#

# From oslo.messaging

#

# Use durable queues in AMQP. (boolean value)

# Deprecated group/name - [DEFAULT]/amqp\_durable\_queues

# Deprecated group/name - [DEFAULT]/rabbit\_durable\_queues

#amqp\_durable\_queues = false

# Auto-delete queues in AMQP. (boolean value)

#amqp\_auto\_delete = false

# Enable SSL (boolean value)

#ssl = <None>

{%- if \_data.get('ssl',{}).get('enabled', False) %}

ssl = true

# SSL version to use (valid only if SSL enabled). Valid values are

# TLSv1 and SSLv23. SSLv2, SSLv3, TLSv1\_1, and TLSv1\_2 may be

# available on some distributions. (string value)

# Deprecated group/name - [oslo\_messaging\_rabbit]/kombu\_ssl\_version

#ssl\_version =

{%- if \_data.ssl.version is defined %}

ssl\_version = {{ \_data.ssl.version }}

{%- elif salt['grains.get']('pythonversion') > [2,7,8] %}

ssl\_version = TLSv1\_2

{%- endif %}

{%- if \_data.get('x509',{}).get('enabled', False) %}

# SSL key file (valid only if SSL enabled). (string value)

# Deprecated group/name - [oslo\_messaging\_rabbit]/kombu\_ssl\_keyfile

#ssl\_key\_file =

ssl\_key\_file = {{ \_data.x509.key\_file }}

# SSL cert file (valid only if SSL enabled). (string value)

# Deprecated group/name - [oslo\_messaging\_rabbit]/kombu\_ssl\_certfile

#ssl\_cert\_file =

ssl\_cert\_file = {{ \_data.x509.cert\_file }}

# SSL certification authority file (valid only if SSL enabled).

# (string value)

# Deprecated group/name - [oslo\_messaging\_rabbit]/kombu\_ssl\_ca\_certs

#ssl\_ca\_file =

ssl\_ca\_file = {{ \_data.x509.ca\_file }}

{%- else %}

ssl\_ca\_file = {{ \_data.ssl.cacert\_file }}

{%- endif %}

{%- endif %}

# How long to wait before reconnecting in response to an AMQP consumer

# cancel notification. (floating point value)

#kombu\_reconnect\_delay = 1.0

# EXPERIMENTAL: Possible values are: gzip, bz2. If not set compression

# will not be used. This option may not be available in future

# versions. (string value)

#kombu\_compression = <None>

# How long to wait a missing client before abandoning to send it its

# replies. This value should not be longer than rpc\_response\_timeout.

# (integer value)

# Deprecated group/name - [oslo\_messaging\_rabbit]/kombu\_reconnect\_timeout

#kombu\_missing\_consumer\_retry\_timeout = 60

# Determines how the next RabbitMQ node is chosen in case the one we

# are currently connected to becomes unavailable. Takes effect only if

# more than one RabbitMQ node is provided in config. (string value)

# Possible values:

# round-robin - <No description provided>

# shuffle - <No description provided>

#kombu\_failover\_strategy = round-robin

# DEPRECATED: The RabbitMQ broker address where a single node is used.

# (string value)

# This option is deprecated for removal.

# Its value may be silently ignored in the future.

# Reason: Replaced by [DEFAULT]/transport\_url

#rabbit\_host = localhost

# DEPRECATED: The RabbitMQ broker port where a single node is used.

# (port value)

# Minimum value: 0

# Maximum value: 65535

# This option is deprecated for removal.

# Its value may be silently ignored in the future.

# Reason: Replaced by [DEFAULT]/transport\_url

#rabbit\_port = 5672

# DEPRECATED: RabbitMQ HA cluster host:port pairs. (list value)

# This option is deprecated for removal.

# Its value may be silently ignored in the future.

# Reason: Replaced by [DEFAULT]/transport\_url

#rabbit\_hosts = $rabbit\_host:$rabbit\_port

# DEPRECATED: The RabbitMQ userid. (string value)

# This option is deprecated for removal.

# Its value may be silently ignored in the future.

# Reason: Replaced by [DEFAULT]/transport\_url

#rabbit\_userid = guest

# DEPRECATED: The RabbitMQ password. (string value)

# This option is deprecated for removal.

# Its value may be silently ignored in the future.

# Reason: Replaced by [DEFAULT]/transport\_url

#rabbit\_password = guest

# The RabbitMQ login method. (string value)

# Possible values:

# PLAIN - <No description provided>

# AMQPLAIN - <No description provided>

# RABBIT-CR-DEMO - <No description provided>

#rabbit\_login\_method = AMQPLAIN

# DEPRECATED: The RabbitMQ virtual host. (string value)

# This option is deprecated for removal.

# Its value may be silently ignored in the future.

# Reason: Replaced by [DEFAULT]/transport\_url

#rabbit\_virtual\_host = /

# How frequently to retry connecting with RabbitMQ. (integer value)

#rabbit\_retry\_interval = 1

{%- if \_data.rabbit\_retry\_interval is defined %}

rabbit\_retry\_interval = {{ \_data.rabbit\_retry\_interval }}

{%- endif %}

# How long to backoff for between retries when connecting to RabbitMQ.

# (integer value)

#rabbit\_retry\_backoff = 2

{%- if \_data.rabbit\_retry\_backoff is defined %}

rabbit\_retry\_backoff = {{ \_data.rabbit\_retry\_backoff }}

{%- endif %}

# Maximum interval of RabbitMQ connection retries. Default is 30

# seconds. (integer value)

#rabbit\_interval\_max = 30

{%- if \_data.rabbit\_interval\_max is defined %}

rabbit\_interval\_max = {{ \_data.rabbit\_interval\_max }}

{%- endif %}

# DEPRECATED: Maximum number of RabbitMQ connection retries. Default

# is 0 (infinite retry count). (integer value)

# This option is deprecated for removal.

# Its value may be silently ignored in the future.

#rabbit\_max\_retries = 0

# Try to use HA queues in RabbitMQ (x-ha-policy: all). If you change

# this option, you must wipe the RabbitMQ database. In RabbitMQ 3.0,

# queue mirroring is no longer controlled by the x-ha-policy argument

# when declaring a queue. If you just want to make sure that all

# queues (except those with auto-generated names) are mirrored across

# all nodes, run: "rabbitmqctl set\_policy HA '^(?!amq\.).\*' '{"ha-

# mode": "all"}' " (boolean value)

#rabbit\_ha\_queues = false

{%- if \_data.rabbit\_ha\_queues is defined %}

rabbit\_ha\_queues = {{ \_data.rabbit\_ha\_queues }}

{%- endif %}

# Positive integer representing duration in seconds for queue TTL

# (x-expires). Queues which are unused for the duration of the TTL are

# automatically deleted. The parameter affects only reply and fanout

# queues. (integer value)

# Minimum value: 1

#rabbit\_transient\_queues\_ttl = 1800

{%- if \_data.rabbit\_transient\_queues\_ttl is defined %}

rabbit\_transient\_queues\_ttl = {{ \_data.rabbit\_transient\_queues\_ttl }}

{%- endif %}

# Specifies the number of messages to prefetch. Setting to zero allows

# unlimited messages. (integer value)

# NOTE(dmescheryakov) hardcoding to >0 by default

# Having no prefetch limit makes oslo.messaging consume all available

# messages from the queue. That can lead to a situation when several

# server processes hog all the messages leaving others out of business.

# That leads to artificial high message processing latency and at the

# extrime to MessagingTimeout errors.

rabbit\_qos\_prefetch\_count = {{ \_data.get('rabbit\_qos\_prefetch\_count', 64) }}

# Number of seconds after which the Rabbit broker is considered down

# if heartbeat's keep-alive fails (0 disable the heartbeat).

# EXPERIMENTAL (integer value)

#heartbeat\_timeout\_threshold = 60

{%- if \_data.get('heartbeat\_timeout\_threshold', 0) %}

heartbeat\_timeout\_threshold = {{ \_data.get('heartbeat\_timeout\_threshold', 0) }}

{%- endif %}

# How often times during the heartbeat\_timeout\_threshold we check the

# heartbeat. (integer value)

#heartbeat\_rate = 2

{%- if \_data.heartbeat\_rate is defined %}

heartbeat\_rate = {{ \_data.heartbeat\_rate }}

{%- endif %}

# Deprecated, use rpc\_backend=kombu+memory or rpc\_backend=fake

# (boolean value)

#fake\_rabbit = false

# Maximum number of channels to allow (integer value)

#channel\_max = <None>

{%- if \_data.channel\_max is defined %}

channel\_max = {{ \_data.channel\_max }}

{%- endif %}

# The maximum byte size for an AMQP frame (integer value)

#frame\_max = <None>

{%- if \_data.frame\_max is defined %}

frame\_max = {{ \_data.frame\_max }}

{%- endif %}

# How often to send heartbeats for consumer's connections (integer

# value)

#heartbeat\_interval = 3

{%- if \_data.heartbeat\_interval is defined %}

heartbeat\_interval = {{ \_data.heartbeat\_interval }}

{%- endif %}

# Arguments passed to ssl.wrap\_socket (dict value)

#ssl\_options = <None>

# Set socket timeout in seconds for connection's socket (floating

# point value)

#socket\_timeout = 0.25

{%- if \_data.socket\_timeout is defined %}

socket\_timeout = {{ \_data.socket\_timeout }}

{%- endif %}

# Set TCP\_USER\_TIMEOUT in seconds for connection's socket (floating

# point value)

#tcp\_user\_timeout = 0.25

{%- if \_data.tcp\_user\_timeout is defined %}

tcp\_user\_timeout = {{ \_data.tcp\_user\_timeout }}

{%- endif %}

# Set delay for reconnection to some host which has connection error

# (floating point value)

#host\_connection\_reconnect\_delay = 0.25

{%- if \_data.host\_connection\_reconnect\_delay is defined %}

host\_connection\_reconnect\_delay = {{ \_data.host\_connection\_reconnect\_delay }}

{%- endif %}

# Connection factory implementation (string value)

# Possible values:

# new - <No description provided>

# single - <No description provided>

# read\_write - <No description provided>

#connection\_factory = single

{%- if \_data.connection\_factory is defined %}

connection\_factory = {{ \_data.connection\_factory }}

{%- endif %}

# Maximum number of connections to keep queued. (integer value)

#pool\_max\_size = 30

{%- if \_data.pool\_max\_size is defined %}

pool\_max\_size = {{ \_data.pool\_max\_size }}

{%- endif %}

# Maximum number of connections to create above `pool\_max\_size`.

# (integer value)

#pool\_max\_overflow = 0

{%- if \_data.pool\_max\_overflow is defined %}

pool\_max\_overflow = {{ \_data.pool\_max\_overflow }}

{%- endif %}

# Default number of seconds to wait for a connections to available

# (integer value)

#pool\_timeout = 30

{%- if \_data.pool\_timeout is defined %}

pool\_timeout = {{ \_data.pool\_timeout }}

{%- endif %}

# Lifetime of a connection (since creation) in seconds or None for no

# recycling. Expired connections are closed on acquire. (integer

# value)

#pool\_recycle = 600

{%- if \_data.pool\_recycle is defined %}

pool\_recycle = {{ \_data.pool\_recycle }}

{%- endif %}

# Threshold at which inactive (since release) connections are

# considered stale in seconds or None for no staleness. Stale

# connections are closed on acquire. (integer value)

#pool\_stale = 60

{%- if \_data.pool\_stale is defined %}

pool\_stale = {{ \_data.pool\_stale }}

{%- endif %}

# Default serialization mechanism for serializing/deserializing

# outgoing/incoming messages (string value)

# Possible values:

# json - <No description provided>

# msgpack - <No description provided>

#default\_serializer\_type = json

{%- if \_data.default\_serializer\_type is defined %}

default\_serializer\_type = {{ \_data.default\_serializer\_type }}

{%- endif %}

# Persist notification messages. (boolean value)

#notification\_persistence = false

{%- if \_data.notification\_persistence is defined %}

notification\_persistence = {{ \_data.notification\_persistence }}

{%- endif %}

# Exchange name for sending notifications (string value)

#default\_notification\_exchange = ${control\_exchange}\_notification

{%- if \_data.default\_notification\_exchange is defined %}

default\_notification\_exchange = {{ \_data.default\_notification\_exchange }}

{%- endif %}

# Max number of not acknowledged message which RabbitMQ can send to

# notification listener. (integer value)

#notification\_listener\_prefetch\_count = 100

{%- if \_data.notification\_listener\_prefetch\_count is defined %}

notification\_listener\_prefetch\_count = {{ \_data.notification\_listener\_prefetch\_count }}

{%- endif %}

# Reconnecting retry count in case of connectivity problem during

# sending notification, -1 means infinite retry. (integer value)

#default\_notification\_retry\_attempts = -1

{%- if \_data.default\_notification\_retry\_attempts is defined %}

default\_notification\_retry\_attempts = {{ \_data.default\_notification\_retry\_attempts }}

{%- endif %}

# Reconnecting retry delay in case of connectivity problem during

# sending notification message (floating point value)

#notification\_retry\_delay = 0.25

{%- if \_data.notification\_retry\_delay is defined %}

notification\_retry\_delay = {{ \_data.notification\_retry\_delay }}

{%- endif %}

# Time to live for rpc queues without consumers in seconds. (integer

# value)

#rpc\_queue\_expiration = 60

{%- if \_data.rpc\_queue\_expiration is defined %}

rpc\_queue\_expiration = {{ \_data.rpc\_queue\_expiration }}

{%- endif %}

# Exchange name for sending RPC messages (string value)

#default\_rpc\_exchange = ${control\_exchange}\_rpc

{%- if \_data.default\_rpc\_exchange is defined %}

default\_rpc\_exchange = {{ \_data.default\_rpc\_exchange }}

{%- endif %}

# Exchange name for receiving RPC replies (string value)

#rpc\_reply\_exchange = ${control\_exchange}\_rpc\_reply

{%- if \_data.rpc\_reply\_exchange is defined %}

rpc\_reply\_exchange = {{ \_data.rpc\_reply\_exchange }}

{%- endif %}

# Max number of not acknowledged message which RabbitMQ can send to

# rpc listener. (integer value)

#rpc\_listener\_prefetch\_count = 100

{%- if \_data.rpc\_listener\_prefetch\_count is defined %}

rpc\_listener\_prefetch\_count = {{ \_data.rpc\_listener\_prefetch\_count }}

{%- endif %}

# Max number of not acknowledged message which RabbitMQ can send to

# rpc reply listener. (integer value)

#rpc\_reply\_listener\_prefetch\_count = 100

{%- if \_data.rpc\_reply\_listener\_prefetch\_count is defined %}

rpc\_reply\_listener\_prefetch\_count = {{ \_data.rpc\_reply\_listener\_prefetch\_count }}

{%- endif %}

# Reconnecting retry count in case of connectivity problem during

# sending reply. -1 means infinite retry during rpc\_timeout (integer

# value)

#rpc\_reply\_retry\_attempts = -1

{%- if \_data.rpc\_reply\_retry\_attempts is defined %}

rpc\_reply\_retry\_attempts = {{ \_data.rpc\_reply\_retry\_attempts }}

{%- endif %}

# Reconnecting retry delay in case of connectivity problem during

# sending reply. (floating point value)

#rpc\_reply\_retry\_delay = 0.25

{%- if \_data.rpc\_reply\_retry\_delay is defined %}

rpc\_reply\_retry\_delay = {{ \_data.rpc\_reply\_retry\_delay }}

{%- endif %}

# Reconnecting retry count in case of connectivity problem during

# sending RPC message, -1 means infinite retry. If actual retry

# attempts in not 0 the rpc request could be processed more than one

# time (integer value)

#default\_rpc\_retry\_attempts = -1

{%- if \_data.default\_rpc\_retry\_attempts is defined %}

default\_rpc\_retry\_attempts = {{ \_data.default\_rpc\_retry\_attempts }}

{%- endif %}

# Reconnecting retry delay in case of connectivity problem during

# sending RPC message (floating point value)

#rpc\_retry\_delay = 0.25

{%- if \_data.rpc\_retry\_delay is defined %}

rpc\_retry\_delay = {{ \_data.rpc\_retry\_delay }}

{%- endif %}