Supplement to Orchestration Server Deployment Guide

Orchestration Server 8.1.4

11/28/2024

Table of Contents

Supplement to the Orchestration Server 8.1.4 Deployment Guide **3**

ORS with Redis Cluster **4**

- Design ORS with Redis Cluster 5
- Configure ORS with Redis Cluster 7
- Migrate to ORS with Redis Cluster 9

Additional Configuration **11**

- Configuring FIPS for ORS on Windows 12
- Configuring FIPS for ORS on Linux 14

Supplement to the Orchestration Server 8.1.4 Deployment Guide

This supplement describes updates to the Orchestration Server 8.1.4 Deployment Guide.

The complete documentation set for Orchestration Server can be found here.

ORS with Redis Cluster

Starting from 8.1.401.10, ORS supports using Redis for session persistence and recovery. If necessary, secure connection can be enabled between ORS and Redis.

> **Important**
>
> - Currently, ORS supports session persistence and recovery via unsecure connection to Cassandra. However, this support is valid only if Cassandra 3.x or lower is used. For example, if you upgrade Cassandra to 4.x, you cannot enable session persistence and recovery features of ORS.
> - If secure connection is enabled between ORS and Redis, performance of ORS might be impacted. Redis is also designed to be accessed by trusted clients inside trusted environments.

Advantages of using Redis over Cassandra

- Improved ORS performance due to faster read and write operations, and in-memory storage.
- Better support for Redis. Much faster and more stable communication. Redis is also used by many other Genesys Engage components like GWS.
- More reliable session persistence and recovery.

Design ORS with Redis Cluster

Redis is an open source (BSD licensed), in-memory data structure store used as a database, cache, message broker, and streaming engine. To learn more about Redis, visit https://redis.io/docs/about/. Redis scales horizontally with a Redis Cluster topology. Data is automatically shared across multiple Redis Cluster nodes. Redis Cluster also provides some degree of availability.

> **Important**
> ORS supports persistence with Redis Cluster only. Connections to Redis Sentinel or a standalone Redis instance are not supported.

A pool of connections to Redis Cluster will be shared between ORS worker threads; consequently each working thread will have a TCP connection to each Redis Cluster node.

Redis installation and configuration

Refer to Redis documentation at https://redis.io/docs/install/ for instructions on installation and configuration of Redis. There is no special configuration required in ORS with regards to installation of Redis.

> **Important**
> All Redis versions are supported.

ORS will store data in Redis under the following key:

`ORS:<ORS_CLUSTER_NAME>`

Redis recommended capacity estimate

The number of Redis Cluster Nodes and the amount of memory vary vastly depending on the number of expected simultaneous ORS sessions on the ORS Cluster and session complexity. Genesys recommends that you use Redis Cluster with a minimum of 3 nodes. On average, 3 nodes Redis Cluster should be sufficient to handle upto 9 HA pairs of ORS applications.

For Redis Node Memory amount, Genesys recommends a similar memory allocation per Redis node as for the corresponding ORS application.

No special Redis Cluster configuration required for ORS to be able to use it for persistence. However, it is recommended to setup Redis Cluster with high availability and AOF persistence enabled. Additionally, authentication (user/password) or encryption (TLS) can be configured for Redis Cluster for ORS connection.

Expiration and cleanup of data stored in Redis

Redis works with an in-memory dataset. ORS deletes all keys it stores in Redis when they are not needed. Additionally, each record that is added to Redis has an expiry time (TTL) associated with it, or a periodical cleaning up procedure is run. This expiry time (TTL) helps to prevent invalid Redis entries and to limit Redis DB memory growth.

Configure ORS with Redis Cluster

The following configuration options have been added to the Orchestration `persistence` section to support Redis integration:

- **type** (it is recommended to set this to `redis` only after configuring the rest of the options)
- **redis-nodes**
- **username** (optional)
- **password** (optional)
- **tls-enabled** (optional)
- **tls-cert** (optional)
- **tls-key** (optional)
- **tls-cacert** (optional)
- **tls-cacertdir** (optional)
- **tls-sni** (optional)

Refer to the **persistence Section** in Application-Level Options for detailed description of the above options.

Enable Redis

- Configure the following options:
  - **type**
  - **redis-nodes**
- Set **sessionid-with-nodeid** to `true`

You must restart the ORS application for any changes in these options to take effect. Verify that the Redis connection works as expected using clients such as **RedisInsight** before enabling it with ORS.

TLS support for Redis Cluster connection

To encrypt the data communication between ORS and Redis databases, TLS support must be enabled on both ORS and Redis sides. Enabling TLS could impact the performance, so it must be taken into consideration before switching from non-TLS connection to encrypted. Refer to Redis documentation, https://redis.io/docs/management/security/ to know more about TLS support in Redis and how to enable it. Verify that the Redis TLS connection works as expected using clients such as **RedisInsight** before enabling it with ORS.

To enable ORS to connect to Redis Cluster with TLS support, specify the following ORS application options:

- **tls-enabled**
- **tls-cert**
- **tls-key**
- **tls-cacert**

Although **tls-cert** and **tls-key** are optional, if you specify one of them, you must also specify the other. Instead of specifying **tls-cacert**, you can also specify **tls-cacertdir** to the directory where certificates are stored.

> **Important**
>
> The TLS feature is currently not enabled on Windows platform due to its limited support by Redis Plus-plus library which ORS uses to communicate with Redis Cluster.

Disable persistence and recovery of digital sessions

Generally, we recommend to disable persistence and recovery of digital sessions. In case of an ORS switchover, failover, or disconnect, digital interactions are immediately placed back in queue by eServices. This is the major difference from voice interactions, Therefore, there is no need to persist and recover digital sessions. When the connection between ORS and Interaction Server is restored, interactions will continue to be processed according to the interaction queue logic. Even for chat interactions, there is no need for session recovery, as chat logic is not handled by ORS session. There is no risk of incomplete or stuck session on ORS due to restoration or if an event is missed. Persistence of digital session puts performance penalty on ORS applications and Cassandra or Redis. Also persistent data for digital sessions put extra capacity requirement for Cassandra or Redis in-memory data storage.

Therefore, persistence and recovery of digital sessions triggered by eServices interaction to Redis is disabled.

Migrate to ORS with Redis Cluster

Preparation steps

1. Deploy Redis Cluster and configure it for ORS persistence.
2. Deploy Redis Cluster CPU and memory monitoring tools.
3. Upgrade ORS applications to 8.1.401.10 or higher.
4. Configure all necessary ORS Application options for persistence to Redis, but do not enable it yet (do not set [persistence] **type**=`redis` yet).
5. Restart ORS using the standard procedure. That is, you must restart backups, switchover, and restart new backups.
6. Ensure that ORS applications continue to work as expected. Verify that persistence using Cassandra continue to work as expected, if enabled.

Enable Redis persistence

1. Enable Redis persistence by setting `[persistence]` **type**= `redis` for backup applications. Restart backup applications for the new setting to take effect. Trigger a switchover to make them Primary.
2. Verify that Primary ORS applications are now connected and are doing serialization to Redis. After the switchover, ORS sessions which were in progress, will not be restored and routing may be failed for those calls. Verify that new ORS sessions are processed as expected.
3. Monitor Redis Cluster nodes CPU and memory consumption. Add more Redis nodes or upgrade memory capacity depending on the monitoring data. Consult Redis support, if necessary.
4. Enable Redis persistence by setting `[persistence]` **type** = `redis` for backup applications. Restart backup applications for the new setting to take effect. Trigger switchover to make them Primary.
5. Verify that new Primary ORS applications are now connected and are doing serialization to Redis. After switchover, ORS sessions which were in progress, should be restored from Redis. Verify that all calls are processed as expected.

Rollback

If the migration to ORS with Redis is not successful or if the new solution after the migration is not working as expected, do the following steps to rollback:

1. Disable Redis persistence for all ORS applications.
2. Restart ORS using the standard procedure (restart backups, switchover, and restart new backups).
3. Investigate the cause of the issue and resolve it before doing the migration procedure again.

Post migration steps

Remove Cassandra when you are satisfied with the new ORS with Redis Cluster solution.

Additional Configuration

This document describes the latest configuration updates to Orchestration Server.

Configuring FIPS for ORS on Windows

With the newer OpenSSL 3.x that comes supported with ORS 8.1.401.11+, in addition to setting the option **scxml\fips-enabled**, additional configuration steps are necessary to enable FIPS.

In ORS 8.1.401.11 and later versions, FIPS related modules/tools are located as follows:

- **fips.dll** is located in **<ORS_installation_folder>\fips\fips.dll**
- **openssl.exe** is located in **<ORS_installation_folder>\tools\openssl.exe**

To enable FIPS, follow these steps:

> **Important**
>
> The sample commands given in this procedure are based on the following assumptions which can vary depending on the user environments:
>
> - ORS installation path - **C:\GCTI\ORS_8.1.401.11**
> - Custom path for FIPS related config files - **C:\GCTI\ORS_FIPS_CFG**

1. Create an FIPS module configuration file, **fipsmodule.cnf** using the `openssl` utility:

   ```
   openssl fipsinstall -out C:\GCTI\ORS_FIPS_CFG\fipsmodule.cnf -module C:\GCTI\
   ORS_8.1.401.11\fips\fips.dll
   ```

2. Create an OpenSSL configuration file, **openssl.cnf** that includes **fipsmodule.cnf**, in the **C:\GCTI\ORS_FIPS_CFG** folder.

   ```
   config_diagnostics = 1
   openssl_conf = openssl_init

   .include C:/GCTI/ORS_FIPS_CFG/fipsmodule.cnf

   [openssl_init]
   providers = provider_sect

   [provider_sect]
   fips = fips_sect
   base = base_sect

   [base_sect]
   activate = 1
   ```

   > **Important**
   >
   > When mentioning the path for **fipsmodule.cnf**, you must use forward slashes, for example, `C:/GCTI/ORS_FIPS_CFG/fipsmodule.cnf`.

3. Set the OPENSSL_CONF environment variable that points to the location and filename of **openssl.cnf**.

```
set OPENSSL_CONF=C:\GCTI\ORS_FIPS_CFG\openssl.cnf
```

4. Set the OPENSSL_MODULES environment variable that points to the folder with **fips.dll**.

```
set OPENSSL_MODULES=C:\GCTI\ORS_8.1.401.11\fips
```

5. (Optional) You can verify that FIPS is configured properly using the `openssl list -providers` command before starting ORS.

```
openssl list -providers
Providers:
  base
    name: OpenSSL Base Provider
    version: 3.0.10
    status: active
  fips
    name: OpenSSL FIPS Provider
    version: 3.0.10
    status: active
```

6. Start ORS only after the variables OPENSSL_CONF and OPENSSL_MODULES are set.

Configuring FIPS for ORS on Linux

With the newer OpenSSL 3.x that comes supported with ORS 8.1.401.13+, in addition to setting the option **scxml\fips-enabled**, additional configuration steps are necessary to enable FIPS.

To enable FIPS, follow these steps:

1. Run the `fipsinstall.sh` script provided in the ORS installation folder.
   This script runs the FIPS module self-test and generates proper OpenSSL configuration files **openssl.cnf** and **fipsmodule.cnf** that are mandatory for FIPS mode.
2. Set the following environment variables as `fipsinstall.sh` suggests:
   - `LD_LIBRARY_PATH` to **<ORS install directory>/fips** and **<ORS install directory>/lib64** directory
   - `OPENSSL_MODULES` to **<ORS install directory>/fips** directory
   - `OPENSSL_CONF` to **<ORS install directory>/openssl.cnf** file

Important

- If ORS is started by Local Control Agent (LCA), ensure that the above environment variables are set for the session/account from which ORS is starting.
- The procedure for enabling FIPS mode is similar to the procedure for Genesys Security Pack, but it is important to have `LD_LIBRARY_PATH` to point to **<ORS install directory>/fips** instead of the folder of Genesys Security Pack.
- When configuring ORS for a secure connection, like TLS, in non-FIPS mode, ensure that:
  - ORS option `scxml\fips-enabled` is set to `false`
  - None of the `OPENSSL_MODULES` and `OPENSSL_CONF` environment variables are defined
  - `LD_LIBRARY_PATH` includes the Genesys Security Pack folder instead of **<ORS install directory>/fips**