# New quorum features in Corosync 2

| | | | |
|---|---|---|---|
| 0.1 | Initial version | Christine Caulfield | April 2012 |
| 0.2 | Added appendix of libcman replacements | Christine Caulfield | February 2013 |
| 0.3 | Added mention of new quorum options and some formatting changes | Christine Caulfield | June 2014 |
| 0.4 | Clarify fence delay | Christine Caulfield | May 2015 |
| 0.5 | Update some ATB detail | Christine Caulfield | July 2015 |
| 0.6 | Fix wait_for_all_status doc | Christine Caulfield | January 2016 |

## Introduction

This document describes the new quorum system in corosync for Corosync 2. It updates a lot of the content of my previous document “Whither cman” which should now be considered obsolete. I will assume a good knowledge of cman and openais/corosync from RHEL-5 and RHEL-6 – this is not an introduction to clustering. If you want one of those then may I gently recommend the documentation provided by proper authors who can actually write documentation. The purpose of this little document is to highlight the differences and help describe the new features that are available. It is aimed at support staff and people who want a more technical overview of the systems.

With the release of corosync 2.0.0 “Needle”, quorum majority voting is an integral part of the daemon, and not an add-in as cman was in previous versions. This has many implications for how it is configured but also introduces many new features that should help people manage their cluster nodes.

The new module is called votequorum and is not loaded by default into corosync. Without it (or any quorum module) corosync will consider the cluster always quorate by default.

I'll go through the changes at a high level first so you get a feel for what has been done, then I'll delve down into some more detail and show you how the new features can help make a cluster more highly available and, hopefully, easier to manage than before.

votequorum has a manpage votequorum(5) which contains extra detail and also some examples that are not shown here and I recommend you consult it. But I hope this is easier reading to get you started.

## Similarities

votequorum is substantially similar to cman in previous versions in many ways. At its most basic level it is a quorum-based majority voting system where a cluster needs expected_votes/2+1 votes for it to be quorate. This means there should be no nasty surprises, we hope, in upgraded clusters as the basic concepts are retained. The cman two_node mode, which depends on a power fence race, is also still supported for 2 node clusters, as is the quorum device interface for use by qdiskd. Though qdiskd is not available for votequorum at the time of writing (June 2014) the hope is that the additional options will make it unneeded. Read on for more detail...

## Config file changes

The most obvious change with this new system is that we have totally changed the configuration file. Actually we haven't ... in a way. We are now using corosync.conf, which has been standard for corosync in RHEL6 and (as openais.conf) in RHEL5 too but we didn't use it there as all the information was in cluster.conf. But now all of the quorum and node information is also held in corosync.conf. cluster.conf had been kept reasonably consistent since RHEL4 so we hope there won't be too many complaints about it going away now.

Unlike cluster.conf it is not mandatory to list all of the cluster nodes in corosync.conf. It might be helpful for you to do so, for reasons I'll go into later, but we don't insist any more.

## Command-line changes

With the demise of cman there is also the demise of cman's imaginatively command-line tool “cman_tool”. There is now the equally imaginatively named corosync-quorumtool, that shows similar (though a little more) information in a slight different format.

To be honest, most people seemed to prefer using clustat anyway, as that showed rgmanager services too, so we're not expecting a huge public uprising about this. Though, having said that rgmanger is now replaced by pacemaker so there are bigger things to worry about anyway.

## New Quorum Features

Votequorum has some handy new features that make quorum handling a lot more flexible than cman could ever manage, we hope these will mostly remove the need for that pesky qdisk and its complicated timings and obscure heuristics options (what were *they* all about?!).

With these new options things might look even more complicated than before if you're not used to them – and you're not, of course, or you wouldn't be reading this. But I hope that some explanations will help make this seem clearer. And if not, I'm sorry. But it's also always good to experiment with these things.

### *two_node*

OK, this is not a new feature, it's been in cman since the start. But I mention it here out of completeness really as it's still available, still pretty useful. *two_node* is, shockingly, designed for clusters with only two nodes where you, not unreasonably, want one node to continue working if the other fails. This mode requires that *expected_votes* is set to 2 and that you have hardware fencing configured and connected over the same network interface as the cluster heartbeat. The way it works is that in the event of a network outage both nodes race in an attempt to fence each other and the first to succeed continues in the cluster. The system administrator can also associate a delay with a fencing agent so that one node can be given priority in this situation so that it always wins the race. This is the simplest setup for a twin-node cluster.

Enabling two_node also, by default, enables the following option *wait_for_all*...

### *wait_for_all*

This is the first of the new options. When starting up a cluster from scratch (ie all nodes down or, at least, not part of the cluster) it will prevent the cluster from becoming quorate until all of the nodes have joined in. It does this by comparing the number of active votes in the cluster with the value of expected_votes. It's important to realise that it does not use the nodes list for this even if it exists.

Without *wait_for_all*, the normal behaviour of a cluster is for quorum to be enabled as soon as the required number of votes is achieved. *wait_for_all* is a useful way of booting up a cluster and making sure that it is not partitioned at startup. In the *two_node* case this is very important. Though you can disable *wait_for_all* in a *two_node* cluster, it is not recommended.

Note that once all of the nodes have been seen, normal quorum behaviour resumes. Nodes can join and leave the cluster and the value of quorum will be honoured as normal. *wait_for_all* only affects the initial startup of a cluster from no nodes active to all nodes active. In a cluster that is larger than two nodes this might seem like a strange thing to enable. Bear with me, all will become clear, I hope, in the next few sections.

### *auto_tie_breaker*

Auto Tie Breaker allows the cluster to continue working in the event that a cluster containing an even number of nodes is split in half. Under the old quorum system a cluster needed n/2+1 nodes to continue working and a 50/50 split of nodes could result in a split-brain that is not recoverable without manual intervention or, (hushed whisper) qdiskd. This is why reliable clusters are best made with odd numbers. Sorry *were* made with odd numbers, no longer!

This new option tells votequorum that in the event of a 50/50 split of the cluster then the half with the lowest node ID (by default) in it should be deemed the quorate half, and the other half not. If fencing is in operation than the other half will be fenced. The point of this option is to avoid the stand-off that can occur when there is a network outage that splits the cluster into two halves and is the first of the “please sack qdiskd” options that are implemented in votequorum. *auto_tie_breaker* can actually be told which node or list of nodes are used to determine the quorate half of the cluster so you can make sure that an important or especially beefy node or nodes gets priority. See the man page for the gory (and they *are* gory) details.

auto_tie_breaker is not compatible with two_node as both are systems for determining what happens should there be an even split of nodes. If you have both enabled, then an error message will be issued and two_node will be disabled.

If you have a cluster with an odd number of nodes and auto_tie_breaker enabled then it is compulsory to also specify wait_for_all. The reasons for this are complex (aren't they always with clusters) but, roughly, it ensures that if the larger 'half' (no, I am not a mathematician) is down – caused by incremental outages, then it won't gain quorum over the currently active partition when those nodes reboot.

### *allow_downscale*

This new option is in some ways a tidier version of the “cman_tool leave remove” option from olden days. “leave remove” was always a slightly clumsy option, often misunderstood and even more often misused.

Under normal operation the expected_votes value can only ever increase or stay stable. *allow_downscale* will let the value go downwards too, under very specific circumstances. 1) if the node leaves the cluster tidily (most likely due to an ordered shutdown) and 2) that it does not go below the value for expected_votes set in corosync.conf. Point 2 is important here. You cannot set this option and reduce a seven node cluster down to 2 nodes using it, in that case *expected_votes* will remain at its original value of 7 (unless it was overridden to something else manually).

The main purpose of this option is to allow nodes to be added temporarily to a cluster to cover extra workload, they can be safely added as *expected_votes* will be increased by the normal mechanisms and then decreased again as the extra nodes are removed down to the initial setting in corosync.conf.

## *last_man_standing* & *last_man_standing_window*

These two options are probably the most interest to those wanting to get rid of qdiskd and its foibles. This is why I left it till last, cos I'm cruel like that.

*last_man_standing* is the, slightly sexist, name for a system that allows votequorum to reduce the number of expected_votes automatically when nodes leave the cluster. It does this *last_man_standing_window* milliseconds after the nodes leave the cluster. It's important to note that the remaining cluster must be quorate for this calculation to happen. This allows a cluster to be partitioned and the quorate side can be reduced and still stay active. It will also tolerate further node losses as expected votes and quorum will be reduced as if this was the normal running of the cluster.

That sounds complicated and is best illustrated with an example. I make no apology for stealing this 8 node example from the man page votequorum.5

Here's snippet from corosync.conf:

```
quorum {
    provider: corosync_votequorum
    expected_votes: 8
    wait_for_all: 1
    last_man_standing: 1
    last_man_standing_window: 10000
}
```

Example chain of events:

1. The cluster is fully operational with 8 nodes. (expected_votes: 8 quorum: 5)
2. 3 nodes die, cluster is quorate with 5 nodes.
3. After last_man_standing_window timer expires,  expected_votes and quorum are recalculated.  (expected_votes: 5 quorum: 3)
4. At this point, 2 more nodes can die and cluster will still be quorate with 3.
5. Once again, after last_man_standing_window timer expires expected_votes and quorum are recalculated. (expected_votes: 3 quorum: 2)
6. At this point, 1 more node can die and cluster will still be quorate with 2.
7. After one more last_man_standing_window timer (expected_votes: 2 quorum: 2)

It's important to note that the normal operation of *last_man_standing* only allows the cluster to go down to 2 nodes (but *last_two_men_standing* just sounds weird). If you want to go down to running with only 1 node then you also need to set *auto_tie_breaker*. Which you now are an expert in.

**More On The Configuration File**

As I mentioned above the configuration for a cluster is now held in /etc/corosync/corosync.conf and you got a sneak preview of its contents above. All of the new features are enabled in the quorum{} stanza as you can see, and setting the value to 1 enables that option. The default is 0 (disabled). You must load votequorum before it can be used, otherwise corosync will assume that the cluster is ***always*** quorate, regardless of what happens. This is not usually what you want if you come from a normal HA background. So the lines

```
quorum {
  provider: corosync_votequorum
}
```

should be the very least configuration you can get away with. As in cluster.conf you can also specify the nodes you expect to join the cluster, and their (optional) node IDs. These go in the nodelist stanza like this:

```
nodelist {
        node {
                ring0_addr: 192.168.1.101
                nodeid: 1
        }
        node {
                ring0_addr: 192.168.1.102
                nodeid: 2
        }
        node {
                ring0_addr: 192.168.1.103
                nodeid: 3
        }
        node {
                ring0_addr: 192.168.1.104
                nodeid: 4
        }
```

The nodeids are optional for IPv4 but recommended if you want to keep your sanity reading the output of corosync-quorumtool. If you don't then corosync will use the node IP address as the nodeid and they don't look pretty when printed out as a pure number ... not even in hexadecimal. For IPv6, nodeids are compulsory as before. If you are using udpu as the corosync transport you will have to include a nodelist section – but the pcs tool will do this for you, so don't worry too much about it.

When you specify all the nodes like this then, like cman before it, corosync will calculate *expected_votes* by assuming all nodes have a single vote. I really don't recommend giving nodes different votes, it only works well for very specialised cases that almost never apply. If you do want to change the votes for a node from 1, then I'm going to make you look in the man page to find out how to do it. So there :P

I've listed IP addresses in the nodelist above but you can use host names if you want. This is much simplified since cman days – that system just got too messy and too confusing. So many times we have had to refer people to the 4 steps that cman went through as it tried to turn a name into a cluster node ID. So now they are just resolved at corosync startup (or config reload) time using resolver calls. No funny stuff, no fancy parsing, just a call to getaddrinfo(3). We hope this makes things easier, and keeps you off the phone (or email) to us. Note that the names are not stored in corosync at all, so don't look too shocked if they look different to the command-line tools. See ...

**corosync-quorumtool**

corosync-quorumtool is the cman_tool for the new age ... ish. It allows you to query votequorum for the list of nodes, their votes, quorum-status and all those good things that you have become used to. It also allows you to look at the node IDs in hexadecimal, just in case you forgot to assign them in corosync.conf and can't work out which system 827483261 is.

So, here are some examples of corosync-quorumtool to warm you up.

```
[root@amy ~]# corosync-quorumtool -l

Membership information
----------------------
    Nodeid      Votes Name
         1          1 amy.chrissie.net
         2          1 anna.chrissie.net
         3          1 clara.chrissie.net
         4          1 fanny.chrissie.net
1761716416          1 judith.chrissie.net
1778493632          1 kaija.chrissie.net
1795270848          1 nadia.chrissie.net
1812048064          1 sofia.chrissie.net
```

This is the equivalent of 'cman_tool nodes'. I've left four nodes without node IDs in corosync.conf so you can see what sort of node IDs get generated for you.  You will also notice that it has displayed node names, despite them not being in my corosync.conf. corosync-quorumtool resolves the names for you when it is run, they are not stored within corosync at all. So, if your DNS is a bit odd or flighty, you might see different names here some times. The -i switch will force corosync-quorumtool to display IP addresses instead.

```
[root@amy ~]# corosync-quorumtool -siH
Quorum information
------------------
Date:             Mon Apr 23 13:22:33 2012
Quorum provider:  corosync_votequorum
Nodes:            8
Ring ID:          12
Quorate:          Yes

Votequorum information
----------------------
Node ID:          1
Node state:       Member
Node votes:       1
Expected votes:   8
Highest expected: 8
Total votes:      8
Quorum:           5
Flags:            Quorate

Membership information
----------------------
    Nodeid      Votes Name
0x00000001          1 192.168.1.101
```

```
0x00000002          1 192.168.1.102
0x00000003          1 192.168.1.103
0x00000004          1 192.168.1.104
0x6901a8c0          1 192.168.1.105
0x6a01a8c0          1 192.168.1.106
0x6b01a8c0          1 192.168.1.107
0x6c01a8c0          1 192.168.1.108
```

This example is similar to cman_tool status but it shows the membership information too. I've added the -i and -H switches here so you can see the effect they have on the output. The IP address is shown rather than a node name (so it doesn't need to be resolved), and the node IDs are shown in hexadecimal so that the assigned ones are a little less frightening. There are options to sort the output by nodeid, nodename or IP address should you be fussy about these things ... I am, that's why I added the option.

## Runtime information

The very sharp-eyed among you will notice that corosync-quorumtool doesn't have the comma-separated values option that cman_tool had. I'm not actually sure how many people used this option but it was there, and now it's not. Sorry. There is a '-p' option that looks, from the usage text, that it might do that. But it doesn't. Sorry. It's just a more consistent version of the normal output. Sorry. You can sort the list by node name or node ID though. Yeah. Sorry.

If you want run-time information about corosync and votequorum then corosync-cmaptool is actually the best place to look for getting program-readable output. It also shows a comprehensive list of variables that quorumtool does not, so it can be very useful for finding out just what is happening in your cluster node. If you ask us complicated questions about a corosync cluster we'll almost certainly come back to you for this information if you don't provide it initially.

Along with all of the settings I've talked about above and, in fact, the whole of corosync.conf, corosync-cmaptool also dumps out information in the runtime.xxxx area. I'll mention the specific quorum-related ones here but if you're interested in the workings of your cluster node and don't value your sanity too highly I can recommend browsing the full output.

*runtime.votequorum.lowest_node_id*
This is the lowest active nodeid known to votequorum. If you remember what I said about auto_tie_breaker (and you'd better!) this is the node that will be used to determine the quorate partition should the cluster get split in half and auto_tie_breaker_node: is set to lowest.

*runtime.votequorum.highest_node_id*
This is the highest active nodeid known to votequorum. It's similar to above but the, er, highest node id instead of the lowest. Obvs.

*runtime.votequorum.wait_for_all_status*
If wait_for_all is active then this indicates whether all of the nodes have yet been seen. If 0 then all of the nodes have been seen and quorum will behave normally. If 1, then there are nodes that have not been seen (ie expected_votes has not been reached, remember?) and the cluster will not be quorate until they have.

*runtime.votequorum.two_node*
Predictably this indicates that two_node mode is active. Normally this will reflect the contents of quorum.two_node but it can be cleared if a quorum device is active when two_node is attempted to

be set. This value indicates the actual active value and always overrides what you might see in quorum.two_node.
*runtime.votequorum.ev_barrier*
The expected votes barrier is the lowest that expected_votes is allowed to fall to without manual intervention. It's used in the operation of allow_downscale.

*runtime.votequorum.qdevice_can_operate*
Indicates that the quorum device API can be used. Options such as two_node, last_man_standing, auto_tie_breaker, wait_for_all, and allow_downscale are all incompatible with a quorum device and will clear this flag. If this flag is 0 (it defaults to 1) then an attempt to register a quorum device will fail.

*runtime.votequorum.this_node_id*
No prizes for guessing what this is, it is the local node ID! It's put here to save you trying to work it out by reading the corosync.conf file and parsing the node IP addresses by hand. Aren't we lovely to you?

*runtime.votequorum.atb_type*
This is a number that corosync uses internally to check the nodelist when auto_tie_breaker is in use. If it is set to 0, ATB is disabled; 1, ATB uses the lowest nodeid; 2, ATB uses the highest nodeid; 3, ATB consults the list of nodes in quorum.auto_tie_breaker_nodes. Although this is really for internal use it might be handy for you to check that corosync did actually do what you asked it to.

### Quorum device

I have touched on quorum device above but not mentioned it at all otherwise. With all of the above new options it's hoped that most installations will no longer need qdiskd to get the functionality they need from the cluster, and thus life should be a lot simpler. For those few poor souls that do need to use qdiskd still, err ... it's not there yet! Sorry.

### Well, Well, Well

So, that was a brisk overview of the new quorum system and I hope it's warmed you up. If you need more information read the man pages or contact me (chrissie) or fabbione and we'll do our best to help.

# Appendix A. Migrating code from libcman

This appendix is just a list of libcman calls showing you where in corosync to find equivalents. Most of them have very similar names so it shouldn't be too hard I hope.

**`cman_admin_init`**
**`cman_init`**
**`cman_finish`**
**`cman_setprivdata`**
**`cman_getprivdata`**
**`cman_start_notification`**
**`cman_stop_notification`**
**`cman_start_confchg`**
**`cman_stop_confchg`**

**cman_get_fd**
**cman_dispatch**

These are not really cman-specific, so just use the equivalent calls for the subsystem(s) you are using.

**cman_set_votes**
**cman_set_expected_votes**
**cman_set_dirty**
**cman_register_quorum_device**
**cman_unregister_quorum_device**
**cman_poll_quorum_device**
**cman_get_quorum_device**
**cman_get_node_count**
**cman_get_nodes**
**cman_get_node**

Use libvotequorum. The equivalents should be self-evident I hope.

**cman_is_quorate**

Use libquorum, or libvotequorum. You should only use libvotequorum if you are also using other features of that module. If all you need to know is the quorum status then you should use libquorum as that will work regardless of quorum provider.

**cman_get_version**
**cman_set_version**
**cman_get_cluster**
**cman_set_debuglog**

Use libcmap to read/write to the cmap database

**cman_kill_node**
**cman_shutdown**
**cman_replyto_shutdown**
**cman_get_node_addrs**

Use libcfg

**cman_leave_cluster**

If you just need to leave the cluster then libcfg will work. Votequorum has no equivalent of the LEAVE flag, you should configure your cluster for this behaviour in corosync.conf (allow downscale) if you need it.

**cman_send_data**
**cman_start_recv_data**
**cman_end_recv_data**

Use libcpg

**cman_get_node_extra**

**`cman_get_subsys_count`**
**`cman_is_active`**
**`cman_is_listening`**
**`cman_barrier_register`**
**`cman_barrier_change`**
**`cman_barrier_wait`**
**`cman_barrier_delete`**
**`cman_get_extra_info`**
**`cman_get_fenceinfo`**
**`cman_node_fenced`**
**`cman_get_disallowed_nodes`**

These calls are not implemented at all, sorry. You're on your own. Most of these calls should not have been used by non Red Hat application in the first place, they were there to support cman_tool and system daemons. If you've been using cman_is_listening then the cpg API provides all the necessary functionality.