Chat Server Administration Guide

Sizing Guide, Setting Load Limits, and Health Monitoring

4/1/2025

Sizing Guide, Setting Load Limits, and Health Monitoring

Sizing Guide

The following guidelines are recommended for a Genesys Chat solution running on a single host with two Intel Xeon 3.0 GHz processors. Observe these recommendations for an optimum (without significant delay) performance.

Item	Maximum
Message size	4 KB (Genesys Desktop limitation; Chat Server does not have this restriction.)
Transcript size	54 KB (Genesys Desktop limitation; Chat Server does not have this restriction.)
Concurrent sessions (in a realistic simple scenario)	1000 per Chat Server
Messages per second	50 (rare temporary peaks up to 150)
Sessions opened and closed per second	10 (rare temporary peaks up to 30)

Chat Server limits the maximum number of concurrent opened connections (for all protocol and participants) with:

- 32768 on Linux.
- 4096 on Windows.

When opening and closing the connection, Chat Server prints in the log the current number of opened connections (as "conns="). Connections, with the exception of a few persistent system ones, are used in a couple of different ways:

- Each agent or bot connected to a chat session consumes one connection and then releases it when they leave the chat session.
- Each time a customer chat application sends a request (either via REST API or CometD), GMS opens the connection and then, after receiving a reply from Chat Server, closes the connection.

Note: You can configure a timeout in Chat Server that prevents keeping unused connections open. Use the `user-register-timeout` option to set the maximum time between establishing the connection and receiving:

- Either a basic protocol `Register` request
- Or any flex protocol request

Connection Delay with Antivirus

It may take some time (up to several minutes on some UNIX Platforms) for Chat Server to connect to an unopened port on a Windows host that is running an antivirus program. For example, if Chat Server is running on Linux and is trying to connect to an inactive UCS instance, it could take up to three minutes for Chat Server to detect that the listening port is not open.

Setting Load Limits

Starting in the 8.5.0 release of Chat Server, you can impose load limits on Chat Server: when Chat Server reaches the specified limit, it no longer creates new sessions or restores existing sessions.

Set load limits using the following configuration options (full descriptions are in the eServices Options Reference):

- Enable or disable the general functionality of load limitation using limits-control-enabled.
- Set specific limits:
  - limit-for-flex-users — Maximum number of currently logged-in flex users.
  - limit-for-reply-delay — Configures the maximum average delay (in milliseconds) for processing requests. The average value is calculated on the limit-average-interval interval. This delay increases if the Chat Server instance is overloaded with a large number of incoming requests. **Note:** This delay directly corresponds to CPU consumption by the Chat server process; it starts to grow when the Chat Server CPU is running closer to, or higher than, 100% of one CPU core.
  - limit-for-sessions — Maximum number of concurrent chat sessions (the value of which must not be larger than the value for limit-for-flex-users).

  If any of these limits is reached, Chat Server stops creating and restoring sessions.
- If Chat Server is configured in primary/backup mode (not recommended; see Deploying High-Availability Chat Server), you may want to stop it from reporting `service unavailable` to SCS when a limit is reached. You can do this using limits-reached-report-scs. Blocking the reports avoids a scenario in which Chat Server in primary/backup mode closes a chat session because (1) Chat Server reaches any of its configured load limits, (2) Chat Server sends a `service unavailable` notification to Solution Control Server, (3) SCS switches Chat Server to backup state, which closes the chat session. (This scenario does not apply if Chat Server is in N+1 mode: multiple Chat Servers with no backup configured).
- Set the point at which Chat Server returns to full functionality using limits-restore-threshold. This value is a percentage of the limit set by the three **limit-for-X** options.

Example

If **limit-for-flex-users** is set to 400 and **limits-restore-threshold** is set to 80, then:

1. When the number of flex users reaches 400, Chat Server stops creating and restoring sessions, and rejects login attempts by flex users.
2. When the number of flex users falls to 320, Chat Server returns to full functionality.

KPI (Key Performance Indicator) counters

Starting with release 8.5.103, Chat Server includes KPI (Key Performance Indicator) counters that monitor activity within the server.

Accessing KPI counters

Access KPI counters in one of two ways:

- The Chat Server log. by configuring options log-output-content, log-output-proviso, log-output-timeout in the **[health-service]** section.
- The web REST interface which you can configure by:
    - Adding a port with the ID=`health` to the ports of Chat Server.
    - Adjusting the **soap-*** options found in the **[health-service]** section.

Web interface

Access the web interface through the following URL format: `http://ServerName:ServerPort/Counters?<list of parameters>` where:

- *ServerName* is the host name where Chat Server is running.
- *ServerPort* is the web service port, also specified as the `health` port of Chat Server.

Web interface parameters:

Name	Description	Valid Value	
metadata	Returns a list (in JSON format) of all supported counters, with descriptions.	`true`, `yes`, `false`, `no`	`false`
content	Returns a list (in JSON format) of counters according to the provided parameter value.	`all`—returns all available counters new—returns only recently updated counters set—returns all initialized (non-zero) counters	`all`

Name	Description	Valid Value	
reset	Resets all counters to zero.	`true`, `yes`, `false`, `no`	`false`
prometheus	Returns metrics in Prometheus format. (Introduced in Chat Server version 8.5.312.10)	`true`, `yes`, `false`, `no`	`false`
gms_info	Returns metrics about GMS nodes and the associated chat session (only the Prometheus format is supported). When used, all other parameters are ignored. (Introduced in Chat Server version 8.5.312.10)	`true`, `yes`, `false`, `no`	`false`

If a parameter is omitted or has an invalid value, then the default value is used for that parameter. Parameters are processed according to the following rules:

- If **reset** is `true`, then all the counters will be reset and then the rest of the parameters will be processed.
- If **metadata** is `true`, then the `content` parameter will be ignored and the metadata will be sent.

Example URLs:

- `http://`*`hostname`*`:7000/Counters?content=set`
- `http://`*`hostname`*`:7000/Counters?metadata=true&reset=true`

How counter values are calculated

All counters, except the **process memory** counter, are cumulative. The value begins to accumulate the moment the application is launched or the counters are reset. To calculate the difference, the user must use two sample counters from different times and subtract the earlier sample from the later one. To find the counter's rate value, divide the difference by the number of elapsed seconds between the two samples.

Important

On some platforms, the time for processing internal activities may be reported as zero. This does not indicate an issue with the counter. On the contrary, a rapidly growing value on the counter for internal activity indicates that the server is overloaded.