# queue: Queue Server and Access Method Using Recio Version 1

## CML00082-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com
Copyright © 2014 by Code Magus Limited
All rights reserved


August 16, 2016

# Contents

**1 Introduction** **2**

**2 The queue access method** **3**
2.1 Open String Specification . . . . . 3
2.2 Open Specification Access Method Name . . . . . 3
2.3 Open Specification Object Name . . . . . 3
2.4 Open Specification Option Name-Value Pairs . . . . . 3
2.5 Open String Specification Examples . . . . . 3
2.6 queue access method definition file . . . . . 4
2.7 Environment . . . . . 6

**3 The queue server** **8**
3.1 Introduction . . . . . 8
3.2 Starting the Queue Server . . . . . 8
3.2.1 Queue Server Start up Parameters . . . . . 8
3.3 Server Processing . . . . . 9
3.4 Stopping the Queue Server . . . . . 9

# 1 Introduction

The `queue` access method is a module which implements the `recio` [1] provider interface allowing the `recio` [1] user interface to support en-queueing and de-queueing records to and from a `queue` server on a remote platform.

The `queue` access method effectively supplies two access methods; the first, `ENQUEUE` for en-queueing (writing) records on to the queue; and the second, `DEQUEUE` for de-queueing (reading) records off the queue.

The queue is implemented as a first in first out (FIFO) queue and the queue position is maintained from one session to the next. Any number of clients may connect to the server and enqueue records simultaneously. Only one client at a time may connect to the server to dequeue records. There are various actions that may be taken when a client requests to dequeue a record and there is no record available (the queue is effectively empty), some of which may violate the first in first out dequeue order of records.

The `queue` access method is explained in section 2 on page 3.‘

The `queue` server is a daemon program that implements the queue on a server and accepts and processes requests to enqueue and dequeue records to and from the different queues. The `queue` server is explained in section 3 on page 8.

# 2 The queue access method

## 2.1 Open String Specification

As with all `recio` [1] library open specification strings, three components comprise the open string: access method, object, and options name-value pairs.

## 2.2 Open Specification Access Method Name

The access method name should be specified as en`queue` when en-queueing records and de`queue` when de-queueing records.

## 2.3 Open Specification Object Name

The object name is the queue name as it is known on the remote host.

## 2.4 Open Specification Option Name-Value Pairs

Consult the access method definition file for the option name-value pairs and their description supported by the `queue` access method. The access method definition file also supplies details of the default values (if any) of the options.

## 2.5 Open String Specification Examples

The following open string specifications could be used to

- enqueue a record to the queue "people".

  ```
  enqueue(people,host=codemagus.it.nednet.co.za)
  ```

- dequeue a record from the queue "people" and if a record is not available return an end of file condition.

  ```
  dequeue(people,host=codemagus.it.nednet.co.za,empty_queue=eof)
  ```

- dequeue a record from the queue "people" and if a record is not available to wait until one is available.

  ```
  dequeue(people,host=codemagus.it.nednet.co.za,empty_queue=block)
  ```

- dequeue a record from the queue "people" and if a record is not available to return a zero length record.

  ```
  dequeue(people,host=codemagus.it.nednet.co.za,empty_queue=zerolength)
  ```

- dequeue a record from the queue "people" and if a record is not available to restart the queue as though all the previous records have been instantly re-enqueued. If the queue has never had records queued to it then the request is blocked until a record is queued to the queue.

```
dequeue(people,host=codemagus.it.nednet.co.za,empty_queue=restart)
```

## 2.6 queue access method definition file

The access method definition file should be consulted for the description of the options and their default values. This includes the description of the options. The access method definition file should also be consulted for the processing modes supported by the access method.

Refer the `recio` [1] library documentation for interpreting the contents of the access method definition file.

```
access enqueue(host,port=60070);

  -- File: ENQUEUE.amd
  --
  -- This file contains an access method definition which is used to write
  -- (enqueue) records on to a FIFO queue so that they will be available for
  -- retrieval at a future later date.
  -- The service is supplied by a remote server which must be identified in the
  -- parameters of the recio open string.
  --
  -- Author: Code Magus Limited [www.codemagus.com].
  --
  -- Copyright (c) 2011 Code Magus Limited. All rights reserved.

  -- $Author: hayward $
  -- $Date: 2011/03/17 17:28:02 $
  -- $Id: ENQUEUE.amd,v 1.1.1.1 2011/03/17 17:28:02 hayward Exp $
  -- $Name:  $
  -- $Revision: 1.1.1.1 $
  -- $State: Exp $
  --
  -- $Log: ENQUEUE.amd,v $
  -- Revision 1.1.1.1  2011/03/17 17:28:02  hayward
  -- Import to CVS.
  --

   modes seq_input, seq_output;

   implements open;
   implements close;
   implements write;

   describe host as
```

```
      "The remote host to connect to. "
      "This may be either a host name or an IP address. "
      ;

   describe port as
      "The port on the remote host to connect to."
      ;

   -- Host can be an IP number nnn.nnn.nnn.nnn (e.g. 10.168.1.123)
   -- OR a host name (e.g. www.codemagus.com)
   constrain host as
       "^\(\(\([0-9]\{1,3\}\)\(\.[0-9]\{1,3\}\)\{3\}\)"
       "\|\(\([^. ]\+\)\(\.[^. ]\+\)*\)\)$";

   -- Port must be an integer number.
   constrain port as "^[0-9][0-9]*$";

   path = ${CODEMAGUS_AMDLIBS} "%s";
   module = "queueam" ${CODEMAGUS_AMDSUFDL};
   entry = queueam_init;

end.

access dequeue(host,port=60070,empty_action='eof');

  -- File: DEQUEUE.amd
  --
  -- This file contains an access method definition which is used to read
  -- (dequeue) records from a FIFO queue queued there by STORE.amd in the
  -- original order.
  -- The service is supplied by a remote server which must be identified in the
  -- parameters of the recio open string.
  --
  -- Author: Code Magus Limited [www.codemagus.com].
  --
  -- Copyright (c) 2011 Code Magus Limited. All rights reserved.

  -- $Author: hayward $
  -- $Date: 2011/03/17 17:28:02 $
  -- $Id: DEQUEUE.amd,v 1.1.1.1 2011/03/17 17:28:02 hayward Exp $
  -- $Name:  $
  -- $Revision: 1.1.1.1 $
  -- $State: Exp $
  --
  -- $Log: DEQUEUE.amd,v $
  -- Revision 1.1.1.1  2011/03/17 17:28:02  hayward
  -- Import to CVS.
  --

   modes seq_input, seq_output;

   implements open;
   implements close;
```

```
   implements read;

   describe host as
      "The remote host to connect to. "
      "This may be either a host name or an IP address. "
      ;

   describe port as
      "The port on the remote host to connect to."
      ;

   describe empty_action as
      "The action to take when there are no more records to dequeue "
      "(the queue is empty). "
      "The actions are: 'eof' meaning return and end of file indicator; "
      "'block' meaning wait until a record is available; "
      "'zerolength' meaning return a zero length record; "
      "'restart' meaning start at the beginning of the queue again."
      ;

   -- Host can be an IP number nnn.nnn.nnn.nnn (e.g. 10.168.1.123)
   -- OR a host name (e.g. www.codemagus.com)
   constrain host as
       "^\(\(\([0-9]\{1,3\}\)\(\.[0-9]\{1,3\}\)\{3\}\)"
       "\|\(\([^. ]\+\)\(\.[^. ]\+\)*\)\)$";

   -- Port must be an integer number.
   constrain port as "^[0-9][0-9]*$";

   -- empty_action constrained as per the description.
   constrain empty_action as
      "^\(eof\)\|\(block\)\|\(zerolength\)\|\(restart\)$";

   path = ${CODEMAGUS_AMDLIBS} "%s";
   module = "queueam" ${CODEMAGUS_AMDSUFDL};
   entry = queueam_init;

end.
```

## 2.7 Environment

The location and format of the access method definition file is required to be specified by the environment variable `CODEMAGUS_AMDPATH`. This environment variable supplies a pattern to the full path of where access method definition (or `amd`) files are located. The format of the environment variable is that of a path with a %s appearing in the position in which the access method member name should appear. For example, on MVS systems this might have the form:

```
CODEMAGUS_AMDPATH='DNCT00.SRDA1.AMDFILES(%s)'
```

On a Unix-based system, the value might be set in a shell profile file such as:

```
export CODEMAGUS_AMDPATH=$HOME/bin/%s.amd
```

On Windows systems, the value might be supplied from the environment variables and look something like:

```
C:\CodeMagus\bin\%s.amd
```

# 3 The queue server

## 3.1 Introduction

The queue server is a program started on the server machine that waits for incoming connection requests from a queue access method and services the request.

## 3.2 Starting the Queue Server

The Queue Server program cmlqueued is started from the command line and by specifying the --help parameter the server will print out the parameters available.

```
Code Magus Limited Queue Server V1.0: build 2011-03-17-17.53.22
[15193]# [cmlqueued] $Id: cmlqueued.c,v 1.1.1.1 2011/03/17 17:28:03 hayward Exp $
# Copyright (c) 2011 by Code Magus Limited. All rights reserved.
# [helpinfo@codemagus.com].
Usage: cmlqueued [OPTION...]
  -h, --host={any|<host name>|<IP address>}     host or IP address to listen on
  -p, --port={60070|<port number>}              Port number to accept
                                                connections on
  -r, --root={.|<directory>}                    Base directory for queue files
  -l, --log-file-mask={none|<file mask>}        Log file mask (%d=PID,
                                                %s=program name)
  -b, --buffer-size={131072|<integer>}          Network buffer size in bytes
  -v, --verbose                                 Verbose processing mode

Help options:
  -?, --help                                    Show this help message
  --usage                                       Display brief usage message
```

### 3.2.1 Queue Server Start up Parameters

The start up parameters are as follows:

- -h, --host={any|<host name>|<IP address>}
  This optional parameter will limit the host or IP address that the server will listen on. It can be used to restrict incoming connections to only one interface. If not specified, the server will listen on all interfaces.
- -p, --port={60070|<port number>}
  This optional parameter specifies the port number that the server will accept connections on. If not specified, port 60070 is used.
- -r, --root={.|<directory>}
  Base directory for queue and queue control files. This must be an existing directory.

- `-b, --buffer-size={131072|<integer>}`
  Network buffer size in bytes. Any records larger than this will cause an error message to be returned.
- `-l, --log-file-mask=<file mask>`
  This optional parameter specifies the location of the log file for any program processing output (such as verbose messages). It should include the substitution variables `%d` which is replaced with the process ID and `%s` which is replaced by the program name of the `queue` server. The log is written under the control of `rprintf` [2].
- `-v, --verbose`
  This optional parameter will cause the `queue` server to produce verbose information about the processing flow of the program.

## 3.3 Server Processing

The queue name as supplied by the `queue` access method is used to generate the queue file name in which the records are stored and the queue control file name which holds state information about the queue. The queue files are named by appending `.q` to the queue name and the queue control files are named by appending `.qctl` to the queue name. Both the queue files and control files exist in the directory named in the server start up parameters.

## 3.4 Stopping the Queue Server

Before the server can be stopped all clients must be disconnected. To stop the server send a `SIGINT` signal to it or if it is running in the foreground use `ctl-c`.

On Linux and Unix platforms to send a signal to a process first determine the process id (PID) of the process and then use the `kill` command.

```
ps -ef | grep cmlqueued
kill -sigint 12345
```

# References

[1] recio: Record Stream I/O Library Version 1. CML Document CML00001-01, Code Magus Limited, July 2008. PDF.

[2] rprintf API: User Guide and Reference Version 1. CML Document CML00002-01, Code Magus Limited, July 2008. PDF.