Genesys Pulse Deployment Guide

Formula Processor

12/18/2025

Contents

Formula Processor

`Formula Processor` is a microservice that is responsible for calculation of alert conditions, represented as JavaScript formulas. Formula Processor listens to changes of statistic values on which alert formulas depend and recompute alert formulas when the change is detected. Alert formula values are then distributed to other microservices.

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

You need to have `Formula Processor` configured and running in order to enable alert condition calculations.

`Formula Processor` depends on the following Genesys Pulse microservices:

- StatServer Data Provider
- MeasureWatcher

Configuration

The `Formula Processor` microservice does not use Genesys Configuration Server. All the configuration is defined in the configuration file(s). All changes in the configuration file take effect after the microservice restart.

Tip

Starting with 9.0.008.03, `Formula Processor` requires configuration files in YAML format. You can convert existing configuration from INI to YAML format using the `microservices/FormulaProcessor/contrib/cfg2yaml.py` script.

All options of the configuration file 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 `Formula Processor` 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`

FormulaProcessor

This section provides some general settings. The following options in the section are supported:

- **add_message_timestamp**

  Add additional timestamp to formula result messages.

  Default value: false

  Valid values: true, false
- **announce_period**

  Number of seconds between sending heartbeat messages.

  Default value: 10

  Valid values: Positive integers

Engine

This subsection of `FormulaProcessor` provides settings for the formula computation engine. The following options in the section are supported:

- **script_execution_threads**

  Number of script execution threads.

  Default value: 10

  Valid values: Integers from 1 to 256
- **script_execution_queue_size**

  Maximum size of the queue of requests for every script execution thread. If the value of 0 is specified then the queue is not limited.

  Default value: 0

  Valid values: Positive integers, 0
- **gc_period**

  Number of seconds between garbage collections.

  Default value: 1800

  Valid values: Integers from 1 to 86400
- **min_object_inactivity**

  Number of seconds of object's inactivity (an absence of any new data for the object) sufficient to mark it as garbage.

  Default value: 900

  Valid values: Integers from 1 to 86400
- **script_parsing_timeout**

  Script parsing timeout (in ms).

  Default value: 2000

  Valid values: Integers from 1 to 60000
- **script_execution_timeout**

  Formula evaluation timeout (in ms).

  Default value: 1000

Valid values: Integers from 1 to 60000

- **idle_notification_period**
  Scripting engine idle notification period (in ms).

  Default value: 1000

  Valid values: Integers from 1 to 60000

- **recomputation_period**
- Formula re-computation period (in secs).
  Default value: 60

  Valid values: Integers from 1 to 3600

MeasureSubscriber

The section describes how the Formula Processor microservice interacts with the MeasureWatcher microservice. The following options in the section are supported:

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

  Default value: No default value

  Valid values:

  - [dns:///]<host>:<port> – a hostname/port combination
  - unix:<path> – the UNIX 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 MeasureWatcher service.

  Default value: 10

  Valid values: Positive integers, `0`
- **measure_types**
  A list of measure types which should be queried. If no types are specified, query all measures.

  Default value: Empty list

  Valid values: GENERIC, FORMULA

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 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.

  Default value: No default value

  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.

  Default value: No default value

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

  Default value: `10`

  Valid values: Positive integers, `0`

AeronSubscriber

The section provides detailed Aeron channel options. The following options in the section are supported:

- **driver_directory**
  Aeron driver directory. Should be the same as the one that the media driver is using.

  Default value: System specific

  Valid values: String
- **driver_timeout**
  The amount of time, in milliseconds, that the subscriber waits until it determines the the Aeron media driver is unavailable.

  Default value: 2000

  Valid values: Positive integers
- **reconnect_delay**
  Delay (in secs) before attempting to reconnect to the media driver.

Default value: 10

Valid values: Positive integers, `0`

channel

This subsection of `AeronSubscriber` describes how the `Formula Processor` microservice should receive statistical values from the `StatServer Data Provider` microservice via `Aeron`. The following options in the section are supported:

- **control**
  Multi-Destination-Cast (MDC) control address that is used for dynamically allocating new destination streams.

  Default value: No default value

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

- **endpoint**
  Local endpoint address that is used for listening incoming messages.

  Default value: No default value

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

- **uri**
  Aeron channel URI. If specified, all other options in this section are ignored. If neither control nor endpoint are specified then this option is required. Multiple channel specifications are acceptable. In case of a single channel, the channel name can be omitted.

  Default value: No default value

  Valid values: String

How to Run FormulaProcessor

The FormulaProcessor 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 `FormulaProcessor.cfg` file located in the executable's directory is used.

- -D, --define NAME=VALUE
  Defines the value of the configuration property. The option can be specified multiple times. The property values specified in 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 FormulaProcessor. If the instance name is provided, then the name of the service is FormulaProcessor#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 FormulaProcessor. If the instance name is provided, then the name of the service is FormulaProcessor#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\FormulaProcessor.exe -c path\to\config\file --install`

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

- NET START FormulaProcessor
- NET STOP FormulaProcessor

To remove the service, perform the following command:
`path\to\installation\FormulaProcessor.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\FormulaProcessor.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 FormulaProcessor#<instance_name>
- NET STOP FormulaProcessor#<instance_name>

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

Run as a Service on Linux

Create a separate `systemd` service configuration file for each instance of the `Formula Processor` service you need to run.
For example, create the `systemd` service configuration file `/etc/systemd/system/pulse-formula-processor.service` with the following content:

```
[Unit]
```

```
Description=Pulse Formula Processor

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

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

If multiple instances of the microservice running on the same host are required, you can create an additional instance of the service /etc/systemd/system/pulse-formula-processor-instance_name.service:

```
[Unit]
Description=Pulse Formula Processor <instance_name>

[Service]
ExecStart=/path/to/installation/FormulaProcessor -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.