This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Orchestration Server Deployment Guide

Performance Monitoring

12/14/2025

Contents

Performance Monitoring

Starting with Release 8.1.400.21, you can monitor Orchestration Server performance using a set of performance counters and configure conditions that will trigger Management Layer alarms. The performance data can also be written to the log or displayed in a Genesys runtime metrics client (RTME), such as Pulse.

Performance Parameters and Counters

ORS can monitor the following performance parameters:

PARAMETER DESCRIPTION	MEASUREMENT	COUNTER ID
Number of active sessions (sessions that started or recovered on this node)	Current number	`active-sessions`
Number of pending session creation requests	Current number	`pending-sessions`
Time, required to create a session	Time in msec	`create-session-time`
Document processing time	Time in msec (duration from the doc_retrieved metric)	`document-processing-time`
SCXML Event queuing time	Time in msec	`scxml-event-queuing-time`
Fetch (HTTP methods) response time	Time in msec	`http-fetch-time`
Fetch (ESP method) response time	Time in msec	`esp-fetch-time`
Fetch (URS method) response time	Time in msec	`urs-fetch-time`
ORS/URS request – initial response delay	Time in msec (between a request and a requestID response)	`urs-request-time`
ORS-StatServer – SOpenStat-SStatOpened delay	Time in msec (between SOpenStat and SStatOpened)	`openstat-time`
Cassandra latency	Time in msec (already measured)	`cassandra-latency`
Function execution time	Time in msec	func-execution-time, `param1` is the function name, like `_genesys.session.getListItemValue`
State time	Time in msec	`state-time`, where `param1` is the id of state
Redirect time	Time in msec	`redirect-time`
Rate of session creation in sessions per second (available with 8.1.400.27)	Sessions per second	`creation-session-rate`

PARAMETER DESCRIPTION	MEASUREMENT	COUNTER ID
Rate of exits from state (available with 8.1.400.27)	States per second	`state-rate`
Rate of function execution (available with 8.1.400.27)	Number per second	`func-execution-rate`
Rate of fetch action executions (available with 8.1.400.27)	Fetches per second	`fetch-rate`
Current number of active CTI calls (available with 8.1.400.27)	Current number	`active-calls`

Moving Average

ORS collects performance data samples and then uses them to calculate a *moving average*, which contains the *latest* measurements. The size of this sample depends on the window during which the sample data is collected. You can define the window as the number of last measurements, or as the number of last seconds during which data should be collected. The size of the window you define (in number of last measurements or in the number of last seconds) has a major impact on the value of the moving average. You define the window size based on the particular parameter and your contact center's needs.

ORS Performance Monitoring Options

Configure the performance monitoring options in the Orchestration Server `Application` object `performance-alarms` section. Two new options are introduced to support the performance monitoring functionality: `alarm-name` and `alarm-check-interval`.

You can create different alarms with the same counter-id. Two examples are shown below.

```
slow-session-creation_one=counter-id=create-session-time;threshold=4000;
window-type=time; window-size=120
slow-session-creation_two=counter-id=create-session-time;threshold=4000;
window-type=samples; window-size=120
```

alarm-name

For alarm-name, specify a counter ID from the Performance Parameters and Counters table.

Option section: `performance-alarms`

Configuration object: ORS `Application` object

Default value: None

Valid values: Enter the alarm definition using the format and values below.

Value changes: Immediately upon notification.

```
<alarm-name>=counter-id=<counter-id>; threshold=<value>;window-type=<time|samples>;
window-size=<int>;enabled=<true|false>;
debug=<true|false>;param1="value";alarm-msg-id=<default|alarm-msg-00...| alarm-msg-24>
```

where:

- `counter-id`. Possible values are listed in Performance Parameters and Counters table. Specifies a performance-related counter that ORS is able to monitor.
- `window-type`. Possible values `time`, `samples`. Specifies the type of the window for moving average: time if the window is measured in time (seconds) or samples if the window is defined as the number of last samples.
- `window-size`. Size of the window, if `window-type=time`, then window-size specifies time interval, in seconds, on which the moving average will be calculated. If `window-type=samples`, then it specifies the number of the samples used to calculate the moving average.
- `threshold`. Specifies the threshold value. If the moving average exceeds this value, then alarm will be triggered.
- `enabled`. Optional parameter. Possible values are `true` (default), or `false`. If set to `false`, the alarm is ignored. This options gives the opportunity to temporary disable an alarm without deleting it from the configuration.
- `debug`. Optional parameter. Possible values `true`, `false` (default). If set to `true`, then the value of the moving average for this performance counter is printed in the log even if it does not exceed the threshold.
- `param1`. Optional parameter that may be required for some counters.
- `alarm-msg-id`. Optional parameter, introduced in ORS 8.1.400.27. See Separate Message Identifiers for Performance Counter Alarms for details.

alarm-check-interval

Option section: `performance-alarms`

Configuration object: ORS `Application` object

Default value: `60` seconds

Valid values: Any non-negative integer from `1` to `600`

Value changes: Immediately upon notification.

This option is where you specify how often alarm conditions will be checked.

At the time of the check, if the alarm condition is true for at least one counter and it was false at the time of previous check, an alarm-level message will be printed in the format shown below along with names of all triggered performance alarms, their values and threshold values:

```
Alr 23028 Following alarms exceeds threshold values: <alarm-name1>:
<alarm-value> (<threshold-value>), <alarm-name2>:
<alarm-value2> (<threthold-value2>), ...
```

For example:

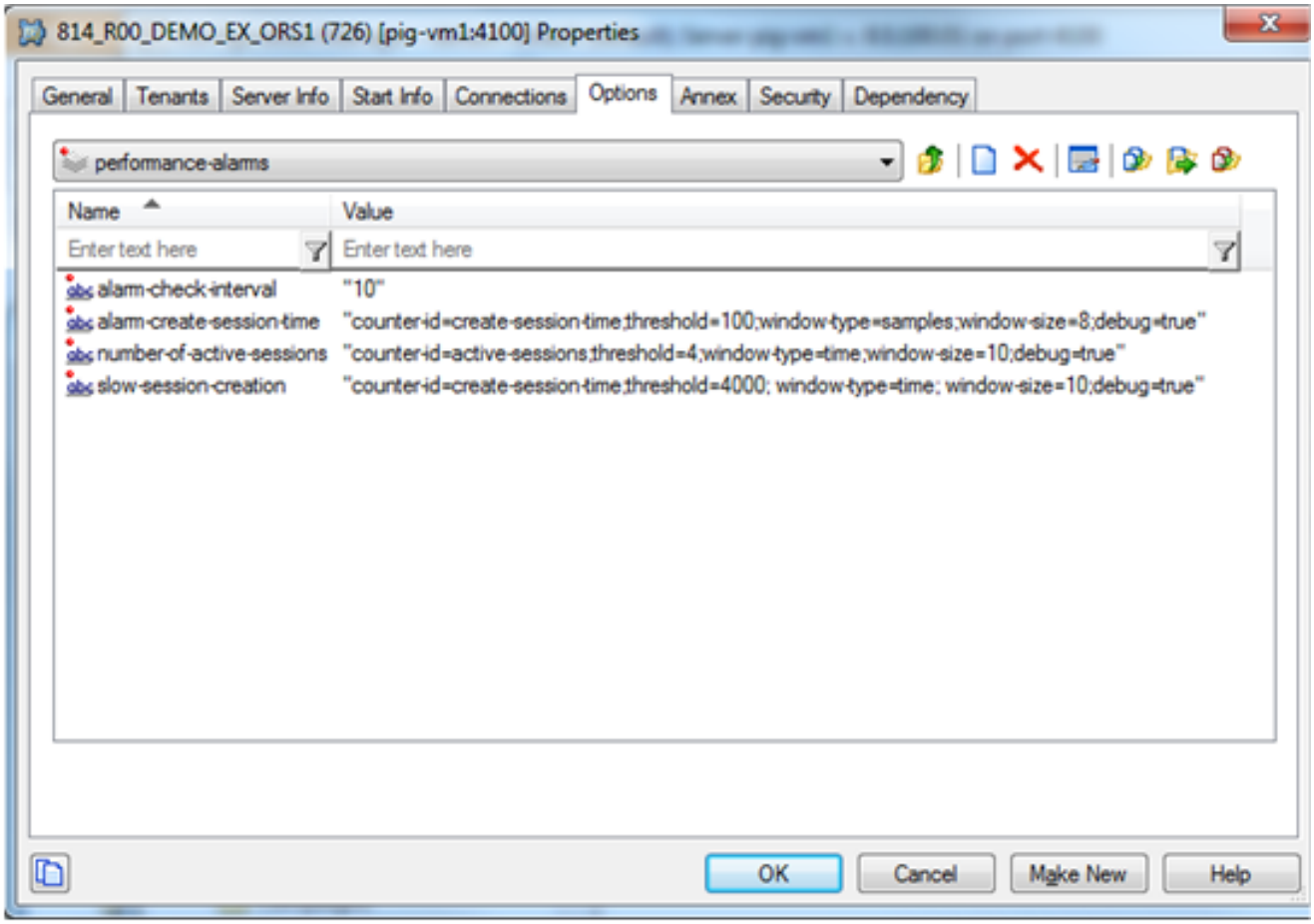
```
Alr 23028 Following alarms exceeds threshold values: SCXML Queuing Time:
40.9 (1.0) State Time: 45.5 (1.0)
```

Example Performance Alarm Configuration

Below is an example ORS performance alarm configuration that triggers when average session creation time exceeds 4 seconds if calculated on 2-minute window.

```
slow-session-creation=counter-id=create-session-time;threshold=4000;
window-type=time; window-size=120
```

An example configuration in the ORS `Application` object `performance-alarms` section (prior to specifying the alarm name) is shown below.



Separate Message Identifiers for Performance Counter Alarms

Starting with Release 8.1.400.27, the ORS performance monitoring feature supplies an individual alarm log entry for each performance counter shown in the above table. This enhancement to the performance monitoring feature supplies:

- A default pair of message identifiers for each type of performance counter: an Alarm-level message identifier to trigger an alarm in Solution Control Interface (SCI) (or Genesys Administrator) and a Standard-level message identifier to clear an alarm in SCI (or in Genesys Administrator).
- A set of 25 reserved message identifier pairs, located in the `*.lms` file, which can be used in a configuration for any performance counter. For example, you could use these identifiers when configuring more than one performance counter for a counter type, such as if configuring `func-execution-time` counters for several different functions.

Configuring Individual Message Identifiers

When configuring message identifiers, you can use an additional, optional parameter, `alarm-msg-id`, in the counter configuration. This parameter defines separate log messages for each type of performance counter. The `alarm-msg-id` values can range from `alarm-msg-00` to `alarm-msg-24`. Or you can use `default` for the default alarm message identifiers for a particular counter type.

The `alarm-msg-id` identifies a pair: the Alarm-level message identifier and the Standard-level message identifier from the reserved range to trigger and clear alarms in SCI (or in Genesys Administrator).

To ensure that each alarm with `alarm-msg-id` has a unique message identifier:

- No two counters may have same non-default alarm identifier.
- No two counters of the same type can have `default` as a value of the `alarm-msg-id` parameter.

Example Confguration

```
Active Calls=counter-id=active-calls; threshold=1000;window-type=samples;
window-size=100;enabled=true;debug=true;
alarm-msg-id=alarm-msg-01;
```

Alarms without the `alarm-msg-id` parameter work as they did prior to ORS Release 8.1.400.27: all alarms are listed in one message. The alarms with `alarm-msg-id` are printed separately with the corresponding message identifier.

Value of message identifiers when `alarm-msg-id=default`:

```
type of alarm           message id to    message id to
                        trigger alarm         clear alarm

active-sessions                25000                      25001
pending-sessions          25002               25003
create-session-time         25004               25005
document-processing-time 25006               25007
scxml-event-queuing-time 25008               25009
http-fetch-time                 25010               25011
esp-fetch-time                25012               25013
```

```
urs-fetch-time                25014               25015
openstat-time                25016               25017
cassandra-latency          25018               25019
func-execution-time          25020                 25021
state-time                  25022                 25023
creation-session-rate          25024                 25025
redirect-time                25026                 25027
urs-request-time          25028               25029
state-rate                   25030                 25031
func-execution-rate          25032                  25033
fetch-rate                   25034                 25035
active-calls                   25036                 25037
```

Value of message identifiers when `alarm-msg-id=alarm-msg-xx`:

```
value of alarm-msg-id   message id to    message id to
                        trigger alarm         clear alarm

alarm-msg-00                 25100                      25101
alarm-msg-01                 25102                      25103
   . . .
alarm-msg-24                 25148                      25149
```

Alarm Triggering Conditions

- It may happen that an alarm condition for a counter becomes true during the interval between the checks specified by the `alarm-check-interval` option, but at the time of the check, the condition is false. In this case, no alarm is reported. If necessary, the `alarm-check-interval` may be decreased to check for alarm conditions more often.
- No alarm is raised if there is not enough data to calculate a moving average according to parameters specified in the performance alarm configuration. The moving average value will not be used to determine an alarm condition if:
  - For `window-type` samples, the number of collected data is less then specified in `window-size`
  - For `window-type` time, the age of the oldest collected sample is less than the value specified in `window-size`

For example, no alarm will be triggered under the following conditions:

- The specified `window-size` is 100 samples, but at the time of the alarm check, only 50 samples were collected.
- The specified `window-size` is 30 seconds, but all data is collected during last five seconds.

Clearing Alarms

If alarm messages are printed and conditions for all configured performance counters become false during the `alarm-check-interval` time, the following `Standard`-level message is printed to clear Management Layer alarms:

```
Std 23028 All performance alarms are cleared.
```

The `alarm-check-interval` option prevents over-populating the log with performance alarm-related messages in cases when improperly configured performance alarms are triggered too often.

For the alarms that do not exceeds threshold values but have parameter `debug=true`, the Debug-level messages are printed in following format:

```
Current alarms values:
        <Counter name1> (<threshold>): <mean> +/- <standard deviation>
[<min>,<max>] %<current sample size>
 . . .
        <Counter nameN> (<threshold>): <mean> +/- <standard deviation>
[<min>,<max>] %<current sample size>
```

If the sample size for some counter is zero (no values to calculate average), the following format will be used:

```
<Counter name1> (<threshold>):N/A
```

Example:

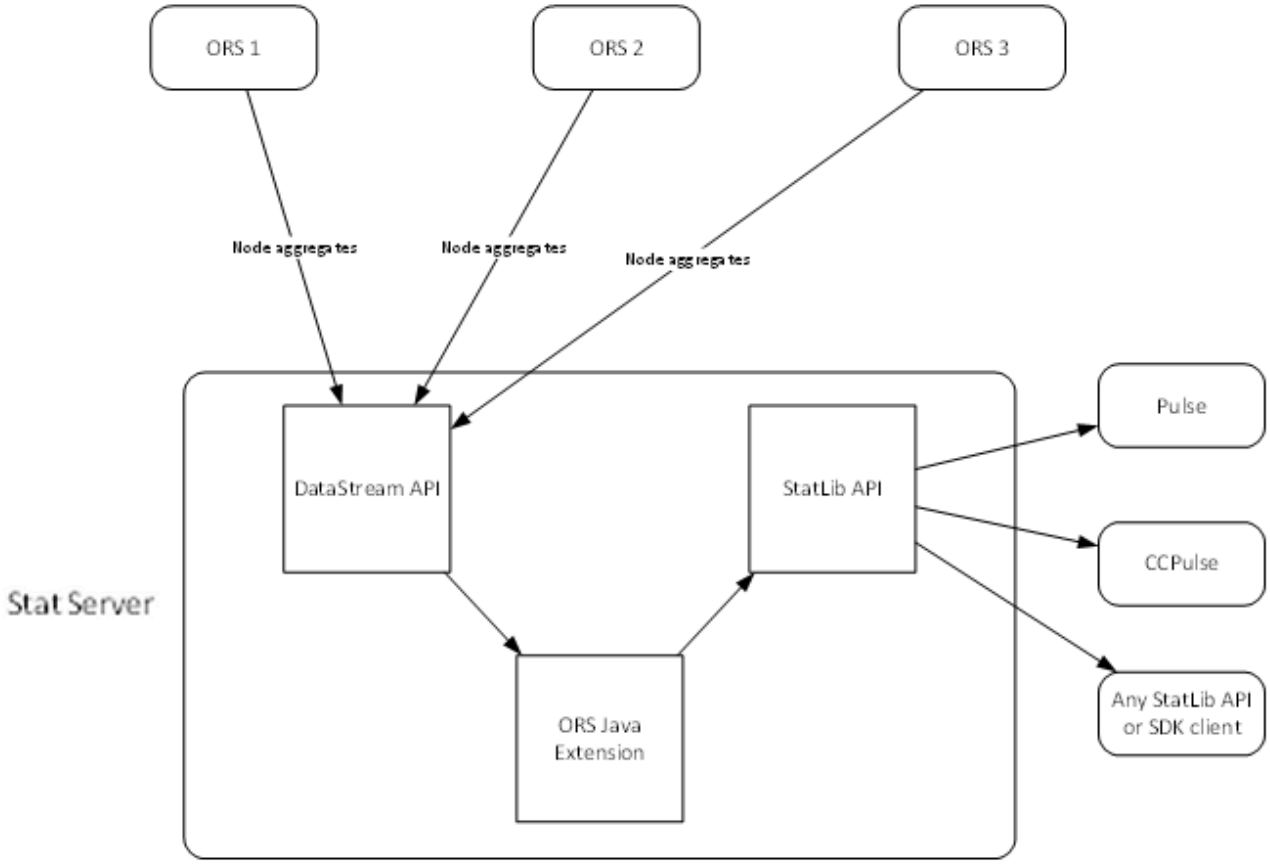
```
Current alarms values:
        Doc Processing Time (1.0):         93.00 +/-22.55 [67.00,122.00] #3
        SCXMP Queuing Time (1.0):         108.00 +/-65.33 [31.00,188.00] #10
        HTTP Fetch (1.0):          N/A
```

The `alarm-check-interval` option also affects how often these messages will be printed.

Displaying Performance Data in a Genesys RTME Client

Starting with ORS Release 8.1.400.30, you can display ORS performance data in a Genesys Real-Time Metrics Engine (RTME) client such as Pulse. Each ORS Node can be configured to supply performance data to a dedicated Stat Server. An ORS Java Extension then processes, aggregates, and delivers the data (via Stat Server) to the GUI client. Any RTME GUI client (such as Pulse) may be used as a presentation layer.

Process Diagram



A summary of the deployment process is as follows:

1. Install the ORS Java Extension.
2. Provision Stat Server.
3. Configure the performance counters as statistics.
4. Configure the statistic update interval.

Installing the ORS Java Extension

- Install the ORS Java Extension located on the Stat Server Real-Time Metrics Engine 8.5 CD. For more information, see the *Stat Server 8.5 Deployment Guide*, Manually Installing the Java Extensions.
- Set parameter `ORSStatExtension.jar` to `True` in the `java-extensions` section of Stat Server `Application` object.

Provisioning Stat Server

To send performance counter information to a dedicated Stat Server, set the `perf-counters-report` in the Stat Server `Application` object to `true`.

perf-counters-report

Option section: `orchestration`
Configuration Object: Stat Server `Application` object
Default value: `false`
Valid values: `true`, `false`
Value changes: Immediately upon change.

If `true`, ORS opens a Data Stream to the ORS Java Extension and submits data. Setting to `true` has no effect if ORS does not have a connection to that Stat Server.

Configuring the Performance Counters as Statistics

To configure a performance counter as a statistic, define an explicit statistical type (Stat Type) configuration for that counter. For additional information on this requirement, see the *Stat Server 8.5 User Guide*, Statistical Type Sections and Java Sections. As shown below, each performance counter must have its own section in the **Options** tab of corresponding Stat Server `Application` object.

- `Counter` = <counter name>
- `Objects` = `Tenant`
- `Category` = `JavaCategory`
- `JavaSubCategory` = `ORSStatExtension.jar:Calculator`
- `Nodes` = <list of ORS Nodes> See Setting the Nodes Parameter below.
- `Aggregation` = <aggregation metric – `last`, `avg`, `min`, `max`> See Configuring the Aggregation Function below.

The value of `Counter` must match the value of alarm-name of a particular performance counter as it is configured in the Orchestration Server `Application`.

Setting the Nodes Parameter

The above `Nodes` parameter defines the set of ORS Nodes whose counters will be aggregated to calculate a particular statistic. The value of that parameter may be:

- `any`
- `cluster:<name of ORS Cluster>`, such as `cluster:ORS_CL1` or a
- Comma-delimited list of ORS application names, such as `ORS_1,ORS_2,ORS_3`. If each ORS node consists of a High Availability pair, include only the name of primary ORS `Application` object.

If the value is any or the parameter does not present at all, the corresponding counter will be aggregated without taking the Node into consideration.

Configuring the Aggregation Function

The above `Aggregation` parameter defines the Aggregation function to be used to calculate a given statistic. The value may be one of the following:

- `last` – The value of the statistic is the latest value of the counter received from any of the ORS Nodes from the `Nodes` list.
- `avg` – The value of the statistic is the average value of the counter received from any of the ORS Nodes from the `Nodes` list within the time interval specified in the statistic subscription request.
- `min` – The value of the statistic is the minimum value of the counter received from any of the ORS Nodes from the `Nodes` list within the time interval specified in statistic subscription request.
- `max` – The value of the statistic is the maximum value of the counter received from any of the ORS Nodes from the `Nodes` list within time interval specified in statistic subscription request.

Example:

An example performance monitoring statistic configuration is shown below. It shows the maximum time required to create and start a session for all ORS nodes in the `ORS_CL1` cluster. Assume that alarm name `SessionCreateTime`, configured for the `create-session-time` counter, exists in the ORS nodes configuration.

```
[MaxSessionCreationTime]
Aggregation=max
Category=JavaCategory
Counter=SessionCreateTime
JavaSubCategory=ORSStatExtension.jar:Calculator
Nodes=cluster:ORS_CL1
Objects=Tenant
```

Configuring the Data Stream (Statistic Update) Interval

The above process_diagram shows the Data Stream to the ORS Java Extension. An ORS Node opens a Data Stream after connection/registration to a Stat Server dedicated to the ORS Java Extension. Only ORS Nodes in primary mode will open Data Streams. An ORS Node will close a Data Stream if switched to backup mode. Both primary and backup ORS Nodes use the name of the primary Node in `SDataStreamInfo.pszDataSourceName` upon opening of a Data Stream. ORS submits statistical data to Stat Server periodically, where time between updates may be configured with the following ORS option:

datastream-update-interval

Configuration object: Orchestration Server `Application`
Option section: `performance-alarms`
Default value: 1
Valid values: A number between 1 and 120
Value changes: Immediately upon notification

Defines the time interval in seconds between performance counters submissions from ORS to Stat Servers.

General Performance Monitoring Notes

- It is possible to configure more than one trigger for same counter that will have a different window size and threshold value. Specifying a large time interval for `window-size` may require a considerable amount of memory in which to keep samples if the value of the counter changes quickly.
- For information about Management Layer alarms, see Alarm-Signaling Functions.
- No additional configuration is necessary to have Management Layer alarms triggered when messages related to performance alarms are logged, because the ORS Performance Monitoring functionality uses `Alarm`-level messaging.