RabbitMQ

Feature: Bus

Editions: Cloud, Corporate, Store

> This feature is for development preview only. It is not intended for production, QA, or demonstration at this time.

> This topic is for reference purposes only. Use Rabbit provisioning to automatically create these resources for each organization.

Kensium RMS uses the RabbitMQ message broker to coordinate communication amongst an organization's RMS servers and tenants. This section describes some of the internal details of how that communication takes place, and can be helpful to diagnose communication problems.

Key Concepts

Before proceeding, it's important to understand the key concepts behind RabbitMQ. Refer to the documentation and examples at https://www.rabbitmq.com.

In particular, the following RabbitMQ concepts should be understood:

- Virtual hosts
- Exchanges
- Queues
- Routing keys
- Bindings

- Shovels

Virtual Hosts

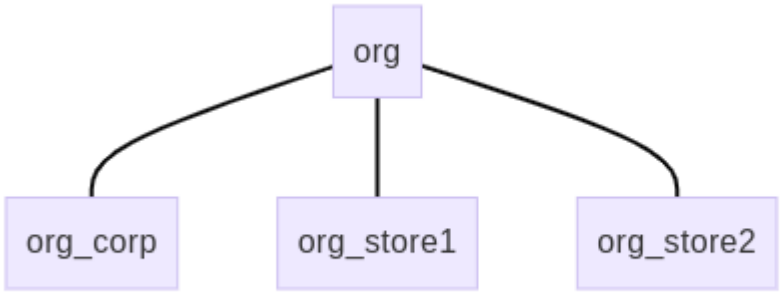
Each RMS tenant – whether it is a cloud, corporate or store tenant – is associated with its own RabbitMQ *virtual host*. The virtual host is responsible for communicating messages to and from the RMS tenant.

Messages are either:

- *Local* – sent and received by the same tenant (useful for queuing work)
- *Remote* – sent by the tenant, and delivered to another tenant (and virtual host) for processing

Remote messages must use RabbitMQ *shovels* to communicate from the source tenant virtual host to the destination tenant virtual host. Furthermore, it does this through a *hub* virtual host that is typically located in an AWS AmazonMQ instance.

Given an example organization named org with two stores, the following diagram shows the arrangement and naming of virtual hosts:



Of note:

- The hub virtual host is named after the organization's organization ID (org).
- The local virtual host for the Corporate tenant is named after the organization and corp tenant role (org_corp).
- The local virtual hosts for the Store tenants are named after the organization ID and store Site IDs (org_store1 and org_store2).

The Cloud tenant does not get its own local virtual host; it uses the hub virtual host (org).

Multiple Brokers

RabbitMQ virtual hosts can be located in one or more broker installations.

For production, there are multiple broker installations:

1. One for the Kensium Cloud environment, hosted using AmazonMQ.
2. One for the organization corporate environment, using a local RabbitMQ installation.
3. Typically one for each store environment, each using local RabbitMQ installations

> For simple deployments – where the organization's corporate and store environments are combined on the same network – a single broker instance can serve both corporate and store virtual hosts.

For development, virtual hosts can be located either:

- On a single RabbitMQ broker running on localhost
- To test offline message delivery: on a localhost broker as well as a *hub* broker hosted using AmazonMQ.

Routing Keys

RMS sends and publishes messages using a routing key, which determines how the message should be routed to its intended recipient(s). The following routing key formats are used:

- default
  - Send the message directly to the same RMS tenant, using the default queue.
  - RMS typically uses these types of messages to offload processing to background tasks.
- error
  - Send the message directly to the same RMS tenant, using the error queue.
  - RMS uses these types of messages to record unexpected errors encountered while processing other incoming messages.
- local.{class}
  - Publish the message directly to the same RMS tenant.
  - The message class indicates the type – and priority – of the message, indicating which queue should receive the message.
  - Currently, there is only the default class. These local.default messages are routed to the

default queue.
    - This behavior is similar to the default message, except that a *published* message does not result in an error if it is not handled by the RMS tenant.
- remote.{dest}.{class}
    - Publish the message to a remote tenant, using the hub virtual host to deliver the message.
    - The message dest indicate which tenants should receive the message.
    - The message class indicates the type – and priority – of the message, indicating which queue should receive the message.

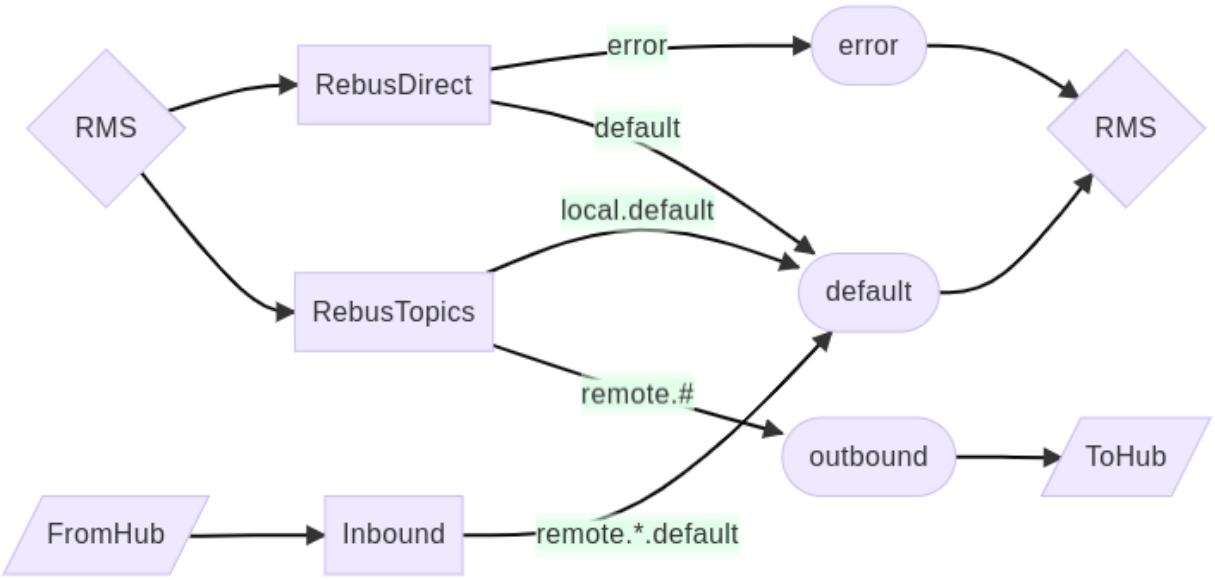
The available message destinations are as follows:

- cloud – the Cloud tenant.
- corp – the Corporate tenant.
- all – all tenants, including the tenant that sent the message.
- stores – all store tenants.
- xcloud – all tenants, except the Cloud tenant. This may include the tenant that sent the message.
- xcorp – all tenants, except the Corporate tenant. This may include the tenant that sent the message.
- xstores – all tenants, except store tenants (i.e. the Cloud and Corporate tenant). This may include the tenant that sent the message.
- Other values are the *Site ID* of the store that should receive the message. The Site ID is configured the first time the store tenant is configured.

The virtual host *topologies* listed below describe how the routing key is used to deliver messages.

Local Topology (Corporate and Store)

The local virtual host for a corporate or store tenant defines the same RabbitMQ exchanges, queues, and bindings.



The local virtual host is provisioned with the following:

- A default queue to receive messages with a default message class and priority.
- An error queue to receive error messages that RMS uses to record unexpected errors while processing other messages.
- A RebusDirect direct exchange that forwards default and error direct messages to their respective queues.
- A RebusTopics direct exchange that:
  - Forwards local default-class messages to the default queue.
  - Forwards remote messages to the outbound queue.
- An outbound queue that temporarily stores messages for other tenants.

A ToHub RabbitMQ shovel forwards remote messages stored in the outbound queue to the hub virtual host. The shovel will deliver the remote messages in near real-time if an internet connection is available; otherwise remote messages remain in the outbound queue until an internet connection is available.

Similarly, a FromHub RabbitMQ shovel forwards incoming messages from the hub virtual host – i.e. messages sent from other tenants – to the Inbound topic exchange. The Inbound exchange then:

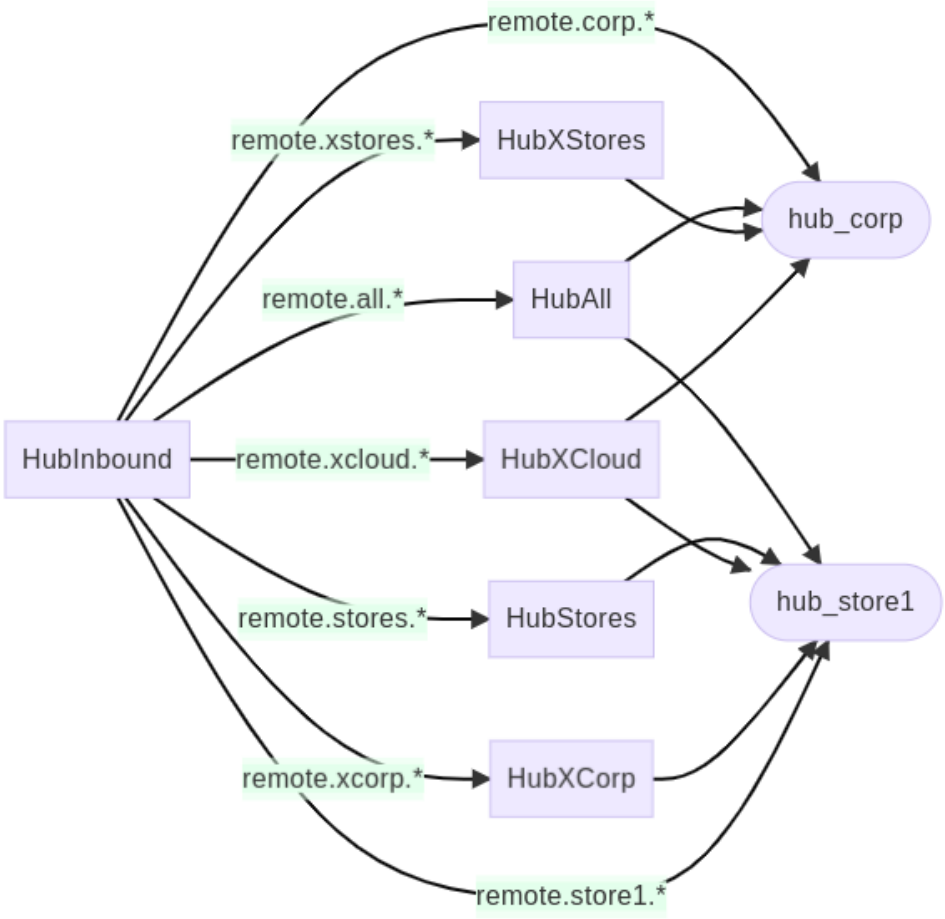
- Forwards messages with the default message class to the default queue.

Hub Topology

The primary purpose of the hub virtual host is to route messages from a tenant to one or more other tenants. Provisioning this virtual host:

- Creates a set of exchanges that are used by all tenants.
- One queue for each corporate and store tenant.
- Queue bindings dependent on the tenant role (corporate or store).

The topology for an organization with one corporate tenant and one store tenant is shown below:



The hub virtual host is provisioned with a HubInbound topic exchange to receive incoming messages from other tenants. The ToHub shovel for the corporate and store local virtual host forwards remote messages to this exchange.

From there, messages are forwarded to fanout exchanges based on the message destination (see message destination types in *routing keys* above).

Tenant Queues

When a corporate or store tenant is provisioned, it will create a *tenant queue* on the hub virtual host to store its messages that are incoming from other tenants.

The name of the queue is based on the tenant role and site ID:

- hub_corp is the queue for the Corporate tenant.
- hub_{SiteId} is the queue for a Store tenant, identified by its SiteId.

The tenant queue is bound to the fanout exchanges, based on the tenant role:

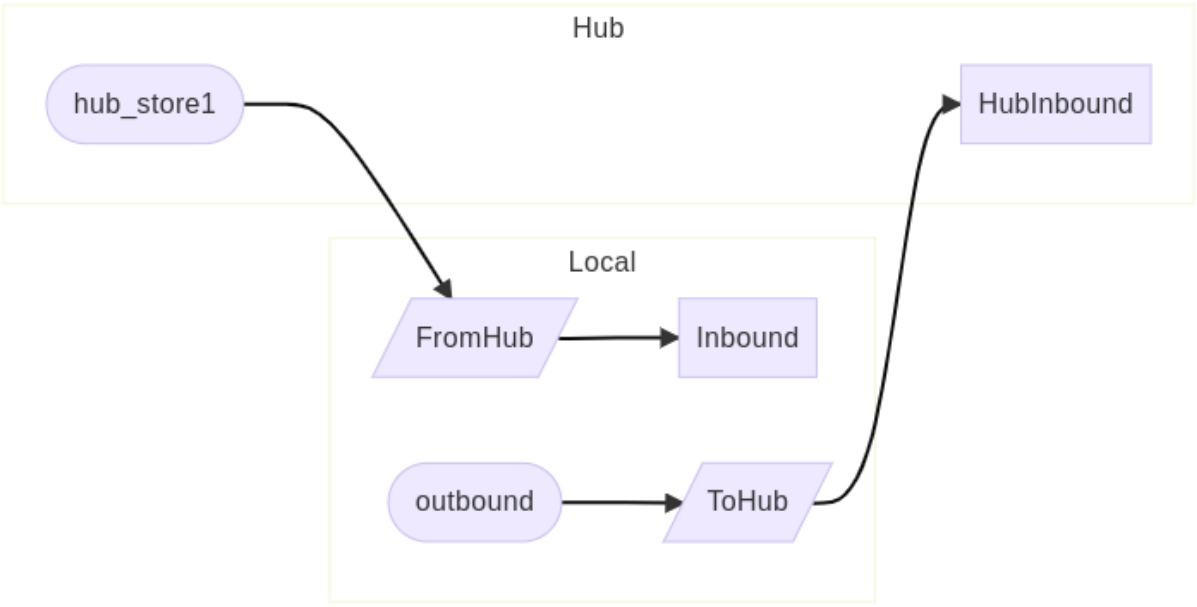
- The Corporate tenant is bound to the HubAll, HubXStores and HubXCloud exchanges.
- Store tenants are bound to the HubAll, HubStores, HubXCorp and HubXCloud exchanges.

In addition, the tenant queue is bound to the HubInbound exchange for any messages that are intended directly for the tenant (e.g. remote.corp.* or remote.store1.*).

The tenant queue will store the tenant messages, until the tenant's local FromHub shovel forwards them to the tenant's local virtual host for processing. The shovel will deliver the remote messages in near real-time if an internet connection is available; otherwise remote messages remain in the outbound queue until an internet connection is available.

Shovels

The following diagram shows how the shovels between the local and hub virtual hosts are configured for an example store *store1*:



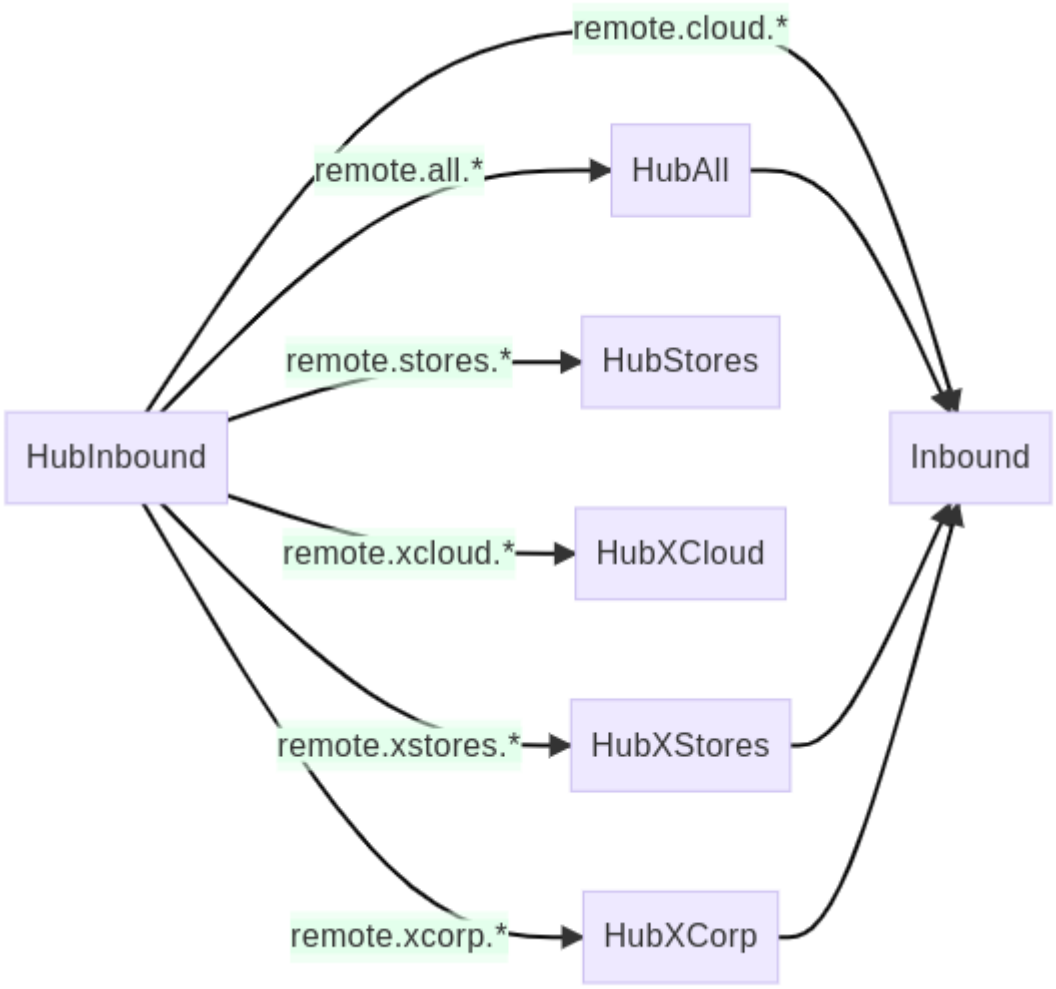
The shovels are maintained at the local virtual host, as network communications and firewalls typically allow connections from the local network to the cloud – but not the reverse. Even though network communication is one direction, a shovel can be configured to send data in either direction.

Hub Topology (Cloud)

As the hub virtual host is already deployed in the cloud (AmazonMQ), as an optimization we do not introduce a separate local virtual host for the cloud tenant. The cloud tenant accesses the hub virtual host directly.

As a result, the hub virtual host also includes exchanges and queues for the Cloud tenant. The topology of these items are similar to the corporate and store tenants, except that:

- No shovels are necessary – messages are delivered within the same virtual host.
- There is no outbound queue. Outgoing remote messages in the RebusTopics exchange are routed directly to the shared HubInbound exchange.
- There is no separate tenant queue. Incoming remote messages are routed directly to the cloud tenant’s Inbound exchange.



And within the same hub virtual host, the exchanges and queues specific to the cloud tenant are: