# ORS and Cassandra deployment suggestions

## Cassandra database usage by ORS

ORS uses Cassandra database to store session information for persistence reason. In current ORS deployment and call flow the stored session uses by ORS only to Recover voice call session status upon ORS switchover.

## Starting the Cassandra Cluster Nodes

Your Cassandra nodes must be started in a certain order:

* Start the seed nodes.
* Start the other non-seed nodes.

The seed node is one of the nodes specified in the seeds option.

- class\_name: org.apache.cassandra.locator.SimpleSeedProvider

parameters:

# seeds is actually a comma-delimited list of addresses.

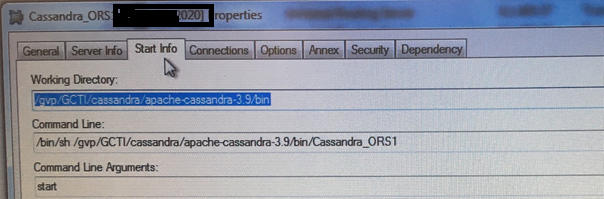
# Ex: "<ip1>,<ip2>,<ip3>"

- seeds: "seed node ip in DC1"

In multiple Datacenter installation seed for each DC

## Create Cassandra applications in CME

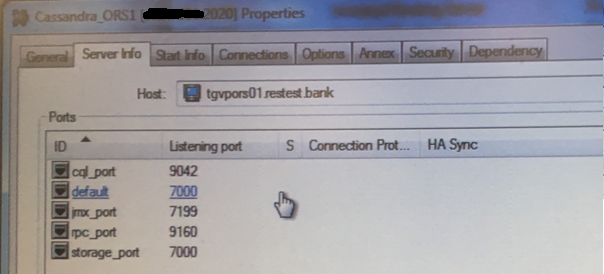
* Cassandra application configured as Third-Party Server name: Cassandra\_ORS1
* Start Info TAB like::



LCA uses the Working Directory and Command line configuration parameters to match started service on linux with the application configured in CME:

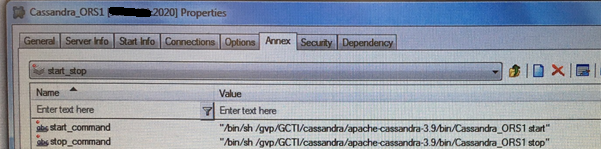
* Server info tab: select host of Cassandra

Note: the ports van also be listed here for informational point of view (that not used by Genesys)



* **Annex Tab**:

Start\_command points a Cassandra startup daemon script



Cassandra start up Daemon script:

#!/bin/sh

# Check user

if [ "$USER" != "gvp" ]

then

echo "Script must be run as user: gvp";

exit 1;

fi

cd /gvp/GCTI/cassandra/apache-cassandra-3.9/bin

start() {

echo -n "Starting Cassandra... "

./cassandra -f

echo "OK"

return 0

}

stop() {

echo -n "Stopping Cassandra... "

./stop-server

echo "OK"

return 0

}

case "$1" in

start)

start

;;

stop)

stop

;;

restart)

stop

start

;;

\*)

echo $"Usage: $0 {start|stop|restart}"

exit 1

esac

exit $?

The Cassandra –f command ensures that the Cassandra starts as foreground process and also the daemon process finishes only when Cassandra also stopped. This way the genesys LCA can monitories the Cassandra just checking status of daemon process on linux host.

* Edit Cassandra stop-server script under bin folder

At end of the file edit the lines:

User=”gvp”

pgrep -u $user -f cassandra | xargs kill -9

1. Start Cassandra from command line on linux:

/gvp/GCTI/cassandra/apache-cassandra-3.9/bin/<CassandraAppName> start &

1. Stop Cassandra from command line on linux:

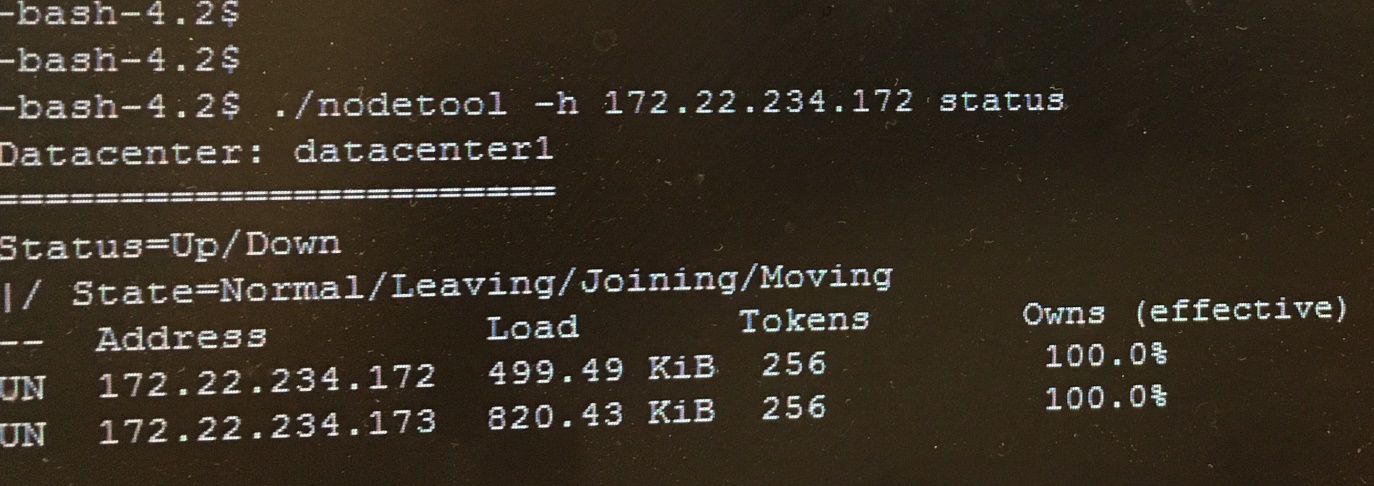
/gvp/GCTI/cassandra/apache-cassandra-3.9/bin/<CassandraAppName>stop

## Cassandra Maintenance hints

### Verify Cassandra status using nodetool command

Because Cassandra is a critical component of ORS solution to ensure session persistence, it is essential to keep track of its health. The Datastax documentation provides some really good information about how to do this at <http://docs.datastax.com/en/cassandra/2.0/cassandra/tools/toolsNodetool_r.html>.

Genesys recommends that you use the nodetool utility that is bundled with your Cassandra installation package and that you make a habit of using the following nodetool commands to monitor the state of your Cassandra cluster.



* **ring command:** nodetool -h <HOST\_NAME> -p <JMX\_PORT> ring

Displays node status and information about the cluster, as determined by the node being queried. This can give you an idea of the load balance and whether any nodes are down. If your cluster is not properly configured, different nodes may show a different cluster; this is a good way to check that every node views the cluster the same way.

* **status command:** nodetool -h <HOST\_NAME> -p <JMX\_PORT> status

Displays cluster information.

* **compactionstats command:** nodetool -h <HOST\_NAME> -p <JMX\_PORT> compactionstats

Displays compaction statistics.

Displays the compaction throughput on the selected Cassandra instance. By default it is 32 MB/s.

You can increase this parameter if you observe permanent growth of database size after the TTL and grace periods are passed. Note that increasing compaction throughput will affect memory and CPU consumption. Because of this, you need make sure to have sufficient hardware to support the rate that you have selected.

nodetool –h <HOST\_NAME> -p <JMX\_PORT> getcompactionthroughput

To increase compaction throughput to 64 MB/s, for example, use the following command:

nodetool -h <HOST\_NAME> -p <JMX\_PORT> setcompactionthroughput 64

### Database Recovery

Depending on the replication factor and consistency levels of a Cassandra cluster configuration, the ORS can handle the failure of one or more Cassandra nodes in the data center without any special recovery procedures and without interrupting service or losing functionality. When the failed node is back up, the ORS automatically reconnects to it.

Therefore, if an eligible number of nodes have failed, you should just restart them.

However, if all the Cassandra nodes (with 3 nodes configuration 2 nodes stopped etc…) in your cluster have failed or stopped, you will lose functionality. To ensure a successful recover from failure of multiple nodes, Genesys recommends that you:

* Stop every node, one at a time, with at least two minutes between operations.
* Then restart the nodes one at a time, with at least two minutes between operations.

### Upgrading Cassandra Nodes

You can upgrade your Cassandra version without interrupting service if:

* The version you are upgrading to is in the same stream (for example, from one 3.9.x version to another)
* You are not changing your database schema

Use the following steps for this task:

* Stop the first Cassandra seed node.
* Preserve your database storage.
* Upgrade your Cassandra version, following the instructions in the Release Notes for the new version.
* Be sure that your database storage is in the preserved state (the same set of files).
* Start the first Cassandra seed node.
* Execute steps 1 through all nodes.
* Verify that the Cassandra cluster is working, as shown above in Verifying Your Cassandra Cluster.

If your upgrade plans include changing your database schema or changing Cassandra versions between streams (for example, from 2.0 to 2.2), then you will have to interrupt service. Use the following steps for this task:

* Stop all of your Cassandra nodes.
* If your database schema has been changed since you installed the previous version, update the Cassandra database, following the instructions in the Release Notes for the new version.
* Configure each node, following the instructions in the Release Notes for the new version.
* Start the Cassandra seed nodes.
* Start the other nodes.
* Verify that the Cassandra cluster is working, as shown above in Verifying Your Cassandra Cluster.

### Cassandra authentication

The Cassandra default deployment does not use any authentication for nodetool (JMX access, that can be accessed from host only) and also the default user is cassandra with default cassandra password

Advised to change the Cassandra user password and switch JMX access with authentication

**Change Cassandra password**:

* Login to cqlsh using command: ./cqlsh -u cassandra -p cassandra 172.22.234
* **Ruin command: ALTER** **USER** cassandra ***WITH******PASSWORD*** *'new password';*

**Configure JMX authentication**

* edit and copy cassandra-env.sh changing lines:

JVM\_OPTS="$JVM\_OPTS -Dcom.sun.management.jmxremote.authenticate=true"

JVM\_OPTS="$JVM\_OPTS - Dcom.sun.management.jmxremote.password.file/gvp/GCTI/cassandra/apache-cassandra-3.9/bin conf/jmxremote.password"

* c**onfigure user and password for JMS remote access** in jmxremote.password file

More info: https://support.datastax.com/hc/en-us/articles/204226179-Step-by-step-instructions-for-securing-JMX-authentication-for-nodetool-utility-OpsCenter-and-JConsole

* + **copy jmxremote.password file to <Cassandra Home>/conf dir**

****

* + **Change owner: chmod 400 jmxremote.password**
  + Add cassandra with readwrite permission to /<jre\_install\_dir>/lib/management/jmxremote.access:

monitorRole readonly  
cassandra readwrite  
controlRole readwrite \  
create javax.management.monitor.\*,javax.management.timer.\* \  
unregister

* + **restart cassandra**
  + **verify Cassandra node status using command:**

./nodetool -h <cassandra node ip> -p <JMX port> -u cassandra -pw cassandra ring

* Create user for ORS authentication

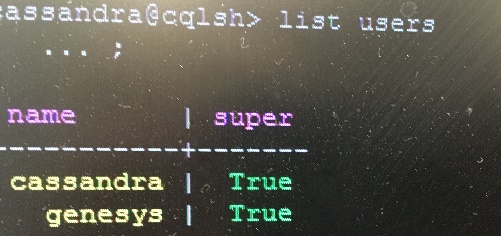
ORS server can access to Cassandra server using a specific user/password configured in ORS srever/persistence section, options: username/password

Create user in Cassandra

* + - Stop ORS servers
    - Configure user and password in ORS:consider this is “gen\_ors\_cassandra”
    - Drop keyspace Orchestration if that exists
    - Login to Cassandra with user cassandra: cqlsh using command: ./cqlsh -u cassandra -p cassandra 172.22.234

CREATE ROLE gen\_ors\_cassandra WITH PASSWORD = 'gen\_ors\_cassandra' AND LOGIN = true AND SUPERUSER = true;

* Verify if user created:



* Start ORS servers and verify of Orchestration keypace have been created and connected to cassandra

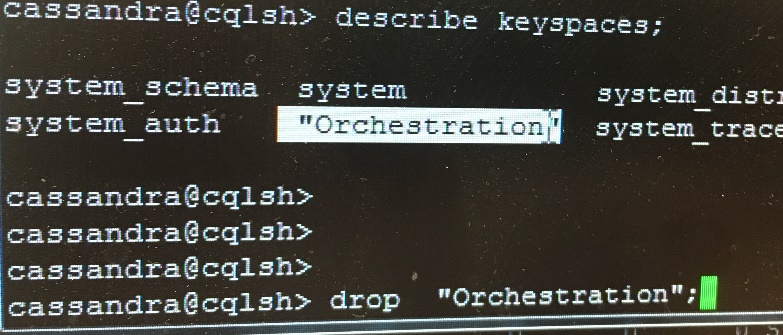
### Recover ORS Keyspace

ORS creates Orchestration keyspace (configuration if ORS/Persistence/Cassandra-keyspace-name) in Cassandra. The creates automatically upon startup if that keyspace does not exists.

If Orchestration keyspace corrupted in any reason and impossible to recover then that can be dropped or recreated from scratch (Note: ORS does not store historical data in this keyspace in current implementation. Just stores when ORS routing used for callback or scheduled sessions)

Drop keyspace:

Login to cqlsh using command: ./cqlsh -u cassandra -p cassandra 172.22.234



Restart ORS servers and verify if keyspace recreated

### Create alarms to monitor ORS and Cassandra connectivity

* Alarm when connection lost:
  + Name: ORS\_Cassandra\_Connection\_Lost
  + Detect Event: 23025
  + Selection mode: Application Type: Orchestration server
  + Cancel event: 23024
  + User action: verify healthy of Cassandra node using nodetool, verify network status
* Alarm when persistence is not available:
  + Name: ORS\_Cassandra\_Persistence\_notAvailable
  + Detect Event: 23002
  + Selection mode: Application Type: Orchestration server
  + Cancel event: 23024
  + User action: verify healthy of Cassandra node using nodetool, verify network status, verify Cassandra logs to check reason of the persistence problem

## Verify replication factor of the keyspaces

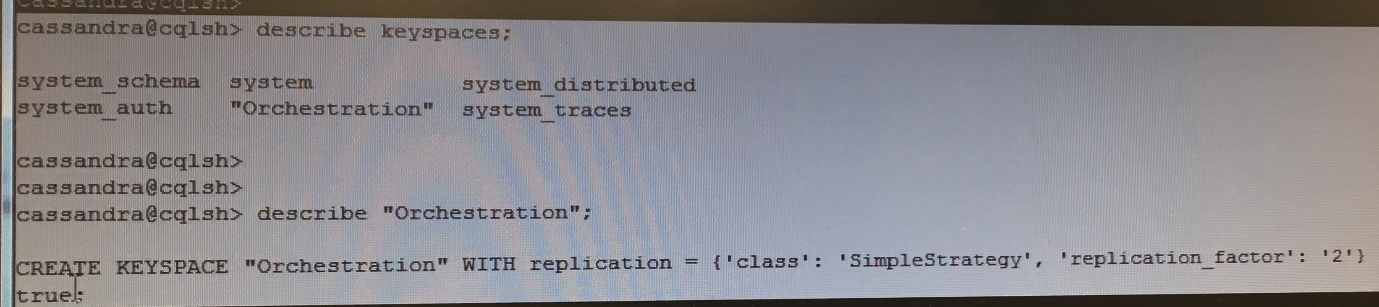
Replication factor of the keypsaces for Orchestration and system\_auth (this keyspace used for authentication) should be the same as ORS/Persistence/Cassandra-strategy-option

Verify keyspace Orchestration configuration using cqlsh:

Note: cqlsh uses local LOCAL\_QUORUM read level constancy to access to Cassandra database. That means with replication\_factor=2 both Cassandra node should be running to access with cqlsh.

ORS uses access R/W consistency level QUORUM and if that does not satisfies by the Cassandra database (EG: one node does not answer) then can go DOWN to consistency level ONE.

The means with consistency level ONE the ORS still can access and work with Cassandra cluster (with 2 nodes) if one node fails …



Verify keyspace system\_auth configuration using cqlsh:

Important Note: by default the replication\_factor of the system\_auth is 1. That means if one of the node stopped the ORS will fail to authenticate and access to Cassandra.

That need to change to replication\_factor to 2 using command:

**ALTER** **KEYSPACE** system\_auth **WITH** **REPLICATION** =

{'class' : 'SimpleStrategy', 'replication\_factor':2};

