Genesys Pulse Deployment Guide

AlertProcessor

5/4/2025

Contents

AlertProcessor

AlertProcessor is a microservice that is responsible for sending Genesys Pulse Advanced Alert emails. When AlertProcessor detects that some Advanced Alert condition has changed its value, it waits for the alert delay interval and then executes the alert action.

AlertProcessor is installed with Genesys Pulse. Files are located inside the <Genesys Pulse installation folder>/alert-processor directory.

AlertProcessor sends emails in the following cases:

- Advanced Alert condition changes its value from the no-alert value (null, false, 0, empty string, etc.) to some alert value. In this case the Alert is triggered.
- Condition changes its value from one alert value to some other alert value (for example, from 1 to 3, from Elevated to Severe). In this case the Alert remains triggered, but AlertProcessor sends email notification about the change of the alert condition value.
- Condition changes its value from alert value to a no-alert value. In this case the Alert is reset.

Below is an example of the email that AlertProcessor sends:

```
Subject:

Alert "TooManyInternalCalls" in widget "Agent Calls" was triggered for 3 and reset for 1
object(s)

Text:

Alert "TooManyInternalCalls" in widget "Agent Calls" was triggered for 3 and reset for 1
object(s)

Alert was triggered for the following objects:

- Emily Peterson: condition value has changed from false to true
- John Smith: condition value has changed from false to true
- Peter Johnson: condition value has changed from false to true

Alert was reset for the following objects:

- Anthony Small: condition value has changed from true to false
```

You need to have AlertProcessor configured and running in order to receive Advanced Alert emails.

AlertProcessor depends on the following Genesy Pulse microservices:

- StatServer Data Provider
- Formula Processor
- WidgetWatcher
- UserPermissions

AlertProcessor also requires an external SMTP server to be available for sending emails.

Configuration

`AlertProcessor` itself is configured using the `alertprocessor.properties` configuration file. It should be placed in a directory with the `AlertProcesor` jar file. All changes in the configuration file take effect after the microservice restart.

There are several configuration options that can be specified in the `alertprocessor.properties` file. The following configuration options must be configured:

- **aeron.driver.directory**
  Directory used by aeron driver. On linux it is better to use a directory inside /dev/shm/.

  Valid values: Path to a directory

> **Tip**
>
> **`AlertProcessor` will start its own embedded `Aeron Media Driver` automatically if the external driver, configured to use the same directory, is not running.**

- **email.from**
  Email address that will be written in "from" field of emails sent by AlertProcessor

  Valid values: Email address
- **email.smtp.host**
  Host of an SMTP server to be used to send emails

  Valid values: hostname or an ip address
- **email.smtp.port**
  Port of an SMTP server to be used to send emails

  Valid values: positive integer
- **service.widgetwatcher.servers**
  WidgetWatcher service instances defined by host and port pairs in a comma separated list

  Valid values: comma separated list of host and port pairs
- **service.userpermissions.servers**
  UserPermissions service instances defined by host and port pairs in a comma separated list

  Valid values: comma separated list of host and port pairs
- **service.statserverdataprovider.aeron.control**
  StatServerDataProvider aeron control address.

  Valid values: Host and port pair
- **service.formulaprocessor.aeron.control**
  FormulaProcessor aeron control address. Host and port pair.

  Valid values: Host and port pair
- **storage.h2.directory**

Path to a local directory where AlertProcessor can store the data that will be persisted during service restarts

Valid values: Path to a directory

There are also some other options that can be used to tweak the way `AlertProcessor` works:

- **limits.alert_queue**
  The maximal allowed size of an alert queue. If size of the queue with Alerts waiting to be reported reaches this value then new Alerts will be dropped and users won't receive emails for dropped alerts.

  Valid values: Positive integer

  Default value: 500000

- **log.level**
  The log level. Note that logging can be customized further by supplying a full `log4j.xml` file. To use a custom `log4j.xml` configuration file pass `-Dlog4j.configurationFile=<path to an xml file>` when starting `AlertProcessor`. When customizing the `log4j.xml` file it is strongly advised to use `AlertProcessorLayout` as a layout for every file appender because it supplies a header with the application version info. Look at the `log4j.xml` file inside `AlertProcessor` jar to see how to use it.

  Valid values: One of log4j log levels

  Default value: INFO

- **processor.alert_group_interval**
  Alert grouping time interval in seconds. This allows to group more alerts into a single email. The bigger the value the more alerts will be reported in a single email and the bigger the alert delay.

  Valid values: Positive integers

  Default value: 5

- **processor.retry_alert_action_after**
  Number of seconds to wait before trying to redo the alert action.

  Valid values: Positive integers

  Default value: 2

- **service.widgetwatcher.reconnect_interval**
  WidgetWatcher reconnect interval in seconds.

  Valid values: Positive integer

  Default value: 10

- **service.userpermissions.expire_cache_after_seconds**
  Number of seconds after which an entry in user permissions cache expires

  Valid values: Positive integers

  Default value: 240

- **service.statserverdataprovider.aeron.channel**
  StatServer Data Provider aeron channel full specification. Should be used if a channel with a single control specification is not enough. Control address option is ignored if this option is filled.

Valid values: Aeron channel definition

- **service.formulaprocessor.aeron.channel**
  FormulaProcessor aeron channel full specification. Should be used if a channel with a single control specification is not enough. Control address option is ignored if this option is filled.

  Valid values: Aeron channel definition

How to Run AlertProcessor

Carry out one of the following procedures to run AlertProcessor:

Run as a Service on Windows

To create a Windows service, perform the following steps:

1. Navigate to the **alert-processor** installation directory, which contains the **alertprocessor_service.ini** and **alertprocessor_service.exe** files.
2. Edit the **alertprocessor_service.ini** service configuration file, and replace the **JVMPath** value with the absolute path to the **jvm.dll** file in your host environment.
3. To start the service, run the following command in the Windows command prompt:

   ```
   sc.exe create alert-processor start=auto
   binPath="\"<path_to_alertprocessor_service.exe>\" -service alert-processor
   -immediate"
   ```

   where <path_to_alertprocessor_service.exe> is the full path to the **alertprocessor_service.exe** file.
4. If needed, you can manage the service using the SC command in the Windows command prompt:

   ```
   sc.exe start alert-processor
   sc.exe stop alert-processor
   ```

Run as a Service on Linux

To run as a Linux service, perform the following steps:

1. Create a separate `systemd` service configuration file for the `AlertProcessor` service.
   For example, create the `systemd` service configuration file `/etc/systemd/system/pulse-alertprocessor.service` with the following content:

   ```
   [Unit]
   Description=Pulse AlertProcessor

   [Service]
   ExecStart=<absolute path to java executable> -jar path/to/installation/
   alertprocessor.jar
   WorkingDirectory=<absolute path to the AlertProcessor home directory>

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

2. If needed, you can use `systemctl(1)` to manage these services. For more information, run the following command:

```
man systemctl
```

Limitations

- `AlertProcessor` does not currently support any authentication with SMTP server.
- `AlertProcessor` does not currently provide High Availability.
- `AlertProcessor` does not guarantee that some Alert email is sent twice, but it sends a single alert email in most cases.
- `AlertProcessor` has a bigger priority on sending emails for triggered alerts. In some cases the emails about reset alerts is not sent:
  - When an `Alert` is removed from the widget, the reset emails are not sent for all the objects this Alert was triggered for.
  - When a `Widget` is deleted, the reset emails are not sent for all the already triggered alerts in this widget.
  - When an object is removed from the widget, the reset emails are not sent for all Alerts that were triggered for this object.