Chat Server Administration Guide

Deploying High-Availability Chat Server

4/15/2026

Deploying High-Availability Chat Server

Overview

Chat Server can run in high availability (HA) mode where in the case of any Chat Server failure, chat sessions can be continued on other running Chat Server instances. Run Chat Server in load-balancing mode (also known as N+1) to enable HA mode. In load-balancing mode, configure Chat Server to run all instances in primary, or active, mode with no backup applications configured. **Note:** Primary and backup mode are still supported, but not recommended.

When running Chat Server in HA mode:

- The web chat application (for instance, Genesys Mobile Services or GMS) selects an active Chat Server instance to begin a new chat session. If the instance becomes unavailable during the course of the chat session, the web chat application selects another active Chat Server instance where the session will be restored and continued. At session restoration, Chat Server updates the interaction properties in Interaction Server with connection parameters that reflect the new location of the chat session.
- Agent Desktop watches for interaction property updates. After receiving the appropriate notification, Agent Desktop reconnects to the chat session at the specified Chat Server instance.

In HA mode, Chat Server stores the intermediate transcript after each submitted chat session message in persistent storage. This allows the chat session to be restored on a different Chat Server instance upon request. You can configure either of the following options to store the intermediate session transcripts:

- UCS – UCS configuration is simpler but creates an additional load on UCS and its database which the system can tolerate for small or medium sized deployments.
- Cassandra – Cassandra requires special configuration but removes the load from UCS which is beneficial to deployments with higher loads.

Note: In both cases, once the session has ended, the final transcript will be stored in UCS.

Configure Chat Server for HA

Configure as follows:

1. Configure and deploy the required number of Chat Server instances based on the expected load.
2. Connect Web API Server (GMS) to the *webapi* port of all Chat Server instances.
3. Connect Interaction Server to *ESP* port of all Chat Server instances.
4. Set the following values for options in the **settings** section for Chat Server applications:

- **session-restoration-mode** = `simple`

  This enables Chat Server's session restoration functionality.

  Note: The **session-restoration-mode** option has no effect unless the **transcript-auto-save** option is set to a valid positive value.

- **transcript-auto-save** = `1`

  This forces Chat Server to update the transcript in persistent storage (UCS or Cassandra) after each submitted message. You may also set this option to 2 (notify clients when the transcript is updated), however that would be effective only if the agent desktop can process special notifications from Chat Server (in particular, the notice **ucs-save-fail/save**). From the standpoint of resources, using the value 2 will slightly increase CPU usage; also Genesys Interaction Workspace does not support this functionality.

- **transcript-save-on-error** = `close`

  This forces Chat Server to close the chat session (without a final update in UCS) if, during the session, Chat Server detects a non-recoverable error or failure message when trying to store the intermediate chat session transcript.

5. Review the values for the following options (see the eServices Options Reference for full descriptions):
   - **transcript-resend-attempts**
   - **transcript-resend-delay**
   - **transcript-save-notices**

   The default values are acceptable for HA functionality; however you may wish to evaluate whether those values produce the behavior that you expect.

6. (Introduced in Chat Server version 8.5.312.10) To purge a chat session from the Chat Server instance that was processing that chat session before the restoration, set the configuration option session-restore-do-purge to `true`. While it should be used cautiously, it can be helpful in chat deployments where a chat session might accidentally be moved to another instance of Chat Server (for example by a malfunction in load balancing, or network issues). When enabled, and after a successful chat session restoration, Chat Server sends a purge request through Interaction Server to the Chat Server instance that was previously processing that chat session. That chat session is then completely removed from the Chat Server.
7. (Introduced in Chat Server version 8.5.316.02) If session-restore-extend-by is enabled (with any non-default value), you must:
   - Set session-restore-push-send to `true` if your solution is using GMS CometD Chat V2 or enables push notifications in Chat V2
   - Set a value for flex-push-on-join (Introduced in Chat Server version 8.5.315.05)
8. (Introduced in Chat Server version 8.5.316.02) If you believe the workflow might stop the interaction without closing the chat session, you may need to provide a non-empty value in ixn-submittedby-name. This forces Interaction Server to notify all instances of Chat Server when the interaction is stopped in the workflow. When this event is received, Chat Server closes the chat session. Otherwise, if the chat session is restored on a different Chat Server, it will not be closed.

Deploying Chat Server with Cassandra (Optional)

When Chat Server uses UCS to save intermediate transcripts it produces an additional load on the UCS database, especially for deployments with a high volume of customer chat interactions. To improve the performance of UCS and its database, Chat Server (starting with release 8.5.104) can

use Cassandra for this functionality. With Cassandra, Chat Server requires UCS only to store the final chat transcript upon chat session completion.

Important

- There are no configuration options for Chat Server to enable Cassandra. Instead, the presence of the application connection to Cassandra RAP forces Chat Server to use Cassandra.
- Chat Server supports Cassandra only when Chat Server is deployed either on Windows or Linux.
- Chat Server, deployed with Cassandra, must run in the UTF-8 mode to support non-ASCII characters in chat conversations. **Note:** This is not required starting from Chat Server version 8.5.301.06

To facilitate the process above, the following steps must be completed after configuring Chat Server for HA:

- Deploy Cassandra.
- Initialize Cassandra for Chat Server.
- Connect Chat Server with Cassandra.

Deploy Cassandra

Officially, Chat Server supports Apache Cassandra 3.11, 4, and 5. Download the latest stable release of Cassandra here.

Important

For multi-node (cluster) Cassandra installation, use NTP (Network Time Protocol) to synchronize the clocks on all nodes.

Cassandra installation
For simple one-node Cassandra deployment:

1. Modify the **cassandra.yaml** configuration file:
    1. Set the value of **seeds**, **listen_address**, and **rpc_address** to the host IPv4 address.
    2. Make sure that Cassandra ports specified in the **cassandra.yaml** configuration file do not overlap with ports used by existing applications.
2. For load testing purposes, modify the **cassandra.yaml** configuration file:
    1. Set the value of **auto_snapshot** to `false`.

2. Set the value of **compaction_throughput_mb_per_sec** to `0`.
3. Set the value of **write_request_timeout_in_ms** to `10000`.

3. Start Cassandra node.

Initialize Cassandra for Chat Server

The initialization scripts are located in the *cassandra* sub-folder of the Chat Server installation folder. Before running the scripts using the Cassandra CQL Shell to create a new keyspace and the required tables, you must set the following values:

- Replication factor: In production, Genesys recommends a **replication_factor** of at least 3. The replication strategy should be set according to the cluster and datacenter configuration.
- Time-to-live: Occasionally, in case of failures some records in Cassandra are not be deleted by Chat Server. To setup a Cassandra self-cleanup procedure that will delete records after a certain time, select the appropriate script from the "Cassandra" sub-folder and set **default_time_to_live** and **gc_grace_seconds**.

Note: For version 8.5.104 of Chat Server, initialization scripts are not included with installation package. Scripts could be found here.

Connect Chat Server to Cassandra

To Connect Chat Server to Cassandra:

1. Create and configure a Chat Server Cassandra RAP application object based on the *ChatServerCassandraRAP* template provided in the installation package.
   1. Configure **Host** in the **Server Info** tab to point to one of the Cassandra cluster nodes. Configure the *default* port to the Cassandra cluster connection port and set the **Reconnect Timeout**.
      Note:
      - The **Reconnect Attempts** parameter in the **Server Info** tab is not used.
   2. In the **Options** tab, configure the **cassandra** section. Refer to the Options Reference Guide or the contents of ChatServerCassandraRAP.xml.
      Note:
      - In production, the recommended read and write consistency level is **quorum**.
      - If the **contact-points** option is not set, the Cassandra Cluster uses the **Host** defined in the **Server Info** tab as a contact point. If set, this option's value is used instead of the **Host** defined in the **Server Info** tab.
      - In order to configure authenticated access to Cassandra nodes, specify username and password. **Note**: You need to provide required configuration for Cassandra in the **cassandra.yaml** configuration file for **authenticator**.

2. Optionally, see Configure a secure connection to Cassandra for more information.
3. Add a connection from each Chat Server application to the newly created Chat Server Cassandra RAP.
4. Restart Chat Server.

> **Tip**
>
> In order to monitor Cassandra availability for Chat Server, configure alarms in management framework for log messages:
>
> - 59520: Cassandra status changed to "UNAVAILABLE".
> - 59521: Cassandra status changed to "AVAILABLE".

Run a Test

A properly configured solution with HA mode must work without any additional configuration for other components. This section describes a simple test.

Requirements:

- GMS and Widget CX
- Interaction Workspace (agent desktop)
- At least two running instances of Chat Server

Conduct the test as follows:

1. Start a chat session using Widget CX.
2. Send a message to verify that the chat session is active.
3. Then kill the Chat Server process with the ongoing chat session using Task Manager on Windows or `kill -9` on UNIX. GMS will then attempt to connect to another Chat Server instance where the chat session will be restored. At this point you will see a message showing that a user was disconnected and connected again.
4. Send a message to verify that the chat session continues.
5. Optionally, examine the Chat Server logs to see what actions were performed by the server to restore the chat session.

How to Upgrade Running Chat Solution

To upgrade Chat Server to a newer version in the N+1 deployment (also called load-balancing mode, when two or more instances are running in primary mode) without any service interruption the

following steps are necessary for each Chat Server instance:

1. Make a certain instance of Chat Server unavailable for the creation of a new chat session (from GMS) by disabling Chat Server application in Configuration Server/Layer.
2. Wait until all current chat sessions are finished in this particular instance. This can be validated in Chat Server logs by the message "Int 59245 data: deleting session with sid=... and intx=.. (current sessions=0)". Or by requesting KPI counters via the web (REST) interface and validating that "session_created-sessoin_closed" is equal to zero.
3. Stop Chat Server. Replace Chat Server with a newer version (uninstall existing IP, install a new one). If needed, modify/update the Chat Server application. Enable the application in configuration. Start Chat Server.

If the load allows, several instances of Chat Server could be upgraded at the same time. Make sure to run enough Chat Server instances to handle the current load.

Chat Server could also be shutdown (not recommended) without being disabled and without waiting for current sessions to be finished. In this case, GMS moves chat sessions, running on this instance, to a different instances of Chat Server. This might result in short disruptions for those chat sessions (with additional participant left/added messages in chat session transcript).