Federation Plugin

The high-level goal of the federation plugin is to transmit messages between brokers without requiring clustering. This is useful for various reasons:

Loose coupling
: The federation plugin can transmit messages between brokers (or clusters) in different administrative domains:
  - they may have different users and virtual hosts;
  - they may run on different versions of RabbitMQ and Erlang.

WAN-friendly
: The federation plugin uses AMQP to communicate between brokers, and is designed to tolerate intermittent connectivity.

Specificity
: A broker can contain federated *and* local-only components - you don't need to federate everything if you don't want to.

Scalability
: Federation does not require O(n^2) connections between *n* brokers (although this is the easiest way to set things up), which should mean it scales better.

What does it do?

The federation plugin allows you to make exchanges and queues *federated*. A federated exchange or queue can receive messages from one or more *upstreams* (remote exchanges and queues on other brokers). A federated exchange can route messages published upstream to a local queue. A federated queue lets a local consumer receive messages from an upstream queue.

For more details, see the documentation on **federated exchanges** and **federated queues**.

Getting Started

The federation plugin is included in the RabbitMQ distribution. To enable it, use **rabbitmq-plugins**:

```
rabbitmq-plugins enable rabbitmq_federation
```

When using the management plugin, you will also want to enable `rabbitmq_federation_management`:

```
rabbitmq-plugins enable rabbitmq_federation_management
```

When using a federation in a cluster, all the nodes of the cluster should have the federation plugin installed.

Information about federation is stored in the RabbitMQ database, along with users, permissions, queues, etc. There are three levels of configuration involved in federation:

- Upstreams - each upstream defines how to connect to another broker.
- Upstream sets - each upstream set groups together a set of upstreams to use for federation.
- Policies - each policy selects a set of exchanges, queues or both, and applies a single upstream or an upstream set to those objects.

In practice, for simple use cases you can almost ignore the existence of upstream sets, since there is an implicitly-defined upstream set called `all` to which all upstreams are added.

Upstreams and upstream sets are both instances of *parameters*. Like exchanges and queues, each virtual host has its own distinct set of parameters and policies. For more generic information on parameters and policies, see the documentation on **parameters and policies**. For full details on the parameters used by federation, see the **federation reference**.

Parameters and policies can be set in three ways - either with an invocation of `rabbitmqctl`, a call to the management HTTP API, or (usually) through the web UI presented by `rabbitmq_federation_management`. (The web UI does not present all possibilities - in particular, it does not allow you to manage upstream sets.)

A simple example

Here we will federate all the built-in exchanges except for the default exchange, with a single upstream. The upstream will be defined to buffer messages when disconnected for up to one hour (3600000ms).

Define an upstream:

rabbitmqctl	`rabbitmqctl set_parameter federation-upstream my-upstream \` `'{"uri":"amqp://server-name","expires":3600000}'`
rabbitmqctl (Windows)	`rabbitmqctl set_parameter federation-upstream my-upstream ^` `"{""uri"":""amqp://server-name"",""expires"":3600000}"`
HTTP API	`PUT /api/parameters/federation-upstream/%2f/my-upstream`

	`{"value":{"uri":"amqp://server-name","expires":3600000}}`
Web UI	Navigate to Admin > Federation Upstreams > Add a new upstream. Enter "my-upstream" next to Name, "amqp://server-name" next to URI, and 36000000 next to Expiry. Click Add upstream.

Then define a policy to use this upstream:

rabbitmqctl	`rabbitmqctl set_policy --apply-to exchanges federate-me "^amq\." \ '{"federation-upstream-set":"all"}'`
rabbitmqctl (Windows)	`rabbitmqctl set_policy --apply-to exchanges federate-me "^amq\." ^ "{""federation-upstream-set"":""all""}"`
HTTP API	PUT `/api/policies/%2f/federate-me` `{"pattern":"^amq\.", "definition":{"federation-upstream-set":"all"}, \ "apply-to":"exchanges"}`
Web UI	Navigate to Admin > Policies > Add / update a policy. Enter "federate-me" next to "Name", "^amq\." next to "Pattern", choose "Exchanges" from the "Apply to" drop down list and enter "federation-upstream-set" = "all" in the first line next to "Policy". Click "Add" policy.

We tell the policy to federate all exchanges whose names begin with "amq." (i.e. all the built in exchanges except for the default exchange) with (implicit) low priority, and to federate them using the implicitly created upstream set "all", which includes our newly-created upstream. Any other matching policy with a priority greater than 0 will take precedence over this policy.

The built in exchanges should now be federated. You can check that the policy has applied to the exchanges by checking the exchanges list in management or with:

```
rabbitmqctl list_exchanges name policy | grep federate-me
```

And you can check that federation links for each exchange have come up with Admin > Federation Status > Running Links or with:

```
rabbitmqctl eval 'rabbit_federation_status:status().'
```

In general there will be one federation link for each upstream that is applied to an exchange. So for example with three exchanges and two upstreams for each there will be six links.

For simple use this should be all you need - you will probably want to look at the **AMQP URI reference**. The **federation reference** contains more details on setting up upstreams and upstream sets.

Federating clusters

Clusters can be linked together with federation just as single brokers can. To summarise how clustering and federation interact:

- You can define policies and parameters on any node in the downstream cluster; once defined on one node they will apply on all nodes.
- Exchange federation links will start on any node in the downstream cluster. They will fail over to other nodes if the node they are running on crashes or stops.
- Queue federation links will start on the same node as the downstream queue. If the downstream queue is mirrored, they will start on the same node as the master, and will be recreated on the same node as the new master if the node the existing master is running on crashes or stops.
- To connect to an upstream cluster, you can specify multiple URIs in a single upstream. The federation link process will choose one of these URIs at random each time it attempts to connect.