Access Control

When the server first starts running, and detects that its database is uninitialised or has been deleted, it initialises a fresh database with the following resources:

- a virtual host named `/`
- a user named `guest` with a default password of `guest`, granted full access to the `/` virtual host.

It is advisable to **delete** the `guest` user or **change the password** to something private, particularly if your broker is accessible publicly.

"guest" user can only connect via localhost

By default, the `guest` user is prohibited from connecting to the broker remotely; it can only connect over a loopback interface (i.e. `localhost`). This applies both to AMQP and to any other protocols enabled via plugins. Any other users you create will not (by default) be restricted in this way.

This is configured via the `loopback_users` item in the **configuration file**.

If you wish to allow the `guest` user to connect from a remote host, you should set the `loopback_users` configuration item to `[]`. A complete `rabbitmq.config` which does this would look like:

```
[{rabbit, [{loopback_users, []}]}].
```

How permissions work

When an AMQP client establishes a connection to an AMQP server, it specifies a virtual host within which it intends to operate. A first level of access control is enforced at this point, with the server checking whether the user has any permissions to access the virtual hosts, and rejecting the connection attempt otherwise.

Resources, i.e. exchanges and queues, are named entities inside a particular virtual host; the same name denotes a different resource in each virtual host. A second level of access control is enforced when certain operations are performed on resources.

RabbitMQ distinguishes between *configure*, *write* and *read* operations on a resource. The *configure* operations create or destroy resources, or alter their behaviour. The *write* operations inject messages into a resource. And the *read* operations retrieve messages from a resource.

In order to perform an operation on a resource the user must have been granted the appropriate permissions for it. The following table shows what permissions on what type of resource are required for all the AMQP commands which perform permission checks.

AMQP command		configure	write	read
exchange.declare	(passive=false)	exchange		
exchange.declare	(passive=true)			
exchange.declare	(with **AE**)	exchange	exchange (AE)	exchange
exchange.delete		exchange		
queue.declare	(passive=false)	queue		
queue.declare	(passive=true)			
queue.declare	(with **DLX**)	queue	exchange (DLX)	queue
queue.delete		queue		
exchange.bind			exchange (destination)	exchange (source)
exchange.unbind			exchange (destination)	exchange (source)
queue.bind			queue	exchange
queue.unbind			queue	exchange
basic.publish			exchange	
basic.get				queue
basic.consume				queue
queue.purge				queue

Permissions are expressed as a triple of regular expressions - one each for configure, write and read - on per-vhost basis. The user is granted the respective permission for operations on all resources with names matching the regular expressions. *(Note: For convenience RabbitMQ maps AMQP's default exchange's blank name to 'amq.default' when performing permission checks.)*

The regular expression `'^$'`, i.e. matching nothing but the empty string, covers all resources and effectively stops the user from performing any operation. Standard AMQP resource names are prefixed with `amq.` and server generated names are

prefixed with `amq.gen`. For example, `'^(amq\.gen.*|amq\.default)$'` gives a user access to server-generated names and the default exchange. The empty string, `''` is a synonym for `'^$'` and restricts permissions in the exact same way.

RabbitMQ may cache the results of access control checks on a per-connection or per-channel basis. Hence changes to user permissions may only take effect when the user reconnects.

For details of how to set up access control, please see the **Access Control section** of the **rabbitmqctl(1) man page**.