ellucian.

Degree Works

RabbitMQ

Installation Guide

Release 5.0.3.1
March 2020

Notices

Contents

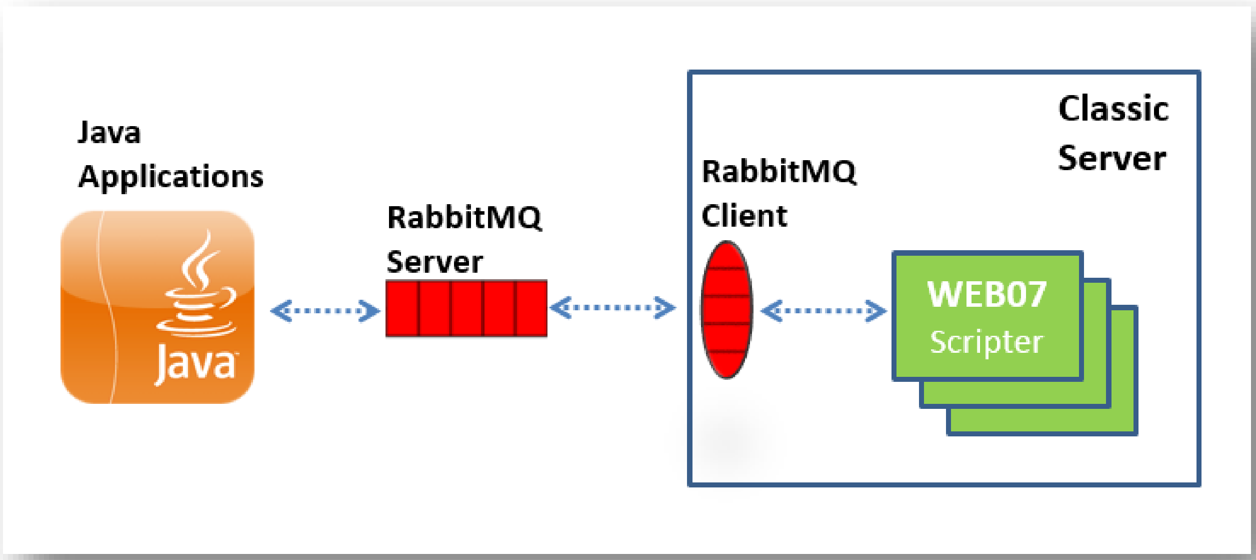
Introduction

RabbitMQ is a third party messaging system which allows Degree Works applications to communicate with one another. The RabbitMQ Server component does not generate heavy load on the system even with high activity in Degree Works. When you monitor the system you should find that it is not a resource intensive application. More information can be found at http://www.rabbitmq.com/.

There are two software components that must be installed: RabbitMQ Server and RabbitMQ Client. The **server** software component may be installed on any of the existing servers or on a new server. For example, you may install it on the classic server or on one of your application servers or on a new server – including a Windows server.

The **client** component must be installed on the classic server.

RabbitMQ allows applications to communicate as shown in this diagram. The Java applications communicate with each other and the classic daemons communicate via the RabbitMQ Server.



The traffic between the RabbitMQ broker and our applications is not encrypted. For that reason, **we strongly recommend that the traffic travel only in a private subnet not exposed to unsecured servers.**

RabbitMQ Server

Recommended platforms for installing RabbitMQ Server are: Linux flavors (Red Hat, Oracle Linux, CentOS), Sun Solaris and Windows. We recommend you do not attempt to install RabbitMQ Server on AIX or HP-UX; instead install it on a Windows server or on a Linux server. Before installing **RabbitMQ Server** the **Erlang Client** must be installed.

Note: We recommend you install the latest available versions of RabbitMQ and its recommended Erlang so you have the latest features and security patches. As of this writing the most current version of RabbitMQ is 3.8.2 with Erlang 21.3.8.

Linux Clients: Before installing RabbitMQ Server you first need to install Erlang. You will be installing the “Zero-dependency Erlang from RabbitMQ” using this link: https://github.com/rabbitmq/erlang-rpm/releases

We recommend you install the 21.3 version. You may have to scroll or click Next to find this version. If you are on Red Hat 6 then download **erlang-21.3.8.13-1.el6.x86_64.rpm**. If you are on Red Hat 7 then download **erlang-21.3.8.13-1.el7.x86_64.rpm**. Upload the rpm file to /tmp as Binary. As the root user install the rpm file you uploaded, for example:

```
# rpm -vih  erlang-21.3.8.13-1.el7.x86_64.rpm
```

Next create **/etc/yum.repos.d/rabbitmq-erlang.repo** and insert these contents:

```
[rabbitmq-erlang]
name=rabbitmq-erlang
baseurl=https://dl.bintray.com/rabbitmq-erlang/rpm/erlang/21/el/7
gpgcheck=1
gpgkey=https://dl.bintray.com/rabbitmq/Keys/rabbitmq-release-signing-key.asc
repo_gpgcheck=0
enabled=1
```

Now that Erlang is installed you need to download and install RabbitMQ Server. Using this link http://www.rabbitmq.com/install-rpm.html download either **rabbitmq-server-3.8.2-1.el6.noarch.rpm** or **rabbitmq-server-3.8.2-1.el7.noarch.rpm** based on your version of Red Hat.

As the root user issue this command:

```
# rpm --import https://www.rabbitmq.com/rabbitmq-release-signing-
key.asc
```

Upload the rpm file to /tmp as Binary and install, for example:

```
# yum install rabbitmq-server-3.8.2-1.el7.noarch.rpm
```

In order for rabbitmq to start automatically on boot run this command:

```
# chkconfig rabbitmq-server on
```

After RabbitMQ Server is installed, add the location of the RabbitMQ sbin directory to the root user’s path so you will not have to run it from its installation directory. You will need to run the **rabbitmqctl** script located in sbin when configuring RabbitMQ for Degree Works. For Linux clients a rabbitmq user account is created and will own the processes after being started as root.

Solaris clients: Go to the section below titled **Solaris sites Erlang and RabbitMQ Server Install instructions**.

Windows clients: Select **Install: Windows** to the right side of the screen. Are provided at http://www.rabbitmq.com/install-windows.html. Download the Erlang installation file from the **Erlang for Windows** link**.** Download the 20.3 version. Be sure to install Erlang as Administrator. When completed Erlang should be running as a background process “erl”.

Next download the RabbitMQ installation file from http://www.rabbitmq.com/install-windows.html and install it as Administrator. When the installation is complete, RabbitMQ should be running as a Windows service. Set the system path adding the location of the RabbitMQ sbin directory. To manage RabbitMQ open a command window as Administrator.

If you encounter a cookie mismatch error, it can be corrected by following instructions here: http://www.rabbitmq.com/windows-quirks.html.

ALL CLIENTS: After completing the installation, see the section **Managing the RabbitMQ Server** in this document so that you can test the installation and add a user for Degree Works.

Solaris sites Erlang and RabbitMQ Server Install Instructions

Using this link click on **Install: Generic Unix** http://www.rabbitmq.com/install-rpm.html

Solaris clients can install Erlang here: http://www.erlang.org/downloads. Click on the link **OTP 21.3 Source File** (you may have to scroll down to find the OTP 21.3 link). The Erlang installation instructons are here https://github.com/erlang/otp/blob/maint/HOWTO/INSTALL.md and are also inside the download file at otp_src_21.3/HOWTO/INSTALL.md. **Please refer to this document for details on the Erlang installation**. Basic steps are as follows.

As the root user transfer the file to an appropriate location for binaries on your RabbitMQ server and unpack:

```
# tar -zxf otp_src_21.3.tar.gz
```

Change directory to the new directory and set the ERL_TOP variable:

```
# cd otp_src_21.3
# export ERL_TOP=`pwd`
```

Run the following command to configure the build:

```
# ./configure
```

Next build the Erlang software with gmake (do not try make; this is a known Solaris issue).

```
# gmake
# gmake install
```

Erlang is now installed. Create a symbolic link for erlang in /usr/bin. Substitute the location of your Erlang install directory in the following command:

```
# ln -s /u02/otp_src_21.3/bin/erl /usr/bin/erl
```

Next begin the installation for RabbitMQ Server. On the RabbitMQ download page, click on the link **Install: Generic Unix** on the right then download the Generic Unix release tar file named rabbitmq-server-generic-unix-3.8.2.tar.xz (you may download a more recent version if it is available).

Decompress the file at an appropriate location on your server. A directory named rabbitmq_server-3.8.2 will be created, containing an sbin directory.

For Solaris you must change all of the scripts to use the following shell:

```
#!/usr/xpg4/bin/sh -e
```

In the sbin directory are 5 scripts. Change the first line of each script to the one above. In the rabbit-env script you must make a global change, changing *local* to *typeset*, e.g.:

```
rmq_realpath() {
    #local path=$1 #ellucian
    typeset path=$1
```

You may comment-out the original line and mark your changes as illustrated above. Make sure you change all occurrences of *local* to *typeset*.

You are now ready to start the RabbitMQ server. (You may need to use *svccfg* to setup rabbitmq-server as a service.)

```
# ./rabbitmq-server
```

You should add the location of the RabbitMQ sbin directory to the root user's path so you will not have to run it from its installation directory. You will need to run the **rabbitmqctl** script located in sbin when configuring RabbitMQ for Degree Works.

```
# export PATH=$PATH:/u02/rabbitmq-server/rabbitmq_server-3.8.2/sbin
```

Congratulatons, your installation of RabbitMQ Server is complete. You may now move on to the **Managing the RabbitMQ Server** section of this document.

In the future if you are having issues with RabbitMQ, as the root user check to see if it is up and running using one of these commands:

```
# rabbitmqctl status
# service rabbitmq-server status
```

If you see this error:

```
      Status of node 'rabbit@ip-10-100-26-72' ...
      Error: unable to connect to node 'rabbit@ip-10-100-26-72': nodedown
```

Then you need to restart:

```
# service rabbitmq-server restart
```

Managing the RabbitMQ Server

Once RabbitMQ server has been installed you should find a *rabbitmqctl* script in your path. Solaris users can set up rabbitmq-server as a service with svccfg. Windows users can manage RabbitMQ as a typical service. Linux users can use the following commands to manage the server:

Find out if rabbitmq is running: **`service rabbitmq-server status`**

Start the rabbitmq service: **`service rabbitmq-server start`**

Stop the rabbitmq service: **`service rabbitmq-server stop`**

Running **rabbitmqctl status** should give you similar information to what you see using the service command.

You must add a user for Degree Works with the *rabbitmqctl* script. This is the same user and password that you will be prompted for when you install Degree Works for the first time or when you perform the 5.0.0 update.

The following commands can be run on all platforms, including Windows from a command prompt.

```
# rabbitmqctl add_user <someuser> <somepwd>
```

This user is purely to allow access to RabbitMQ and is independent of all other Oracle or Unix users you need for Degree Works. Next set permissions for your user:

```
# rabbitmqctl set_permissions -p / <someuser> ".*" ".*" ".*"
```

You may use the *authenticate_user* option to verify that your user and password are setup and working.

```
# rabbitmqctl authenticate_user <someuser> <somepwd>
```

Please see the rabbitmqctl man page for more useful commands:

https://www.rabbitmq.com/man/rabbitmqctl.1.man.html

Enabling SSL/TLS in RabbitMQ Server

The default installation of RabbitMQ has SSL disabled and only TCP connections are used. Starting with the 5.0.2 release Degree Works can be optionally configured to communicate with RabbitMQ over SSL. To enable SSL connections you need to setup the RabbitMQ Server configuration file. If you choose not to enable SSL in RabbitMQ you may skip this section.

Here is an overview of how to enable SSL in RabbitMQ:

- The default port for TCP connections is 5672 and the default port for SSL connections is 5671 but these ports can be changed in rabbitmq.config.
- To enable SSL connections you need to change your **core.amqp.broker.port** shp-setting to your SSL port specified in rabbitmq.config – the default port is 5671. You also need to set **core.amqp.useSsl** to **true**.
- As the root user, save the configuration settings below into a new file named **/etc/rabbitmq/rabbitmq.config**.
- You must stage your SSL certificate, key and root ca file and configure their location in the rabbitmq.config file.

Before making these changes please review the scenarios below. In these examples, the TCP port is 5672 and the SSL port is 5671. If you are using different ports in rabbitmq.config then please substitute those ports below. For more information on SSL certificates see the section *Configuring SSL Certificates* below.

Scenario 1: You installed Degree Works for first time

These steps will help you setup your TEST environment to use SSL. At some point you will create a PROD environment wherein you will enable SSL there also.

Step 1: Test using TCP in TEST

After setting up the config file and restarting RabbitMQ Server you should leave the TEST shp-setting set to the TCP port 5672 before switching to SSL to ensure your config file is setup correctly. Run **webrestart** in TEST on your classic environment and restart the apps on the Java application servers. Review web.logs in TEST and perform basic testing.

Step 2: Test using SSL in TEST

In TEST change your **core.amqp.broker.port** to 5671 and **core.amqp.useSsl** to **true**. Run **webrestart** in TEST on your classic environment and restart the apps on the Java application servers. Review web.logs in TEST and perform basic testing.

Step 3: Test with SSL in PROD

When you eventually create your PROD environment then change **core.amqp.broker.port** to 5671 and **core.amqp.useSsl** to **true** in PROD. Run **webrestart** in PROD on your classic environment and restart the apps on the Java application servers. Review web.logs in both environments and perform basic testing.

Scenario 2: You upgraded TEST but PROD is on a pre-502 version

Because your production environment is not able to work with RabbitMQ over SSL you have to configure RabbitMQ to work with both SSL (in TEST) and TCP (in PROD). While you are testing SSL in your test environment, your production environment needs to continue to use the TCP communication protocol. The steps below will help you test with SSL in TEST while continuing with TCP in PROD. At some point you will then upgrade PROD to the same version as TEST and then you may configure PROD to use SSL.

Step 1: Test using TCP in both environments to ensure TCP connectivity works with the config file in place

After setting up the config file and restarting RabbitMQ Server you should leave both the PROD and TEST shp-setting set to the TCP port 5672 before switching to SSL to ensure your config file is setup correctly. Run **webrestart** in both PROD and TEST on your classic environments and restart the apps on the Java application servers. Review web.logs in both environments and perform basic testing.

Step 2: Test using TCP in PROD and SSL in TEST

In TEST change your **core.amqp.broker.port** to 5671 and also change **core.amqp.useSsl** to **true.** Run **webrestart** in TEST on your classic environment and restart the apps on the Java application servers. Review web.logs in both environments and perform basic testing.

Step 3: Test with SSL in PROD

When you eventually upgrade PROD to the same version as TEST you then can change your **core.amqp.broker.port** to 5671 and **core.amqp.useSsl** to **true** in PROD. Run **webrestart** in PROD on your classic environment and restart the apps on the Java application servers. Review web.logs in PROD and perform basic testing.

Scenario 3: Your TEST and PROD environments are both on a 502+ version

While you are testing SSL in your test environment, your production environment should continue to use the TCP communication protocol while you ensure SSL is functional in TEST. The steps below will help you test with SSL in TEST while continuing with TCP in PROD.

Step 1: Test using TCP in both environments to ensure TCP connectivity works with the config file in place

After setting up the config file and restarting RabbitMQ Server you should leave both the PROD and TEST shp-setting set to the TCP port 5672 before switching to SSL to ensure your config file is setup correctly. Run **webrestart** in both PROD and TEST on your classic environments and restart the apps on the Java application servers. Review web.logs in both environments and perform basic testing.

Step 2: Test using TCP in PROD and SSL in TEST

In TEST change your **core.amqp.broker.port** to 5671 and **core.amqp.useSsl** to **true**. Run **webrestart** in TEST on your classic environment and restart the apps on the Java application servers. Review web.logs in both environments and perform basic testing.

Step 3: Test with SSL in PROD

When you are ready, in PROD change your **core.amqp.broker.port** to 5671 and **core.amqp.useSsl** to **true**. Run **webrestart** in PROD on your classic environment and restart the apps on the Java application servers. Review web.logs in PROD and perform basic testing.

In the configuration example file below the **tcp_listeners** and the **ssl_listeners** are enabled. Once you have both TEST and PROD configured to use SSL you may choose to disable the tcp_listeners. The **ssl_options** must be configured based on where you have placed your pem files. See the section below on **Configuring SSL Certificates** for setting up your pem files.

```
%% -*- mode: erlang -*-
%% ----------------------------------------------------------------------------
%% RabbitMQ Sample Configuration File.
```

```
%%
%% Related doc guide: http://www.rabbitmq.com/configure.html. See
%% http://rabbitmq.com/documentation.html for documentation ToC.
%% ------------------------------------------------------------------------------
[
 {rabbit,
  [%%
   %% Networking
   %% ====================
   %%
   %% Related doc guide: http://www.rabbitmq.com/networking.html.

   %% To listen on a specific interface, provide a tuple of {IpAddress, Port}.
   %% For example, to listen only on localhost for both IPv4 and IPv6:
   %%
   %% {tcp_listeners, [{"127.0.0.1", 5672},
   %%                  {"::1",       5672}]},
   {tcp_listeners, [5672]},

   %% TLS listeners are configured in the same fashion as TCP listeners,
   %% including the option to control the choice of interface.
   %%
   {ssl_listeners, [5671]},

   %% Number of Erlang processes that will accept connections for the TCP
   %% and TLS listeners.
   %%
   {num_tcp_acceptors, 10},
   {num_ssl_acceptors, 10},

   %% TLS configuration.
   %%
   %% Related doc guide: http://www.rabbitmq.com/ssl.html.
   %%
   %% {ssl_options, [{cacertfile,           "/path/to/testca/cacert.pem"},
   %%                {certfile,             "/path/to/server/cert.pem"},
   %%                {keyfile,              "/path/to/server/key.pem"},
   %%                {verify,               verify_peer},
   %%                {fail_if_no_peer_cert, false}]},
   {ssl_options, [{cacertfile,           "/path/to/cacert.pem"},
                  {certfile,             "/path/to/cert.pem"},
                  {keyfile,              "/path/to/key.pem"},
                  {verify,               verify_none},
                  {fail_if_no_peer_cert, false}]}

  ]}

%% Other options can be found in your rabbitmq install directory, for example:
%% /usr/share/doc/rabbitmq-server-3.8.2/rabbitmq.config.example

].
```

Make sure to copy this full set of text including the ending bracket and period.

More configuration options exist. You may view them in the example config file that was installed on your server, for example:
`/usr/share/doc/rabbitmq-server-3.8.2/rabbitmq.config.example`

To find out more information about setting up RabbitMQ with SSL please review the RabbitMQ web site:

https://www.rabbitmq.com/ssl.html

Configuring SSL Certificates

When configuring RabbitMQ for SSL you will be required to configure rabbitmq.config to reference an SSL certificate, key and intermediate rootca file. You may use a self-signed certificate and key in TEST but in PROD you should use a certficate issued by a verified Certificate Authority. Instructions to create and configure a self-signed certificate can be found at https://www.rabbitmq.com/ssl.html.

When configuring RabbitMQ to use your certificate, rootca and key, you may need to convert them to the PEM format used by OpenSSL:

```
openssl x509 -in mycert.crt -out mycert.pem -outform PEM
```

If you have not received a separate rootca file from your CA, the certificate itself may contain a URI where you can identify and retrieve the rootca file. For example, print the certificate contents with the following command:

```
openssl x509 -in cert.pem -noout -text
```

Look for the following in the output:

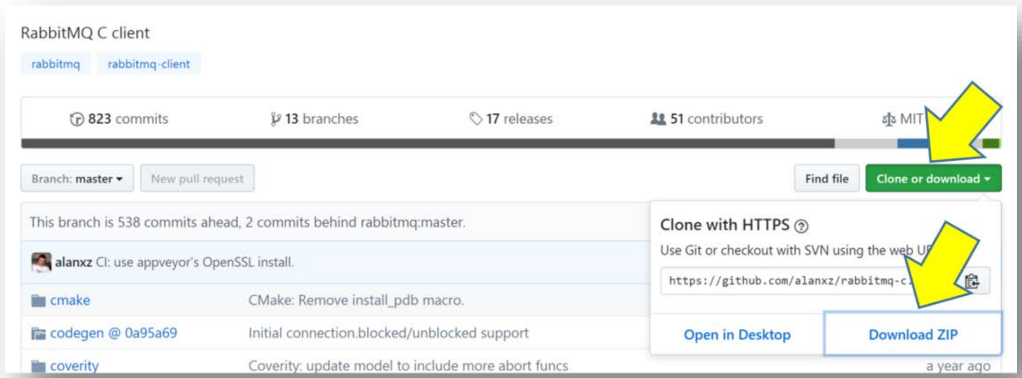
```
    Authority Information Access:
        OCSP - URI:http://ocsp.digicert.com
        CA Issuers - URI:http://cacerts.digicert.com/DigiCertSHA2HighAssuranceServerCA.crt
```

To retrieve this rootca file for Digicert, place the CA Issuers URI in a browser and the associated rootca file should download. Transfer this file to your RabbitMQ server's SSL certificate location and convert it to the PEM format.

RabbitMQ Client

All clients must install the client component on your Degree Works classic server. Visit this page to download the RabbitMQ zip file https://github.com/alanxz/rabbitmq-c.

Click the green “**Clone or download**” button and then click the “**Download ZIP**” button:


RabbitMQ C client

This branch is 538 commits ahead, 2 commits behind rabbitmq:master.

alanxz CI: use appveyor's OpenSSL install.

cmake	CMake: Remove install_pdb macro.
codegen @ 0a95a69	Initial connection.blocked/unblocked support
coverity	Coverity: update model to include more abort funcs

a year ago

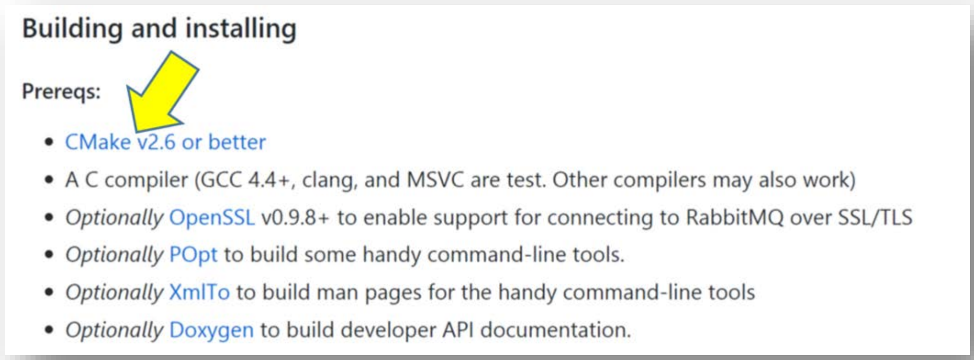
Once the **rabbitmq-c-master.zip** file has been downloaded keep it in a safe place to be used later in the instructions.

Before installing RabbitMQ you must first install **cmake** it you don’t already have it. On the classic server check to see if cmake is already installed:

```
# which cmake
```

If it is not found then install it following the instructions below. If cmake is already installed then jump down to the **Install RabbitMQ C client** section.

On the https://github.com/alanxz/rabbitmq-c web page scroll down to the **Getting Started** section. Under **Prereqs**, right-click the link **Cmake v2.6 or better** as shown below and choose “Open link in a new tab”.

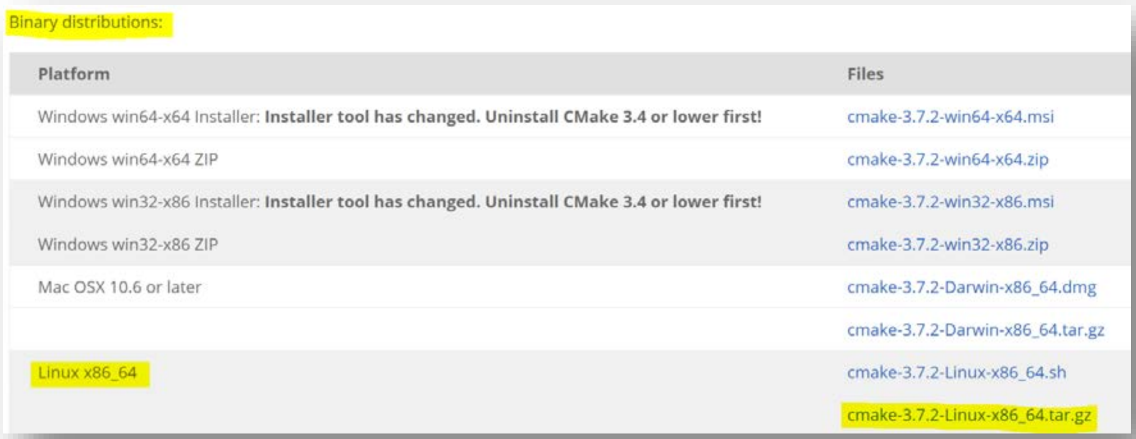

Building and installing

Prereqs:

- CMake v2.6 or better
- A C compiler (GCC 4.4+, clang, and MSVC are test. Other compilers may also work)
- *Optionally* OpenSSL v0.9.8+ to enable support for connecting to RabbitMQ over SSL/TLS
- *Optionally* POpt to build some handy command-line tools.
- *Optionally* XmlTo to build man pages for the handy command-line tools
- *Optionally* Doxygen to build developer API documentation.

On the CMake page, click the **Download** button in the top-right corner to take you to the real download page. Scroll down to the **Latest Release** section. The version you see may be higher than what is shown here in this documentation.

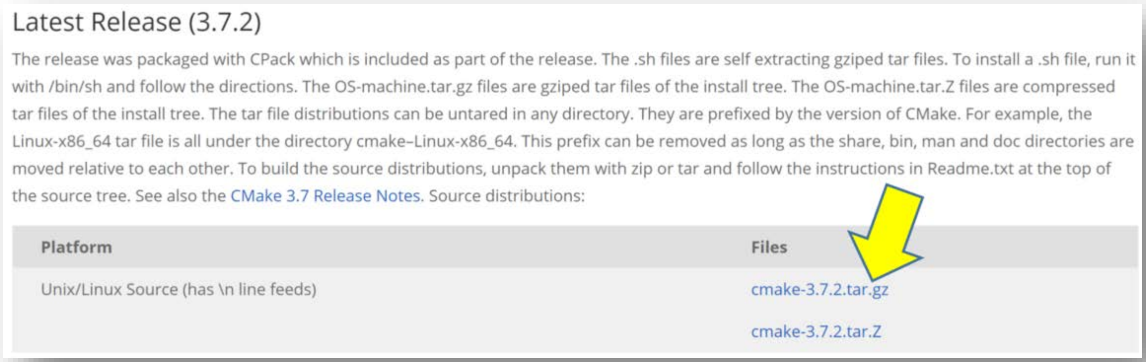
Linux sites on **64-bit** machines should find the **Linux x86_64** line in the **Binary distributions** section and click on the **gz** file to download it.


Binary distributions:

Platform	Files
Windows win64-x64 Installer: **Installer tool has changed. Uninstall CMake 3.4 or lower first!**	cmake-3.7.2-win64-x64.msi
Windows win64-x64 ZIP	cmake-3.7.2-win64-x64.zip
Windows win32-x86 Installer: **Installer tool has changed. Uninstall CMake 3.4 or lower first!**	cmake-3.7.2-win32-x86.msi
Windows win32-x86 ZIP	cmake-3.7.2-win32-x86.zip
Mac OSX 10.6 or later	cmake-3.7.2-Darwin-x86_64.dmg
	cmake-3.7.2-Darwin-x86_64.tar.gz
Linux x86_64	cmake-3.7.2-Linux-x86_64.sh
	cmake-3.7.2-Linux-x86_64.tar.gz

Linux sites on **32-bit** machines may find the **Linux i386** line in the **Binary distributions** section of a previous version and click on the gz file to download it. If you do not find such section for Linux i386 then you may follow the instructions for non-Linux sites.

Non-Linux sites should download the **Unix/Linux Source** distribution. Click the **gz** link to download the file.


Latest Release (3.7.2)

The release was packaged with CPack which is included as part of the release. The .sh files are self extracting gziped tar files. To install a .sh file, run it with /bin/sh and follow the directions. The OS-machine.tar.gz files are gziped tar files of the install tree. The OS-machine.tar.Z files are compressed tar files of the install tree. The tar file distributions can be untared in any directory. They are prefixed by the version of CMake. For example, the Linux-x86_64 tar file is all under the directory cmake–Linux-x86_64. This prefix can be removed as long as the share, bin, man and doc directories are moved relative to each other. To build the source distributions, unpack them with zip or tar and follow the instructions in Readme.txt at the top of the source tree. See also the CMake 3.7 Release Notes. Source distributions:

Platform	Files
Unix/Linux Source (has \n line feeds)	cmake-3.7.2.tar.gz
	cmake-3.7.2.tar.Z

Linux sites Cmake install instructions

As the root user create a cmake directory under /etc and cd into it:

```
# cd /etc
# mkdir cmake && cd cmake
```

Transfer the cmake gz file you downloaded above to the ***/etc/cmake*** directory and run gunzip and then untar it:

```
# gunzip   cmake-3.7.2-Linux-x86_64.tar.gz
# tar -xvf cmake-3.7.2-Linux-x86_64.tar
# rm       cmake-3.7.2-Linux-x86_64.tar
```

Create a link to the cmake binary in /usr/bin to where cmake was installed:

```
# ln -s /etc/cmake/cmake-3.7.2/bin/cmake /usr/bin/cmake
```

At this point, cmake is installed and can be used in the installation of RabbitMQ.

Non-Linux sites Cmake install instructions

As the root user create a cmake directory under /etc and cd into it:

```
# cd /etc
# mkdir cmake && cd cmake
```

Transfer the cmake gz file you downloaded above to the **/etc/cmake** directory and run gunzip and then untar it:

```
# gunzip   cmake-3.7.2.tar.gz
# tar -xvf cmake-3.7.2.tar
# rm       cmake-3.7.2.tar
```

Place yourself into the newly created directory:

```
# cd cmake-3.7.2
```

These next two steps will take several minutes to complete. Issue the following command:

```
# ./bootstrap
```

HP-UX sites stop here and see the *HP-UX Special Cmake Instructions* below then return here to complete the Cmake installation.

Solaris sites stop here and see the *Solaris Special Cmake Instructions* below then return here to complete the Cmake installation.

AIX sites can continue from here.

After running ./bootstrap, next run the two make commands:

```
# make && make install
```

Create a symbolic link to the cmake binary in /usr/bin to where cmake was built:

```
# ln -s /etc/cmake/cmake-3.7.2-Linux-x86_64/bin/cmake /usr/bin/cmake
```

Confirm that cmake is now in your path by issuing the following command:

```
# which cmake
/usr/bin/cmake
```

The Cmake installation is complete. You may now install the RabbitMQ Client.

HP-UX Special Cmake Instructions

The CMakeCache.txt has a setting that needs to be changed for HP-UX. You will use **sed** to make this change.

Begin by making a backup of the CMakeCache.txt file.

```
# cp CMakeCache.txt CMakeCache.bak
```

Run sed to add the 64-bit compile flag. This will be needed when we run cmake for the RabbitMQ install (the following sed command is one line, do not include a carriage return):

```
# sed 's/CMAKE_C_FLAGS:STRING=/CMAKE_C_FLAGS:STRING=-m64/' CMakeCache.txt >
CMakeCache.new
```

Rename the newly created file to the real txt file that make will use:

```
# mv CMakeCache.new CMakeCache.txt
```

The special change required for HP-UX is complete. Please return to the instructions above to complete the cmake installation.

Solaris Special Cmake Instructions

The CMakeCache.txt has a setting that needs to be changed for Solaris. You will use **sed** to make this change.

Begin by making a backup of the CMakeCache.txt file.

```
# cp CMakeCache.txt CMakeCache.bak
```

Run sed to create a new file based on your version of Solaris. Note that the sed command is one line and does not contain carriage returns:

Solaris 10 uses /usr/**ccs**/lib:

```
# sed 's/CURSES_NCURSES_LIBRARY\:FILEPATH\=CURSES_NCURSES_LIBRARY\-
NOTFOUND/CURSES_NCURSES_LIBRARY\:FILEPATH\=\/usr\/ccs\/lib\/libncurses.so/'
CMakeCache.txt > CMakeCache.new
```

Solaris 11 uses /usr/**gnu**/lib:

```
# sed 's/CURSES_NCURSES_LIBRARY\:FILEPATH\=CURSES_NCURSES_LIBRARY\-
NOTFOUND/CURSES_NCURSES_LIBRARY\:FILEPATH\=\/usr\/gnu\/lib\/libncurses.so/'
CMakeCache.txt > CMakeCache.new
```

Rename the newly created file to the real txt file that make will use:

```
$ mv CMakeCache.new CMakeCache.txt
```

64-bit only: Run another sed command to add the 64-bit compile flag if you are running Solaris on a 64-bit machine. This will be needed when we run cmake for the RabbitMQ install:

```
# sed 's/CMAKE_C_FLAGS:STRING=/CMAKE_C_FLAGS:STRING=-m64/' CMakeCache.txt >
CMakeCache.new
```

Rename the newly created file to the real txt file that make will use:

```
$ mv CMakeCache.new CMakeCache.txt
```

The special change required for Solaris is complete. Please return to the instructions above to complete the Cmake installation.

Installing the RabbitMQ Client Software

As the root user transfer the **rabbitmq-c-master.zip** file you downloaded earlier directly into the **/opt** directory. When the file is unzipped you will haven a **/opt/rabbitmq-c-master** directory. This is where the Degree Works $LIBS and $INCLUDES environment variables will point.

Unzip the zip file in the /opt directory.

```
# cd /opt
# unzip rabbitmq-c-master.zip
# rm rabbitmq-c-master.zip
```

In the rabbitmq-c-master directory create a build directory and place yourself there:

```
# cd rabbitmq-c-master
# mkdir build && cd build
```

All clients: Run cmake pointing to the parent directory. This will take several minutes.

```
# cmake ..
```

AIX sites stop here – please see the *AIX Special RabbitMQ Client instructons* below before proceeding with the final step.

HP-UX sites stop here – please see the *HP-UX Special RabbitMQ Client instructons* below before proceeding with the final step.

Final step: run cmake again with the --build option and do not forget the space and dot at the end. This will take several minutes to complete:

```
# cmake --build .
```

Next create a symbolic link to the librabbitmq.so* library file from /usr/local/lib:

```
# ln -s /opt/rabbitmq-c-master/build/librabbitmq/librabbitmq.so.4
/usr/local/lib/librabbitmq.so.4
```

Congratulations, you are now done with the RabbitMQ Client installation.

AIX Special RabbitMQ Client Instructions

On AIX the C++ style comments cause the **build** step to fail so we will remove these comments using **sed**. Note that the sed command below is one line and should not include a carriage return. Begin by making a backup of the amqp_openssl_hostname_validation.c file.

```
# cd ../librabbitmq
# mv amqp_openssl_hostname_validation.c amqp_openssl_hostname_validation.bak
# sed -e 's/ \/\/\( .*\)$/\/*\1*\//g' amqp_openssl_hostname_validation.bak >
amqp_openssl_hostname_validation.c
```

Next modify /opt/rabbitmq-c-master/**CMakeLists.txt** – use the shell and **perl** commands below to add this **set** command before the “if (MSVC)” line around line 130. This is being shown to you here so you can understand the change being made.

```
set(CMAKE_C_FLAGS "-q64 ${CMAKE_C_FLAGS}")
if (MSVC)
  set(CMAKE_C_FLAGS "/W4 /nologo ${CMAKE_C_FLAGS}")
elseif (CMAKE_C_COMPILER_ID MATCHES ".*Clang")
  set(CMAKE_C_FLAGS "-Wall -Wextra -Wstrict-prototypes -Wno-unused-function -fno-common -
fvisibility=hidden ${CMAKE_C_FLAGS}")
elseif (CMAKE_COMPILER_IS_GNUCC)
  set(RMQ_C_FLAGS "-Wall -Wextra -Wstrict-prototypes -Wno-unused-function -fno-common")
  execute_process(COMMAND ${CMAKE_C_COMPILER} -dumpversion OUTPUT_VARIABLE GCC_VERSION)
  if (GCC_VERSION VERSION_GREATER 4.0 OR GCC_VERSION VERSION_EQUAL 4.0)
      set(RMQ_C_FLAGS "${RMQ_C_FLAGS} -fvisibility=hidden")
  endif()
  set(CMAKE_C_FLAGS "${RMQ_C_FLAGS} ${CMAKE_C_FLAGS}")
endif ()
```

Use **perl** to add the set line before the “if (MSVC)” line in CMakeLists.txt:

```
# cd /opt/rabbitmq-c-master
# cp CMakeLists.txt CMakeLists.bak
# perl -plne 'print "set\(CMAKE_C_FLAGS \"-q64 \$\{CMAKE_C_FLAGS\}\"\)" if(/if
\(MSVC\)/);' \
        CMakeLists.txt > CMakeLists.new
# mv CMakeLists.new CMakeLists.txt
```

The AIX Special instructions are now complete. Change directories back into the build directory then return to the main instructions above to run the final cmake command.

```
# cd /opt/rabbitmq-c-master/build
```

HP-UX Special RabbitMQ Client Instructions

If you are building and running Degree Works in 64-bit mode, you will need to make the following change to build/CMakeCache.txt on or around line 52:

Change

```
CMAKE_C_FLAGS:STRING=
```

to

```
CMAKE_C_FLAGS:STRING=+DD64
```

Adding the “+DD64”.

On HP-UX a compile warning will display and an error will result when attempting to connect to the RabbitMQ server. We will correct the issue by making a change to a C source file using sed. Begin by making a backup of the amqp_socket.c file.

```
# cd ../librabbitmq
# mv amqp_socket.c amqp_socket.c.bak
# sed 's/socklen_t result_len/int result_len/' amqp_socket.c.bak >
amqp_socket.c
```

With HP-UX you may notice the following compile warning which can be ignored.

```
/opt/rabbitmq-c-master/librabbitmq/amqp_socket.c, line 302: warning #2111-D:
    statement is unreachable
    return AMQP_STATUS_OK;
```

The HP-UX Special instructions are now complete. Change directories back into the build directory then return to the main instructions above to run the final cmake command.

```
# cd /opt/rabbitmq-c-master/build
```