>
</tr>
</tbody>
</table>

containers

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>resources</td>
<td></td>
</tr>
<tr>
<td>optional</td>
<td>object</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>enum (admin.enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>kind</td>
<td>enum (StandardInfraConfig)</td>
</tr>
<tr>
<td>metadata</td>
<td>ObjectMeta</td>
</tr>
<tr>
<td>spec</td>
<td>io.enmasse.admin.v1beta1.StandardInfraConfigSpec</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>enum (admin.enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>items</td>
<td>&lt; io.enmasse.admin.v1beta1.StandardInfraConfig &gt; array</td>
</tr>
<tr>
<td>kind</td>
<td>enum (StandardInfraConfigList)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>io.enmasse.admin.v1beta1.StandardInfraConfigSpecAdmin</td>
</tr>
<tr>
<td>broker</td>
<td>io.enmasse.admin.v1beta1.StandardInfraConfigSpecBroker</td>
</tr>
<tr>
<td>networkPolicy</td>
<td>networkPolicy</td>
</tr>
<tr>
<td>router</td>
<td>io.enmasse.admin.v1beta1.StandardInfraConfigSpecRouter</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
</tr>
</tbody>
</table>
### networkPolicy

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>egress</td>
<td>optional &lt; io.k8s.api.networking.v1.NetworkPolicyEgressRule &gt; array</td>
</tr>
<tr>
<td>ingress</td>
<td>optional &lt; io.k8s.api.networking.v1.NetworkPolicyIngressRule &gt; array</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>podTemplate</td>
<td>optional io.enmasse.admin.v1beta1.InfraConfigPodSpec</td>
</tr>
<tr>
<td>resources</td>
<td>optional resources</td>
</tr>
</tbody>
</table>

### resources

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory</td>
<td>string</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressFullPolicy</td>
<td>optional enum (PAGE, BLOCK, FAIL)</td>
</tr>
<tr>
<td>connectorIdleTimeout</td>
<td>optional integer</td>
</tr>
<tr>
<td>connectorWorkerThreads</td>
<td>optional integer</td>
</tr>
<tr>
<td>podTemplate</td>
<td>optional io.enmasse.admin.v1beta1.InfraConfigPodSpec</td>
</tr>
<tr>
<td>resources</td>
<td>optional resources</td>
</tr>
</tbody>
</table>
### Name | Schema
---|---
storageClassName  | string  
(updatePersistentVolumeClaim  | boolean

**resources**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
</table>
| memory  | string  
(storage  | string

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
</table>
| idleTimeout  | integer  
(initialHandshakeTimeout  | integer

| linkCapacity  | integer
| minAvailable  | integer
| minReplicas  | integer
| podTemplate  | io.enmasse.admin.v1beta1.InfraConfigPodSpec

| policy  | policy
| resources  | resources
### workerThreads
- **Name:** workerThreads
- **Schema:** integer

### policy
- **Name:**
  - maxConnections
    - **Schema:** integer
  - maxConnectionsPerHost
    - **Schema:** integer
  - maxConnectionsPerUser
    - **Schema:** integer
  - maxReceiversPerConnection
    - **Schema:** integer
  - maxSendersPerConnection
    - **Schema:** integer
  - maxSessionsPerConnection
    - **Schema:** integer

### resources
- **Name:**
  - memory
    - **Schema:** string

### D.1.3.17. io.enmasse.admin.v1beta2.AddressPlan
- **Name:**
  - apiVersion
    - **Schema:** enum (admin.enmasse.io/v1beta2)
  - kind
    - **Schema:** enum (AddressPlan)
  - metadata
    - **Schema:** ObjectMeta
<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>spec</strong></td>
<td>io.enmasse.admin.v1beta2.AddressPlanSpec</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>enum (admin.enmasse.io/v1beta2)</td>
</tr>
<tr>
<td>items</td>
<td>&lt; io.enmasse.admin.v1beta2.AddressPlan &gt; array</td>
</tr>
<tr>
<td>kind</td>
<td>enum (AddressPlanList)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressType</td>
<td>string</td>
</tr>
<tr>
<td>displayName</td>
<td>string</td>
</tr>
<tr>
<td>displayOrder</td>
<td>integer</td>
</tr>
<tr>
<td>longDescription</td>
<td>string</td>
</tr>
<tr>
<td>partitions</td>
<td>integer</td>
</tr>
<tr>
<td>resources</td>
<td>resources</td>
</tr>
<tr>
<td>shortDescription</td>
<td>string</td>
</tr>
</tbody>
</table>

resources
### APPENDIX D. REST API REFERENCE

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>broker</td>
<td>optional number</td>
</tr>
<tr>
<td>router</td>
<td>optional number</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>required enum (admin.enmasse.io/v1beta2)</td>
</tr>
<tr>
<td>items</td>
<td>required (&lt;\text{io.enmasse.admin.v1beta2.AddressSpacePlan})&gt; array</td>
</tr>
<tr>
<td>kind</td>
<td>required enum (AddressSpacePlanList)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressPlans</td>
<td>required (&lt;\text{string})&gt; array</td>
</tr>
<tr>
<td>addressSpaceType</td>
<td>required string</td>
</tr>
<tr>
<td>Name</td>
<td>Schema</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
</tr>
<tr>
<td>displayName</td>
<td>string</td>
</tr>
<tr>
<td>displayOrder</td>
<td>integer</td>
</tr>
<tr>
<td>infraConfigRef</td>
<td>string</td>
</tr>
<tr>
<td>longDescription</td>
<td>string</td>
</tr>
<tr>
<td>resourceLimits</td>
<td>resourceLimits</td>
</tr>
<tr>
<td>shortDescription</td>
<td>string</td>
</tr>
</tbody>
</table>

**resourceLimits**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregate</td>
<td>number</td>
</tr>
<tr>
<td>broker</td>
<td>number</td>
</tr>
<tr>
<td>router</td>
<td>number</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>enum (user.enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>kind</td>
<td>enum (MessagingUser)</td>
</tr>
<tr>
<td>metadata</td>
<td>ObjectMeta</td>
</tr>
</tbody>
</table>
## D.1.3.24. io.enmasse.user.v1beta1.MessagingUserList

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>enum (user.enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>items</td>
<td>&lt; io.enmasse.user.v1beta1.MessagingUser &gt; array</td>
</tr>
<tr>
<td>kind</td>
<td>enum (MessagingUserList)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>authentication</td>
<td>authentication</td>
</tr>
<tr>
<td>authorization</td>
<td>&lt; authorization &gt; array</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
</tr>
</tbody>
</table>

### authentication

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>federatedUserId</td>
<td>User id of the user to federate when ‘federated’ type is specified.</td>
<td>string</td>
</tr>
<tr>
<td>federatedUserName</td>
<td>User name of the user to federate when ‘federated’ type is specified.</td>
<td>string</td>
</tr>
<tr>
<td>password</td>
<td>Base64 encoded value of password when ‘password’ type is specified.</td>
<td>string</td>
</tr>
</tbody>
</table>
### provider

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>Name of provider to use for federated identity when 'federated' type is specified.</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td></td>
<td>enum (password, serviceaccount)</td>
</tr>
</tbody>
</table>

### authorization

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>&lt; string &gt; array</td>
</tr>
<tr>
<td>required</td>
<td>&lt; enum (send, recv, view, manage) &gt; array</td>
</tr>
</tbody>
</table>

---

**D.1.3.26. io.enmasse.v1beta1.Address**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>enum (enmasse.io/v1beta1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>enum (Address)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>ObjectMeta</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>io.enmasse.v1beta1.AddressSpec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>io.enmasse.v1beta1.AddressStatus</td>
</tr>
</tbody>
</table>

---

**D.1.3.27. io.enmasse.v1beta1.AddressList**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>required</td>
<td>Default: &quot;enmasse.io/v1beta1&quot;</td>
<td>enum (enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Schema</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>items</td>
<td>required</td>
<td>&lt; io.enmasse.v1beta1.Address &gt; array</td>
</tr>
<tr>
<td>kind</td>
<td>required</td>
<td>enum (AddressList)</td>
</tr>
</tbody>
</table>

**D.1.3.28. io.enmasse.v1beta1.AddressSpace**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>enum (enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>kind</td>
<td>enum (AddressSpace)</td>
</tr>
<tr>
<td>metadata</td>
<td>ObjectMeta</td>
</tr>
<tr>
<td>spec</td>
<td>io.enmasse.v1beta1.AddressSpaceSpec</td>
</tr>
<tr>
<td>status</td>
<td>io.enmasse.v1beta1.AddressSpaceStatus</td>
</tr>
</tbody>
</table>

**D.1.3.29. io.enmasse.v1beta1.AddressSpaceList**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>Default : &quot;enmasse.io/v1beta1&quot;</td>
<td>enum (enmasse.io/v1beta1)</td>
</tr>
<tr>
<td>items</td>
<td>required</td>
<td>&lt; io.enmasse.v1beta1.AddressSpace &gt; array</td>
</tr>
<tr>
<td>kind</td>
<td>required</td>
<td>enum (AddressSpaceList)</td>
</tr>
</tbody>
</table>

**D.1.3.30. io.enmasse.v1beta1.AddressSpaceSpec**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>authenticationService</strong>&lt;br&gt; optional</td>
<td></td>
<td>authenticationService</td>
</tr>
<tr>
<td><strong>connectors</strong>&lt;br&gt; optional</td>
<td>List of connectors to create.</td>
<td>&lt;io.enmasse.v1beta1.AddressSpaceSpecConnector&gt; array</td>
</tr>
<tr>
<td><strong>endpoints</strong>&lt;br&gt; optional</td>
<td></td>
<td>&lt;endpoints&gt; array</td>
</tr>
<tr>
<td><strong>networkPolicy</strong>&lt;br&gt; optional</td>
<td></td>
<td>networkPolicy</td>
</tr>
<tr>
<td><strong>plan</strong>&lt;br&gt; required</td>
<td></td>
<td>string</td>
</tr>
<tr>
<td><strong>type</strong>&lt;br&gt; required</td>
<td></td>
<td>io.enmasse.v1beta1.AddressSpaceType</td>
</tr>
</tbody>
</table>

**authenticationService**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong>&lt;br&gt; optional</td>
<td>string</td>
</tr>
<tr>
<td><strong>overrides</strong>&lt;br&gt; optional</td>
<td>overrides</td>
</tr>
<tr>
<td><strong>type</strong>&lt;br&gt; optional</td>
<td>string</td>
</tr>
</tbody>
</table>

**overrides**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>host</strong>&lt;br&gt; optional</td>
<td>string</td>
</tr>
<tr>
<td><strong>port</strong>&lt;br&gt; optional</td>
<td>integer</td>
</tr>
</tbody>
</table>
## realm

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>realm</td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>

## endpoints

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>cert</td>
<td>cert</td>
</tr>
<tr>
<td>exports</td>
<td>exports &gt; array</td>
</tr>
<tr>
<td>expose</td>
<td>expose</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>service</td>
<td>string</td>
</tr>
</tbody>
</table>

## cert

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>provider</td>
<td>string</td>
</tr>
<tr>
<td>secretName</td>
<td>string</td>
</tr>
<tr>
<td>tlsCert</td>
<td>string</td>
</tr>
<tr>
<td>tlsKey</td>
<td>string</td>
</tr>
</tbody>
</table>

## exports
### Name

<table>
<thead>
<tr>
<th>kind</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>enum (ConfigMap, Secret, Service)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>string</td>
</tr>
</tbody>
</table>

### expose

<table>
<thead>
<tr>
<th>annotations</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>object</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>loadBalancerPorts</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>&lt; string &gt; array</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>loadBalancerSourceRanges</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>&lt; string &gt; array</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>routeHost</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>routeServicePort</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>routeTlsTermination</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>string</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>type</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>enum (route, loadbalancer)</td>
</tr>
</tbody>
</table>

### networkPolicy

<table>
<thead>
<tr>
<th>egress</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>&lt; io.k8s.api.networking.v1.NetworkPolicyEgressRule &gt; array</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ingress</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>&lt; io.k8s.api.networking.v1.NetworkPolicyIngressRule &gt; array</td>
</tr>
</tbody>
</table>

D.1.3.31. io.enmasse.v1beta1.AddressSpaceSpecConnector
### REST API Reference

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>addresses</td>
<td>Addresses to make be accessible via this address space.</td>
<td><code>&lt; addresses &gt; array</code></td>
</tr>
<tr>
<td>credentials</td>
<td>Credentials used when connecting to endpoints. Either 'username' and 'password', or 'secret' must be defined.</td>
<td>credentials</td>
</tr>
<tr>
<td>endpointHosts</td>
<td>List of hosts that should be connected to. Must contain at least 1 entry.</td>
<td><code>&lt; endpointHosts &gt; array</code></td>
</tr>
<tr>
<td>name</td>
<td>Name of the connector.</td>
<td>string</td>
</tr>
<tr>
<td>tls</td>
<td>TLS settings for the connectors. If not specified, TLS will not be used.</td>
<td>tls</td>
</tr>
</tbody>
</table>

#### addresses

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Identifier of address pattern. Used to uniquely identify a pattern</td>
<td>string</td>
</tr>
<tr>
<td>pattern</td>
<td>Pattern used to match addresses. The pattern will be prefixed by the connector name and a forward slash ('myconnector/'). A pattern consists of one or more tokens separated by a forward slash /. A token can be one of the following: a * character, a # character, or a sequence of characters that do not include /, *, or #. The * token matches any single token. The # token matches zero or more tokens. * has higher precedence than #, and exact match has the highest precedence.</td>
<td>string</td>
</tr>
</tbody>
</table>

#### credentials

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Password to use for connector. Either 'value' or 'secret' must be specified.</td>
<td>password</td>
</tr>
<tr>
<td>username</td>
<td>Username to use for connector. Either 'value' or 'secret' must be specified.</td>
<td>username</td>
</tr>
</tbody>
</table>

#### password
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>optional string</td>
<td></td>
</tr>
<tr>
<td>valueFromSecret</td>
<td>optional valueFromSecret</td>
<td></td>
</tr>
</tbody>
</table>

**valueFromSecret**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Key to use for looking up password entry. Default :&quot;password&quot;</td>
<td>string</td>
</tr>
<tr>
<td>name</td>
<td>Name of Secret containing password.</td>
<td>string</td>
</tr>
</tbody>
</table>

**username**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>optional string</td>
<td></td>
</tr>
<tr>
<td>valueFromSecret</td>
<td>optional valueFromSecret</td>
<td></td>
</tr>
</tbody>
</table>

**valueFromSecret**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Key to use for looking up username entry. Default :&quot;username&quot;</td>
<td>string</td>
</tr>
<tr>
<td>name</td>
<td>Name of Secret containing username.</td>
<td>string</td>
</tr>
</tbody>
</table>

**endpointHosts**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Host to connect to.</td>
<td>string</td>
</tr>
</tbody>
</table>
### APPENDIX D. REST API REFERENCE

#### port

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>port</strong>&lt;br&gt;required</td>
<td>Port to connect to.</td>
<td>integer</td>
</tr>
</tbody>
</table>

#### tls

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>caCert</strong>&lt;br&gt;optional</td>
<td>CA certificate to be used by the connector. Either 'value' or 'secret'.</td>
<td>caCert</td>
</tr>
<tr>
<td><strong>clientCert</strong>&lt;br&gt;optional</td>
<td>Client certificate to be used by the connector. Either 'value' or 'secret'.</td>
<td>clientCert</td>
</tr>
</tbody>
</table>

#### caCert

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>value</strong>&lt;br&gt;optional</td>
<td>PEM encoded value of CA certificate</td>
<td>string</td>
</tr>
<tr>
<td><strong>valueFromSecret</strong>&lt;br&gt;optional</td>
<td>Secret containing CA certificate to be used by the connector.</td>
<td>valueFromSecret</td>
</tr>
</tbody>
</table>

#### valueFromSecret

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>key</strong>&lt;br&gt;optional</td>
<td>Key to use for looking up CA certificate entry. Default: &quot;ca.crt&quot;</td>
<td>string</td>
</tr>
<tr>
<td><strong>name</strong>&lt;br&gt;optional</td>
<td>Name of Secret containing CA certificate.</td>
<td>string</td>
</tr>
</tbody>
</table>

#### clientCert

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>value</strong>&lt;br&gt;optional</td>
<td>PEM encoded value of client certificate</td>
<td>string</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Schema</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>valueFromSecret</td>
<td>Secret containing client certificate to be used by the connector.</td>
<td>valueFromSecret</td>
</tr>
<tr>
<td>key</td>
<td>Key to use for looking up client certificate entry.</td>
<td>string</td>
</tr>
<tr>
<td>name</td>
<td>Name of Secret containing client certificate.</td>
<td>string</td>
</tr>
</tbody>
</table>

**D.1.3.32. io.enmasse.v1beta1.AddressSpaceStatus**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>connectors</td>
<td>List of connectors with status.</td>
<td><code>&lt; io.enmasse.v1beta1.AddressSpaceStatusConnector &gt; array</code></td>
</tr>
<tr>
<td>endpointStatuses</td>
<td></td>
<td><code>&lt; endpointStatuses &gt; array</code></td>
</tr>
<tr>
<td>isReady</td>
<td></td>
<td>boolean</td>
</tr>
<tr>
<td>messages</td>
<td></td>
<td><code>&lt; string &gt; array</code></td>
</tr>
</tbody>
</table>

**endpointStatuses**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>cert</td>
<td>string</td>
</tr>
<tr>
<td>externalHost</td>
<td>string</td>
</tr>
<tr>
<td>Name</td>
<td>Schema</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>externalPorts</td>
<td><code>&lt;externalPorts&gt;</code> array</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>serviceHost</td>
<td>string</td>
</tr>
<tr>
<td>servicePorts</td>
<td><code>&lt;servicePorts&gt;</code> array</td>
</tr>
</tbody>
</table>

**externalPorts**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
</tr>
</tbody>
</table>

**servicePorts**

<table>
<thead>
<tr>
<th>Name</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
</tr>
</tbody>
</table>

**D.1.3.33. io.enmasse.v1beta1.AddressSpaceStatusConnector**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>isReady</td>
<td>'true' if connector is operating as expected, 'false' if not.</td>
<td>boolean</td>
</tr>
<tr>
<td>messages</td>
<td>Messages describing the connector state.</td>
<td><code>&lt;string&gt;</code> array</td>
</tr>
<tr>
<td>name</td>
<td>Name of connector.</td>
<td>string</td>
</tr>
</tbody>
</table>
D.1.3.34. 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.1.3.35. io.enmasse.v1beta1.AddressSpec

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>required</td>
<td>string</td>
</tr>
<tr>
<td>forwarders</td>
<td>optional</td>
<td>list of io.enmasse.v1beta1.AddressSpecForwarder</td>
</tr>
<tr>
<td>plan</td>
<td>required</td>
<td>string</td>
</tr>
<tr>
<td>type</td>
<td>required</td>
<td>io.enmasse.v1beta1.AddressType</td>
</tr>
</tbody>
</table>

D.1.3.36. io.enmasse.v1beta1.AddressSpecForwarder

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>direction</td>
<td>required</td>
<td>enum (in, out)</td>
</tr>
<tr>
<td>name</td>
<td>required</td>
<td>string</td>
</tr>
<tr>
<td>remoteAddress</td>
<td>required</td>
<td>string</td>
</tr>
</tbody>
</table>

D.1.3.37. io.enmasse.v1beta1.AddressStatus

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>forwarders</td>
<td>optional</td>
<td>list of io.enmasse.v1beta1.AddressStatusForwarder</td>
</tr>
</tbody>
</table>

### D.1.3.38. io.enmasse.v1beta1.AddressStatusForwarder

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>isReady</td>
<td>'true' if forwarder is operating as expected, 'false' if not.</td>
<td>boolean</td>
</tr>
<tr>
<td>messages</td>
<td>Messages describing the forwarder state.</td>
<td>&lt; string &gt; array</td>
</tr>
<tr>
<td>phase</td>
<td>Name of forwarder.</td>
<td>string</td>
</tr>
</tbody>
</table>

### D.1.3.39. 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.1.3.40. 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>cidr</td>
<td>CIDR is a string representing the IP Block Valid examples are &quot;192.168.1.1/24&quot;</td>
<td>string</td>
</tr>
<tr>
<td>except</td>
<td>Except is a slice of CIDRs that should not be included within an IP Block Valid examples are &quot;192.168.1.1/24&quot; Except values will be rejected if they are outside the CIDR range</td>
<td>&lt; string &gt; array</td>
</tr>
</tbody>
</table>

### D.1.3.41. 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>ports</td>
<td>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.</td>
<td><code>&lt; io.k8s.api.networking.v1.NetworkPolicyPort &gt; array</code></td>
</tr>
<tr>
<td>to</td>
<td>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.</td>
<td><code>&lt; io.k8s.api.networking.v1.NetworkPolicyPeer &gt; array</code></td>
</tr>
</tbody>
</table>

D.1.3.42. 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>from</td>
<td>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 one item, this rule allows traffic only if the traffic matches at least one item in the from list.</td>
<td><code>&lt; io.k8s.api.networking.v1.NetworkPolicyPeer &gt; array</code></td>
</tr>
<tr>
<td>ports</td>
<td>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.</td>
<td><code>&lt; io.k8s.api.networking.v1.NetworkPolicyPort &gt; array</code></td>
</tr>
</tbody>
</table>

D.1.3.43. io.k8s.api.networking.v1.NetworkPolicyPeer

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipBlock</td>
<td>IPBlock defines policy on a particular IPBlock. If this field is set then neither of the other fields can be.</td>
<td>io.k8s.api.networking.v1.IPBlock</td>
</tr>
</tbody>
</table>
### D.1.3.44. io.k8s.api.networking.v1.NetworkPolicyPort

NetworkPolicyPort describes a port to allow traffic on

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>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.</td>
<td>io.k8s.apimachinery.pkg.util.intstr.IntOrString</td>
</tr>
<tr>
<td>protocol</td>
<td>The protocol (TCP or UDP) which traffic must match. If not specified, this field defaults to TCP.</td>
<td>string</td>
</tr>
</tbody>
</table>

### D.1.3.45. 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>matchExpressions</td>
<td>matchExpressions is a list of label selector requirements. The requirements are ANDed.</td>
<td>&lt; io.k8s.apimachinery.pkg.apis.meta.v1.LabelSelectorRequirement &gt; array</td>
</tr>
</tbody>
</table>
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.1.3.46. 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.

D.1.3.47. 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 2020-11-16 18:49:14 UTC