Genesys Pulse Deployment Guide

StatServer Data Provider

5/4/2025

Contents

StatServer Data Provider

`StatServer Data Provider` is a microservice that is responsible for collecting data from a Genesys Stat Server.

`StatServer Data Provider` is installed with Genesys Pulse Collector. Files are located inside the `<Genesys Pulse Collector installation folder>/microservices/StatServerDataProvider` directory.

You need to have the `StatServer Data Provider` configured and running in order to receive values of statistics.

The `StatServer Data Provider` depends on the following Genesys Pulse microservices:

- LayoutWatcher

Configuration

The `StatServer Data Provider` microservice does not use Genesys Configuration Server. All configuration options are defined in the configuration file(s). All changes in the configuration files take effect after the microservice restart.

> **Tip**
>
> Starting with 9.0.008.03, `StatServer Data Provider` requires configuration files in YAML format. You can convert existing configuration from `INI` to `YAML` format using the `microservices/StatServerDataProvider/contrib/cfg2yaml.py` script.

All options of the configuration files are divided into sections and subsections. Empty sections are ignored.

For versions < 9.0.008.03 (INI format)

```
[Section_1]
Option_1 = Value_1

[Section_1.Subsection]
Option_11 = Value_11

[Section_2]
Option_2 = Value_2

[Section_3]
List_Option_3 = Value_31,Value_32
```

For versions >= 9.0.008.03 (YAML format)

```
Section_1:
     Option_1: Value_1
     Subsection:
          Option_11: Value_11

Section_2:
     Option_2: Value_2

Section_3:
     List_Option_3:
         - Value_31
         - Value_32
```

Below are all the options supported by the `StatServer Data Provider` microservice. All options are grouped by sections.

log

The section is used to configure microservice's log subsystem. The following options in the section are supported:

- **channel**
  Name of the default log channel.

  Default value: Name of the executable

  Valid values: String
- **verbose**
  Log verbose level.

  Default value: interaction

  Valid values:
  - `all` – all messages (same as debug)
  - `debug` – debug/trace/normal/info/warning/error messages
  - `trace` – trace/normal/info/warning/error messages
  - `interaction` – normal/info/warning/error messages
  - `alarm` – warning/error messages
- **all**
  List of log destinations to log all messages.

  Default value: Empty list

  Valid values:
  - `stdout` – standard console output
  - `syslog` – system log
  - `<path>` – path to the log file
- **debug**
  List of log destinations to log debug messages.

Default value: Empty list

Valid values: `stdout`, `syslog`, `<path>`

- **trace**
  List of log destinations to log trace messages.

  Default value: Empty list

  Valid values: `stdout`, `syslog`, `<path>`
- **interaction**
  List of log destinations to log normal messages.

  Default value: Empty list

  Valid values: `stdout`, `syslog`, `<path>`
- **standard**
  List of log destinations to log info messages.

  Default value: Empty list

  Valid values: `stdout`, `syslog`, `<path>`
- **alarm**
  List of log destinations to log warning/error messages.

  Default value: Empty list

  Valid values: `stdout`, `syslog`, `<path>`
- **segment**
  Split log files in segments of the specified size.

  Default value: `false`

  Valid values:

  - `no`, `false` – do not split log files
  - `<size>[kb|mb]` – segment size
    - `kb` – kilobytes (default)
    - `mb` – megabytes
- **expire**
  Keep the specified number of log file segments. If the value of `0` is specified, then segments are not expired.

  Default value: `0`

  Valid values: Positive integers, `0`

StatServerDataProvider

The section describes how the microservice should collect statistics values from Genesys Stat Server. The following options in the section are supported:

- **host**
  Hostname of the primary Stat Server.

  Valid values: String
- **port**
  Port number of the primary Stat Server.

  Valid values: Positive integers
- **host_bk**
  Hostname of the backup Stat Server.

  Default value: Hostname of the primary server

  Valid values: String
- **port_bk**
  Port number of the backup Stat Server.

  Default value: Port number of the primary server

  Valid values: Positive integers
- **app_name**
  Application name used to connect to Stat Server.

  Valid values: String
- **app_pwd**
  Application password used to connect to Stat Server.

  Default value: Empty string

  Valid values: String
- **reconnect_delay**
  Delay (in secs) before reconnecting to Stat Server.

  Default value: `10`

  Valid values: Positive integers, `0`
- **add_message_timestamp**
  Add additional timestamp to statistic value messages.

  Default value: `false`

  Valid values: `true`, `false`
- **announce_period**
  Number of seconds between sending heartbeat messages.

  Default value: `10`

  Valid values: Positive integers
- **layouts_announce_period**
  Number of seconds between announcing layout objects. If the value is `0`, then send them only when the layout objects are changed.

Default value: 120

Valid values: Positive integers, 0

- **layouts_deactivation_delay**
  Number of seconds to delay layout deactivation. If the value is 0, then layouts are deactivated immediately.

  Default value: 0

  Valid values: Positive integers, 0

- reopen_stats_delay
  Number of seconds to delay re-opening statistics.

  Default value: 20

  Valid values: Positive integers

- **Connection.event_details**
  Details level for logging the events received from Stat Server.

  Default value: brief

  Valid values:

  - quiet, none – do not log server events
  - brief, no, false – log server events without details
  - full, yes, true – log server events with details

- **Connection.queue_size**
  Size of the queue of input events. If the value is 0, the queue is not limited.

  Default value: 0

  Valid values: Positive integers, 0

- **Connection.queue_timeout**
  Number of ms to wait if the queue of input events is full.

  Default value: 0

  Valid values: Positive integers, 0

LayoutSubscriber

The section describes how the microservice should interact with the LayoutWatcher microservice. The following options in the section are supported:

- **target**
  The URI of the endpoint to connect to.

  Valid values:

  - [dns:///]<host>:<port> – a hostname/port combination
  - unix:<path> – the scheme is used to create and connect to UNIX domain sockets. The path represents the absolute or relative path to the desired socket

  - `ipv4:<host>:<port>` – a pre-resolved ipv4 dotted decimal address/port combination
  - `ipv6:[<host>]:<port>` – a pre-resolved ipv6 address/port combination

- **reconnect_delay**
  Delay (in secs) before attempting to reconnect to the LayoutWatcher service.

  Default value: `10`

  Valid values: Positive integers, `0`

- **only_with_alert_conditions**
  Query only the layouts that contain Alert Conditions.

  Default value: `false`

  Valid values: `true`, `false`

- **layout_types**
  List of the layout types which should be queried. If no types are specified, all layouts are requested.

  Default value: Empty list

  Valid values: `GENERIC`, `PCREGULAR`, `PCPERFORMANCE`, `DATADEPOT`, `IFRAME`, `ALERT`, `STATICTEXT`

AeronPublisher

The section describes how the microservice should publish the collected statistic values via Aeron. The following options in the section are supported:

- **driver_directory**
  Directory of the Aeron media driver. It should be the same as the media driver uses. On Linux systems it is better to be a directory inside the `/dev/shm/` directory.

  Default value: System specific

  Valid values: String

- **driver_timeout**
  The amount of time, in milliseconds, that the publisher waits until it determines the Aeron media driver is unavailable.

  Default value: 2000

  Valid values: Positive integers

- **channel.control**
  Multi-Destination-Cast (MDC) control address to be used for dynamically allocating new destination streams.

  Valid values: String in the `<host>:<port>` format

- **channel.uri**
  URI of the Aeron channel. If specified, all other channel parameters are ignored. If the `control` option is not specified then the parameter is required.

  Valid values: String

- **reconnect_delay**
  Delay (in secs) before attempting to reconnect to the media driver.

  Default value: `10`

  Valid values: Positive integers, `0`

How to Run StatServer Data Provider

The `StatServer Data Provider` is implemented as a standalone executable. The following command line options are supported:

- -h, --help
  Print help screen and exit.
- -c, --config-file FILE
  Specifies the path to the file that contains the configuration for the microservice. The option can be specified multiple times. In this case, all the configurations are merged. If no configuration files are specified, the StatServerDataProvider.cfg file located in the executable's directory is used.
- -D, --define NAME=VALUE
  Defines the value of the configuration option. The option can be specified multiple times. Options' values specified in the command line take precedence over the values specified in the configuration files.
- --service NAME
  The microservice is running as a system service.

  On Windows, the option is added automatically during the creation of the service and should not be used directly by the user.
  On Linux, if the option is specified in the command line, then the microservice is running in the background.
  This option is modified in the 9.0.001 release. The short option `-s` is no longer supported.
- --install[=INSTANCE_NAME]
  Creates a Windows service for the microservice. If an instance name is not specified, then the name of the service is StatServerDataProvider. If the instance name is provided, then the name of the service is StatServerDataProvider#INSTANCE_NAME.
  If additional command line options are specified, they are used to run the service.
  This option is introduced in the 9.0.001 release.
- --remove[=INSTANCE_NAME]
  Removes a Windows service for the microservice. If an instance name is not specified, then the name of the service is StatServerDataProvider. If the instance name is provided, then the name of the service is StatServerDataProvider#INSTANCE_NAME.
  This option is introduced in the 9.0.001 release.

Run as a Service on Windows

To create a Windows service, perform the following command:
`path\to\installation\StatServerDataProvider.exe -c path\to\config\file --install`

You can use the NET command in a Windows command prompt to manage the service:

- NET START StatServerDataProvider
- NET STOP StatServerDataProvider

To remove the service, perform the following command:
`path\to\installation\StatServerDataProvider.exe --remove`

If multiple instances of the microservice are required to run on the same host, you can create an additional instance of the service:
`path\to\installation\StatServerDataProvider.exe -c path\to\config\file --install=<instance_name>`

You can use the NET command in a Windows command prompt to manage the service:

- NET START StatServerDataProvider#<instance_name>
- NET STOP StatServerDataProvider#<instance_name>

To remove the service, perform the following command:
`path\to\installation\StatServerDataProvider.exe --remove=<instance_name>`

Run as a Service on Linux

Create a separate `systemd` service configuration file for each instance of the `StatServer Data Provider` service you need to run. For example, create the `systemd` service configuration file `/etc/systemd/system/pulse-statserver-data-provider.service` with the following content:

```
[Unit]
Description=Pulse StatServer Data Provider

[Service]
ExecStart=/path/to/installation/StatServerDataProvider -c /path/to/config/file --service
Type=forking

[Install]
WantedBy=multi-user.target
```

If multiple instances of the microservice are required to run on the same host, you can create an additional instance of the service `/etc/systemd/system/pulse-statserver-data-provider-<instance_name>.service`:

```
[Unit]
Description=Pulse StatServer Data Provider <instance_name>

[Service]
ExecStart=/path/to/installation/StatServerDataProvider -c /path/to/config/file --
service=<instance_name>
Type=forking

[Install]
WantedBy=multi-user.target
```

You can use `systemctl(1)` to manage these services. Type `man systemctl` for more information.

> **Important**
> Make sure that Aeron Media Driver is configured and running on the host.

Limitations

- The `StatServer Data Provider` can monitor members of `Agents` and `Places` groups. Monitoring members of other types of groups is not supported.
- Tenants with non-empty passwords are not supported.