Genesys Pulse Deployment Guide

Genesys Pulse 9.0.0

3/5/2025

Table of Contents

Genesys Pulse Deployment Guide

Genesys Pulse is a Genesys application that enables at-a-glance views of real-time contact center statistics within the graphical user interface. On the Genesys Pulse dashboard, widgets display user-defined Donut, Grid, Key Performance Indicator (KPI), or List charts of statistics for objects. You can view and select additional details and options by expanding a widget. Once maximized, you can choose a Stacked Bar, Grouped Bar, Grid or Line Chart view. You can also sort the data, select which objects to include, and edit the widget. See the Genesys Pulse Help for an overview of how to use Genesys Pulse.

New in This Release

This page provides information about new features and enhancements of Genesys Pulse.

Changes Introduced in Release 9.0

Release 9.0.008.02

- **Breaking Change:** Starting with release 9.0.008, Genesys Pulse no longer supports language pack files (`pulse-<locale>.jar`), instead providing out of the box translation for all supported languages.
- **Security:** HTTP security response headers (`X-XSS-Protection`, `Referrer-Policy`, `X-Frame-Options`) were updated according to recent recommendations.
- **Security:** Removed excessive endpoints disclosing information about 3rd party dependencies. New supporting options were added in `pulse.properties` that now allow Pulse to be embedded into an iframe. See Configuring System Security for more information.
- The following 3rd party libraries are upgraded to their latest versions:
  - Jetty 9.4.52.v20230823
  - gRPC 1.51.3
  - Jackson 2.15.2
  - Guava 32.1.2
  - Google Protobuf 3.23.4
  - Aeron 1.41.3
  - Oracle JDBC 21.11.0.0
  - Netty 4.1.94.Final
  - Spring Framework 5.3.29
  - Spring Security 5.8.5
  - Apache Commons Compress 1.24.0
  - Apache Commons DBCP 2 2.9.0
  - h2 2.2.224
  - jQuery 3.7.0
  - jQuery-ui 1.13.2
- Support has been added for PostgreSQL 13.x.
- Alert Processor's debug logging has been improved to be more verbose.
- Added support for different passwords for the keystore and the specific key within the keystore. See TLS: Configuring Genesys Pulse for details.

- Added the ability to open Published Dashboards or Wallboards in a new shared mode, where all changes made and published by the owner are reflected on the Dashboard or Wallboard automatically (enabled with the `[pulse]\enable_push_changes` option).

Release 9.0.007.13

- A new configuration option, [pulse]\disable_cfg_xpath_query, is added to allow more efficient handling of requests to Config Server when checking user permissions. Enable this option to reduce the load on Config Server. The new option works with Config Server versions 8.5.101.21 or higher.
- 3rd party libraries are upgraded for the remediation of the following unexploitable security vulnerabilities: CVE-2022-42889, CVE-2022-22970, CVE-2022-22971, CVE-2022-22978, CVE-2022-2047, and others.

Release 9.0.007.08

- Added a new page to describe the Health Check feature.
- Corrected the order of steps in the procedure Install Genesys Pulse from the Installation Package.

Release 9.0.000

- Genesys Pulse can be started only in a standalone mode with its own `Application` configuration object. Genesys Pulse version 9.0.0 can not be used as a plugin of GAX.
- Ability to fetch historical data series in Genesys Pulse User Interface Extension. Refer to Genesys Pulse User Interface Extension for more information.
- Limits for both personal Widgets and Dashboards (including wallboards). This functionality is supported by new configuration options on the Annex tab of the Access Groups:
  - **max_tabs_per_user**
  - **max_widgets_per_user**
- Ability to specify thresholds for alerts in decimal numbers.
- If the user has no access to the selected objects in the view or headline of the widget, instead of showing "Object Unavailable", Genesys Pulse displays other objects from the selected objects in the widget.
- Advanced Alert capabilities. To enable this feature you need to properly deploy and run services and set the value of the Genesys Pulse option **[pulse]/enable_advanced_alerts** to **true**. After enabling this feature the `Advanced Alerts` section is displayed on the new `Alerts` tab, using which users can specify conditions for an alert and the email address for sending this alert.
  This feature has the following limitations:
  - Alert conditions use the new formula definition format. In this format, values of statistics and formulas are accessed by their alias and not through the global Data object.
    For example:

    ```
    Result = Login_Time <= 200;
    ```

    instead of the old format

    ```
    Result = Data.Login_Time.Value
    ```

- Alert conditions can only use statistics of primitive data types and other alert conditions; it is not possible at the moment to create Alert conditions that depend on:
  - Statistics of the complex data type, like current state
  - `Changes-Based` and `No Notification` statistics
  - Regular formulas
  - Object properties
- At the moment alert conditions cannot use special built-in functions (like `GetFirstUDValue()`) and functions provided by the gts.js script that available to regular formulas.
- It is only possible to configure new standalone services using configuration files.
- There is no HA and horizontal scaling in new services used for this feature.
- As long as the alert condition is stored as javascript code and parsed in order to get condition builder, the original condition's syntax cannot be preserved; therefore, the condition's code can be reformatted/simplified when the builder mode is on.
- New standalone services are AlertProcessor, Formula Processor, and StatServer Data Provider.

- Display format `Status`. This new display format is applicable only for statistics using Statistical Type `ExtendedCurrentStatus`. It allows to specify values displayed for the status (icon, name, media, duration) and to select current status statistics on any chart.
- An option for statistics with display format `Status` to show "Do Not Disturb" status with its duration.
- New logic to update and format heartbeat files. The heartbeat file is always updated, thus the Genesys Pulse Collector configuration option **heartbeat/heartbeat-success-condition** is discontinued. The content of a heartbeat file is in JSON format. The heartbeat file includes the following information:
  - Current timestamp.
  - Startup timestamp.
  - Current uptime.
  - State of connection to Configuration Server.
  - State of connections to Stat Servers.
  - State of connections to DB Servers.
  - Snapshot writing status.

  Example:

```
{
        "currentTimestamp": 1521835976,
        "startupTimestamp": 1521835950,
        "uptime": 26,
        "connectionStatus": {
                "configServer": {
                        "connected": true,
                        "instance": "primary",
                        "lastConnectionTimestamp": 1521835952
                },
                "statServer0": {
                        "connected": true,
                        "lastConnectionTimestamp": 1521835956
                },
                "statServer1": {
```

```
                        "connected": true,
                        "lastConnectionTimestamp": 1521835956
                },
                "dbServerConnection": "all"
        },
        "snapshotWritingStatus": true
}
```

- New command line option `-startup_heartbeat_file` specifying which heartbeat file to use before Genesys Pulse Collector can read regular heartbeat configuration options from its `Application`. This option has only one parameter providing a path to the heartbeat file.
- An automatic reopening of statistics that failed to open due to recoverable Stat Server error. Stat Server release 8.5.108.15 or later is required.
  New Genesys Pulse Collector configuration option in the **[statistic-request-handling]** section controls timeout for reopening statistics in such cases:

  - **open-stat-retry-timeout**
    Valid Values: `1-3600`

    Default Value: `30`

    Change takes effect: After Restart

    Specifies the timeout in seconds, that Genesys Pulse Collector waits before attempting to re-open a failed statistic if Stat Server indicates that the error is recoverable.

Architecture and Components

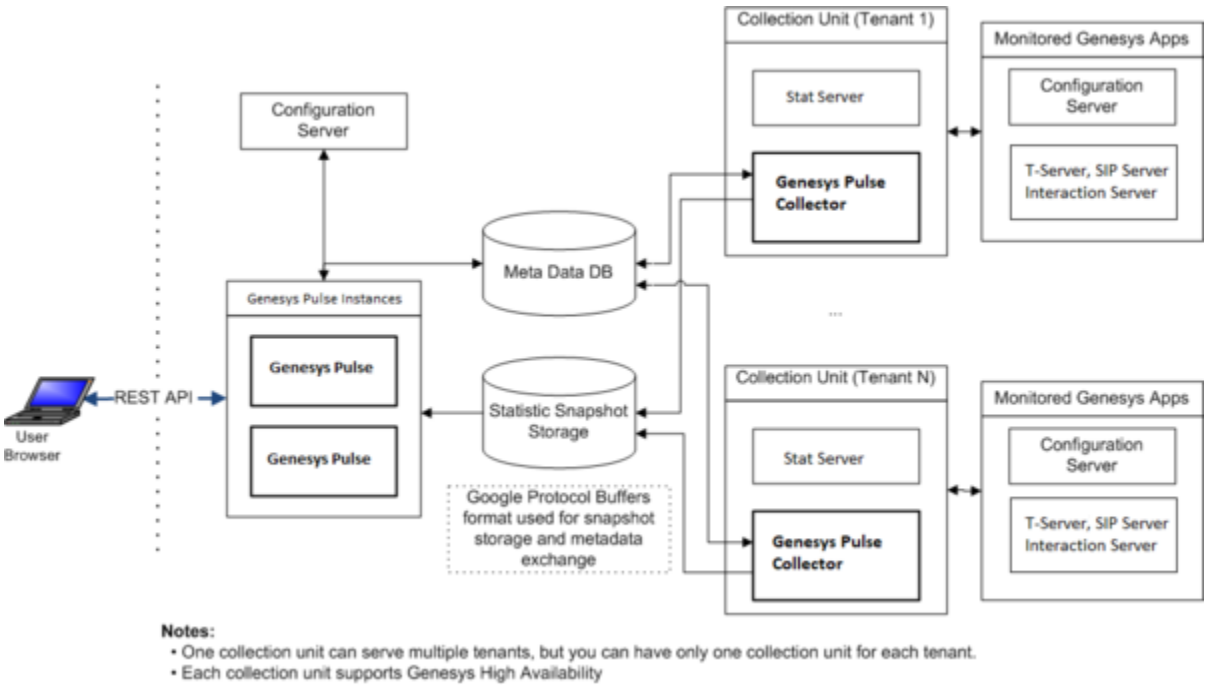
Genesys Pulse is an application that enables at-a-glance views of real-time contact center statistics within the graphical user interface. Genesys Pulse uses widgets to display user-defined List, Donut, Key Performance Indicator (KPI), or Grid charts of statistics for objects.

Using Genesys Pulse you can:

- Create widgets from predefined and user-defined templates for a fast and easy text or graphical presentation of selected or user-defined object statistics.
- Monitor the current state and activity of contact center objects to help make decisions about staffing, scheduling and call routing strategies.

Genesys Pulse Architecture

Major aspects of Genesys Pulse are shown in the following Genesys Pulse Architecture diagram:



Genesys Pulse Collector

Genesys Pulse Collector is a background near-realtime statistical data collection and processing engine. It performs the following activities:

- Reads the metadata from the Genesys Pulse database upon startup and whenever changes are made to report definitions in Genesys Pulse.
- Uses the report definitions stored in the Genesys Pulse database to determine which statistics and objects to include.
- Creates snapshots with current data from Stat Server and formula-based statistics calculated by Genesys Pulse Collector, on the specified file system for reference by Genesys Pulse.
- Maintains a constant connection with Configuration Server to retrieve changes, additions, and deletions to configuration objects.

Genesys Pulse

Genesys Pulse performs the following activities:

- Handles user authentication and permissions validation.
- Filters and delivers report data according to the permissions and tenancy of the user who is requesting the data.

Displays report content in widgets, such as the listing and content of reports.

- Saves the report definitions to the Genesys Pulse Database, which it shares with Genesys Pulse Collector.

See the Genesys Pulse User's Guide to learn how to operate this user interface. This is also accessible from within the software, when you click help.

High Availability

See the Genesys Pulse Hardware Sizing and Performance Information for an example of High Available (HA) architecture.

High Availability Failure Scenarios

The following failure scenarios apply to HA configurations.

Stat Server

The Backup Stat Servers take over immediately. There are an HA pair of Stat Servers running for each business. The user experience is minimal due to the Primary and Backup configuration (two active chains), therefore the stats are not reset.

Genesys Pulse Collector

Genesys Pulse Collectors run in Active and Active mode.

Genesys Pulse reads the data from the Genesys Pulse Collector to which it is connected.

When a Genesys Pulse Collector fails, Genesys Pulse either cannot serve users and the load balancer

routes users to the other Genesys Pulse or, if you configured WebDAV, Genesys Pulse reads snapshots from the other Genesys Pulse Collector through WebDAV.

This also applies to Genesys Pulse Collector in cluster configuration, only with more nodes.

DB Server

The Backup DB Server takes over immediately. There are a pair of DB Servers for each business running as an HA pair. This failover is invisible to users.

Genesys Pulse Instance

If a Genesys Pulse instance fails, the load balancer is responsible for redirecting the user request to the other Genesys Pulse instances.

If the Genesys Pulse instance fails while the supervisor is connected, the load balancer is still responsible for redirecting the user session to the other Genesys Pulse instances and the login page is displayed. The second active Genesys Pulse instance takes over and no statistics are lost. There are a pair of Genesys Pulse for each business running in Active and Active mode.

Database Failure

To ensure the uninterrupted operation of databases for Genesys Pulse use database cluster solutions. For information about supported databases, see the Genesys Supported Operating Environment Reference.

Genesys Framework Components

Genesys Pulse interacts with several products within the Genesys Framework to provide real-time snapshots of contact center data.

Configuration Server

Configuration Server provides the following data to Genesys Pulse Collector:

- Information about the existence of contact center objects (for example, tenants, agents, places, calling lists, or campaigns)
- Statistical parameters (for example, time ranges, time profiles, filters, and statistical types)
- Information about changes to contact center objects (for example, a deleted agent, a renamed queue, or a new Agent Group).

Genesys Pulse Collector uses this data to provide content for Genesys Pulse.

Both Genesys Pulse and Genesys Pulse Collector connect to Configuration Server in order to retrieve their own configuration.

Stat Server

Stat Server tracks information about customer interaction networks (contact center, enterprise-wide, or multi-enterprise telephony and computer networks). Stat Server also converts the data accumulated for directory numbers (DNs), agents, agent groups, and non-telephony-specific object types, such as email and chat sessions, into statistically useful information, and passes these calculations to other software applications that request data.

As a client of Stat Server, Genesys Pulse Collector requests statistics for objects belonging to particular reports. Stat Server supplies the information about interactions that the objects can handle and noninteraction-related data about objects themselves (for example, agent status). Genesys Pulse Collector returns information for all `stattypes` that are configured in the options of all Stat Servers to which Genesys Pulse Collector is connected.

Refer to the Stat Server User's Guide and Stat Server Deployment Guide for information about Stat Server.

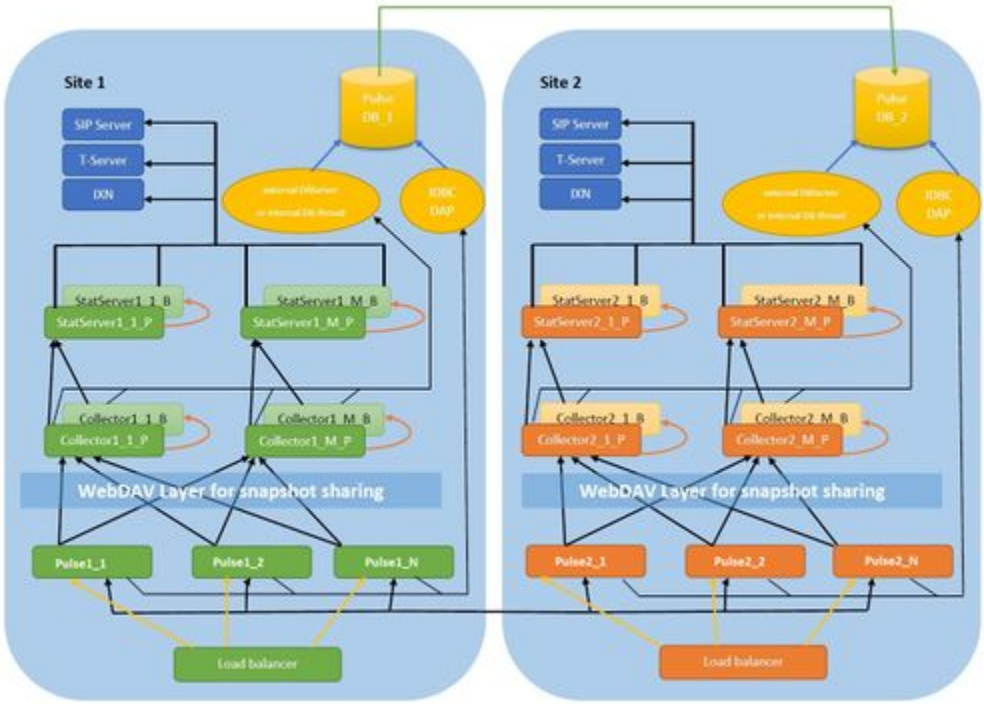
DB Server

DB Server is the Genesys component that handles database requests from multiple client processes. DB Server provides a single interface from the clients to a variety of database engines, including Microsoft SQL Server, Oracle and PostgreSQL. As a client of DB Server, Genesys Pulse Collector reads information about active widgets and updates the layout statuses, when layout status changes occur within the Genesys Pulse Collector.

Refer to Management Framework for information about DB Server.

Multi-site Architecture

Multi-site Architecture allows Genesys Pulse to work in multiple data centers with traffic minimized between sites for Disaster Recovery (DR) purposes.



- Genesys Pulse, Genesys Pulse Collector, Stat Server are recommended to be on their own dedicated hosts (VMs), but all can be deployed on the same host.
- Scale Genesys Pulse initially using vertical scaling (change the host/VM specs). After that, you can add more Genesys Pulse VM pairs to both Site 1 and Site 2 clusters.
- Each Genesys Pulse Collector can handle about 300K statistics with a 10-second refresh rate opened on Stat Server. Adjust this amount based on the refresh rate.

  Note: If you use Changes-based notifications for anything other than Agent Current State, you might have a significant number of notifications for each of these statistics per second. You must configure the sensitivity to account for these additional notifications in sizing.
- Use a dedicated Stat Server for each Genesys Pulse Collector:
  - Each Genesys Pulse Collector supports only a single connection to Stat Server (HA pair).
  - In a cluster configuration, all Stat Servers used for all Genesys Pulse Collectors must be connected to the same sources, such as T-Servers, SIP Servers, and Interaction (IXN) Servers.
- In a cluster configuration, the statistics are dynamically distributed across all Primary Genesys Pulse Collectors. You can have up to 256 Primary Genesys Pulse Collectors for each Site (‘M’ in the name of Genesys Pulse Collectors on diagram).

- In a cluster configuration, each Genesys Pulse instance can render the statistics and views across all Genesys Pulse Collectors on the same Site.
- For High Availability (HA), you need a pair of Genesys Pulse Collectors.
- Genesys Pulse, Genesys Pulse Collector, Stat Server, and DB Server should exist and be operational on both Sites (Site 1: Main region and Site 2: DR region).
- Both Genesys Pulse and Genesys Pulse Collector application objects should have the `site` option set in pulse or collector sections respectively, representing the operational site name (to form Genesys Pulse - Genesys Pulse Collector pairs on each Site).
- Genesys Pulse databases are replicated from Site 1 to Site 2 through periodic backup and restore processes using the capabilities of RDBMS (for example, one-way log shipping or bi-directional regional replication using Postgres BDR).
- If a one-way database replication is used (Site 1 to Site 2), Genesys Pulse on the DR site functions and serves data in read-only mode. You cannot edit dashboards, widgets, or templates on the second site.
- Use WebDAV to let Genesys Pulse pull snapshot data from an instance of Genesys Pulse Collector that is not installed on the same host. Genesys supports Lighttpd http server with lighttpd-mod-webdav on UNIX and IIS on Windows.
- You can use the Site 2 for customer validation of changes and capabilities, before placing them into production (Site 1). In this case the Site 2 cluster needs to be active only before an upgrade is performed.

Deploy Genesys Pulse

You must deploy matching 5-digit releases of Genesys Pulse and Genesys Pulse Collector.

Refer to the following topics to check prerequisites and perform necessary installation/configuration steps:

1. Confirm Software requirements.
2. Prepare the Genesys Pulse Database.
3. Configure a Third-Party Data Source (if you are planning to use the third-party data) or Deploy Genesys Pulse Collector and:
   - Optional: Configure Multi-Language Environments.
   - Optional: Configure Embedded DB Server.
4. Deploy **Genesys Pulse**.
5. Optional: Deploy Genesys Pulse Pull Collector
6. Optional: Deploy Cluster Configuration.
7. Optional: Configure System Security.
8. Configure the Stat Server Application object.
9. Optional: Configure Quick Widget Updates.
10. Optional: Configure links between GAX and Genesys Pulse.
11. Configure User Access.
12. Logging In.

Confirm Software Requirements

You deploy Genesys Pulse on a web application server to be accessed using a web browser through the Genesys Pulse user interface.

The following are prerequisites for Genesys Pulse deployment:

- Genesys DB Server must be release 8.1.300.05 or higher.
- Your Relational Database Management System (RDBMS) must be up and running. Genesys Pulse supports the following relational database platforms:
  - Microsoft SQL Server 2012, 2012 Cluster, 2014, 2016, 2017 (starting with release 9.0.003), 2019 (starting with release 9.0.005) Enterprise Edition, 2019 Cluster (starting with release 9.0.006)
  - Oracle 11g, 12c, 12c RAC, 18c and 18c RAC (starting with release 9.0.003), 19c and 19c RAC (starting with release 9.0.005)
  - PostgreSQL 9.x, 10 (starting with release 9.0.001), 12.x (starting with release 9.0.005), PostgreSQL 13.x (starting with release 9.0.008), PostgreSQL 16.x (starting with release 9.0.009)

> ### Important
>
> **Discontinued relational database support**:
> Support of PostgreSQL versions prior to 9.5 is discontinued as of Genesys Pulse release 9.0.005
> Support of Oracle 11g is discontinued as of Genesys Pulse release 9.0.005
> Support of Oracle 12c (including RAC) is discontinued as of Genesys Pulse 9.0.006
> Support of Oracle 18 (including RAC) is discontinued as of Genesys Pulse 9.0.006.04
> Support of Microsoft SQL Server 2014 is discontinued as of Genesys Pulse 9.0.003

- The computer on which you install Genesys Pulse must be running one of the following:
  - Windows Server 2012, 2016, 2019 (starting with release 9.0.003.03)
    Microsoft Visual C++ Redistributable Packages for Visual Studio 2013 and 2017 (starting with release 9.0.001) must be installed on Genesys Pulse Collector host. See https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads for more details. These packages are included in Genesys Pulse Collector IP and usually are installed automatically.
  - Red Hat Enterprise Linux 7.x (64-bit), Red Hat Enterprise Linux 8.x (64-bit) (starting with release 9.0.004) with Updates from RHN enabled, CentOS 7.x (64-bit) (starting with release 9.0.001), CentOS 8.x (64-bit) (starting with release 9.0.004).

> ### Important
>
> **Discontinued Operating Systems support**:
> CentOS Linux 7 is not supported by Genesys Pulse Collector as of 9.0.008.03. However, it remains supported by Genesys Pulse.
> Red Hat Enterprise Linux 7 is not supported by Genesys Pulse Collector as of 9.0.008.03. However, it remains supported by Genesys Pulse.

- Any required packages must be available on the operating system. The Installer provides a notification if it discovers that any packages are missing, such as compatibility packs for the 32-bit platform. These packages have generic names, for example: compat-glibc*, compat-libstdc+*, libstdc+, compat-openssl10, libcrypt*, libxcrypt*, and libnsl*.
- Beginning with release 9.0.007.00, Pulse Collector requires the OpenSSL 1.1 runtime libraries package from the Extra Packages for Enterprise Linux (EPEL) package repository on Red Hat Enterprise Linux 7.
  To install the package:
  1. Enable the EPEL repository on your system. For specific instructions for each Operating System, see Extra Packages for Enterprise Linux (EPEL).
  2. Execute the following command to install OpenSSL 1.1 runtime libraries: `sudo yum install openssl11-libs`
     No additional packages are required on RHEL8 and UBI8.

> **Important**
>
> Both Genesys Pulse Collector and Genesys Pulse must be installed on the same host, unless you use WebDav to share snapshots. Genesys does not support Genesys Pulse deployments that have components on separate hosts without deploying WebDav server.

- Although not required for deployment, you must have Stat Server release 8.5.109 or higher installed for basic operation.
- You must install Java Runtime Environment (JRE):
  - JRE 8 for Pulse 9.0.002 and older
  - JRE 8 or 11 for Pulse 9.0.003 - Pulse 9.0.008
  - JRE 11 for Pulse 9.0.009 and newer

Both Oracle Java SE and OpenJDK (Red Hat's implementation) runtimes are supported.

Prepare the Genesys Pulse Database

In High Availability (HA) configurations, configure both Genesys Pulse applications to use the same database.

Microsoft SQL Server

1. Create a new empty Microsoft SQL Server database.
2. Create a new Microsoft SQL Server user account.
3. Set the new database as default database for the user.
4. Grant the new user sufficient privileges for the new database. User must be able to create database objects and select, insert, update, and delete data in tables.
5. Run the below statements for the Genesys Pulse database (DB). Replace `<Pulse DB>` with the name of the Genesys Pulse DB and make sure that there are no other connections to the Genesys Pulse DB when you run those statements:
   - `ALTER DATABASE <Pulse DB> SET READ_COMMITTED_SNAPSHOT ON;`
   - `ALTER DATABASE <Pulse DB> SET ALLOW_SNAPSHOT_ISOLATION ON;`

Oracle

Create a new user/schema to be used by Genesys Pulse. The user must have `RESOURCE` and `CREATE VIEW` privileges.

PostgreSQL

1. Create a new empty PostgreSQL database
2. Create a new PostgreSQL user account.
3. Set the new database as default database for the user.
4. Grant the new user sufficient privileges for the new database. User must be able to create database objects and select, insert, update, and delete data in tables.

Configure a Third-Party Data Source

Are you planning to use *only* third-party data such as snapshots populated with the REST API? If this is the case, you still need to set up at least one "Dummy" Collector in the Genesys Pulse connections, since options like output folder and output file extension are stored in the Genesys Pulse Collector object.

> **Important**
>
> You need at least one of a Genesys Pulse Collector or a "Dummy" Collector. If you plan to deploy Genesys Pulse Collector, you can skip this step and continue with Deploy Genesys Pulse Collector.

It is simple:

1. Use the regular Collector template to create the "Dummy" Collector object.
2. Update the `output-folder` configuration option in the [`transport-file`] for your local installation. Set the value to a full path of an existing folder. A relative path is valid only when an actual Collector is used.
3. If you want to use a remote host to store data, add a new configuration option, `external-output-folder`, in the [`transport-file`] section. Make sure you set the value to the full network path of an existing folder.

Deploy Genesys Pulse Collector

Genesys Pulse Collector connects directly to Stat Server to collect statistics for contact center objects. Genesys Pulse Collector also accesses the database that stores reporting layouts. Genesys Pulse Collector accesses the database through DB Server which can be either embedded or external. See Configure Embedded DB Server for more details.

Install Genesys Pulse Collector from the Installation Package

Starting with GAX release 9.0.100.56, configure the necessary objects required by Genesys Pulse Collector using the following steps and then continue with More Installation Steps.

1. Create Genesys Pulse Collector Application from Template:
   - In GAX, go to **Environment > Application Templates > New**
   - Click **Import Application** Template and upload the APD Template file (the `Collector.apd` file from the Templates Installation Package directory).
   - Click **Import Metadata** and upload XML Template file (The Collector.xml file from the Templates Installation Package directory).
   - Click **Save**.
2. In GAX, go to **Environment > Applications > New** and create Genesys Pulse Collector Application from the imported template:
   - Specify the Host where Genesys Pulse Collector will be deployed.
   - Set the `Working Directory` and `Command Line` fields to any value, these fields will be updated to correct values automatically during the installation.
3. Install Genesys Pulse Collector:
   - Copy the Genesys Pulse Collector IP to the Genesys Pulse Collector host.
   - Run the `setup.exe` (Windows) or `install.sh` (Linux) installation file.
   - Follow installer prompts to install Genesys Pulse Collector.

Install Genesys Pulse Collector via GAX IP Deployment Wizard (Deprecated)

Begin with the following steps if your GAX is prior to 9.0.1 release and then continue with More Installation Steps.

1. Upload Genesys Pulse Collector Installation Package and Template:
   - In GAX, go to **Administration > Installation Packages** and click on the plus sign.
   - Select **Installation Package Upload (template uploaded separately)** and click **Next**.
   - For **Upload a Package**, select the zipped file that contains the Genesys Pulse Collector Installation Package (Genesys Pulse Collector IP). The zip file should have in its root the files from the IP folder (such as ip_description.xml and read_me.html).
   - For **Upload an XML template**, select the XML Template file (Collector.xml from the Templates Installation Package directory).
   - For **Upload an APD template**, select the APD Template file (Collector.apd from Templates Installation Package directory).
   - Click **Finish**.
2. Deploy the Genesys Pulse Collector Installation Package and Template:
   - Click on the Genesys Pulse Collector Installation Package to open the Properties tab.

> **Important**
> The Genesys Pulse Collector Installation Package status should be Complete.

   - Click on the **Related** icon (configuration icon) and choose **Install** to open the IP Deployment Wizard.
   - Fill required fields and finish the installation.

> **Important**
> The InstallPath should point to an empty folder where you plan to install Genesys Pulse Collector.

More Installation Steps

> **Important**
> Begin with Install Genesys Pulse Collector from the Installation Package (for GAX starting with release 9.0.1) or Install Genesys Pulse Collector via GAX IP Deployment Wizard (for GAX prior to release 9.0.1) and then proceed with the following steps.

1. Optional: To change the default options for Genesys Pulse Collector in GAX, open the Genesys Pulse Collector `Application` object modify the values of configuration options described in Genesys Pulse Collector Application Object.

2. For a high-availability (HA) deployment, repeat the following steps to install the backup instance to another server:
   - Complete Steps 2 and 3 from Install Genesys Pulse Collector from the Installation Package (for GAX starting with release 9.0.1) or Step 2 from Install Genesys Pulse Collector via GAX IP Deployment Wizard (for GAX prior to release 9.0.1).
   - Complete Step 1 from the More Installation Steps above.

   You need a Genesys Pulse Collector for each instance of the Genesys Pulse application. Configure another Genesys Pulse Collector Application object for each Primary and set it as `Backup Server` in the General Tab in the options of the Primary Genesys Pulse Collector application. Choose the `Redundancy Type` to be `Hot Standby`.

   > **Important**
   >
   > **The Genesys Pulse HA configuration changed in release 8.5.104. In release 8.5.103, Genesys Pulse Collectors do not use Primary and Backup servers for HA. Instead, Genesys Pulse mirrors the load between Genesys Pulse Collectors if more than one is configured and connected to Genesys Pulse.**
   > **This means that in HA configurations, each widget is processed by two different Genesys Pulse Collectors. If you have only two Genesys Pulse Collectors, then all widgets are processed by both Genesys Pulse Collectors.**

3. Create a new Database Access Point (DAP), which is necessary for connectivity to the Genesys Pulse database through the DB Server.

   > **Important**
   >
   > Genesys Pulse Collector does not support Windows Authentication with MS SQL Server.
   > Starting with release 9.0.005, support of external DB Server in DAP for Genesys Pulse Collector is deprecated. Configure Genesys Pulse Collector with an embedded DB Server instead.

   - In GAX, edit the DAP application settings, and in the **Connection Type** list, ensure that **Default** is selected.
   - Specify Database Server `Application` object in **DB Server** field for an external DB Server or leave it as `None` to use an embedded version of the DB Server.
   - Specify the connectivity parameters for your RDBMS.
   - To use an external DB Server, select the **Ports** tab to change the default port value to the value of your DB Server port.
4. In the connections of the Genesys Pulse Collector `Application`, add the DAP that is to be used by Genesys Pulse Collector.
5. In the connections of the Genesys Pulse Collector `Application`, add the primary Stat Server application that is to be used by Genesys Pulse Collector.
6. Add the Tenant objects to the Tenants tab for all Genesys Pulse Collectors that you plan to monitor in Genesys Pulse.
7. Deploy Genesys Pulse. In GAX, add the Genesys Pulse Collector `Application` object to the connections of your Genesys Pulse `Application` object.

8. For a load-balanced environment configuration with two Genesys Pulse applications, associate all instances of Genesys Pulse with all primary instances of Genesys Pulse Collector.
9. Make sure to start Genesys Pulse instances before starting Genesys Pulse Collector instances.

Optional: Configure Multi-Language Environments

If you have a multi-language Configuration Server, you must add the following option to the command line of Genesys Pulse Collector

- On Windows platform:

  ```
  -cs_codepage 65001
  ```

  - You may need to edit the Genesys Pulse Collector service command line in the Windows registry.
- On Linux platform:

  ```
  -cs_encoding UTF-8
  ```

> **Important**
>
> Stat Server is compiled to support MBCS (multi-byte character set), not Unicode. Genesys recommends avoiding the usage of character set specific symbols in configuration object names. However, you can still use character set specific symbols for display names and configure a custom display format for objects in Pulse Collector.

Configure Genesys Pulse Collector with an Embedded DB Server

When Genesys Pulse Collector runs an embedded DB Server, Genesys Pulse Collector does not require a separate DB Server instance to connect to Genesys Pulse database.

You can enable an embedded DB Server in Genesys Pulse Collector by setting the **dbthread** option in the **[collector]** section to **yes**.

The Database Access Point application object used by Genesys Pulse Collector should have **None** value specified in the **DB Server** field on the **General** tab.

> **Tip**
>
> Prior to release 9.0.005, configuration of Genesys Pulse Collector with an Embedded DB Server was optional.

Software Requirements

Genesys Pulse Collector uses Genesys DB Clients, which require DBMS clients to be installed. Refer to Environment Settings for more information.

You should have the necessary clients installed if the following are true:

- You had Genesys DB Server installed on the host with Genesys Pulse Collector.
- Genesys DB Server and Genesys Pulse Collector are of the same architecture (32- or 64-bit).
- To use Oracle client 12.x. on Linux, set the Genesys Pulse Collector application option **embedded-dbserver/oracle_name = ./dbclients_next/dbclient_oracle_64**. Only the 64-bit version is supported.

Embedded DB Server Configuration Options

The embedded DB Server produces its own log messages. To control what goes to the log and where the log should be written add the **[log-db]** section to **collector** application options and add all options that usually go to the **[log]** section of the Genesys DB Server application.

If there is no **[log-db]** section specified in **collector** application options, the embedded DB Server log uses options found in **[log]** section, except for the following:

1. The value for the **verbose** option is set to **standard**.

2. The log is written to the file specified in the **[log]** section, with the **-db** suffix added.

Limitations

- For Genesys Pulse Collector release 9.0.001.04 and earlier, Linux deployments running with an embedded DB Server fail to connect to the MS SQL Server database.
- Prior to release 9.0.001, Genesys Pulse Collector did not support Oracle 12 clients. For Genesys Pulse Collector release 9.0.000, in order to connect to an Oracle 12c database, the Oracle 11.2 client has to be installed on the Genesys Pulse Collector host.

Deploy Genesys Pulse

Install Genesys Pulse from the Installation Package

Starting with GAX release 9.0.100.56, configure the necessary objects required by Genesys Pulse using the following steps and then continue with More Installation Steps.

1. Create Genesys Pulse Application from Template:
    1. In GAX, go to **Environment > Application Templates > New**.
    2. Click **Import Metadata** and upload the XML Template file (the `Pulse.xml` file from the Templates Installation Package directory).
    3. Click **Import Application Template** and upload the APD Template file (the `Pulse.apd` file from the Templates Installation Package directory).
    4. Click **Save**.
2. In GAX, go to **Environment > Applications > New** and create the Genesys Pulse Application from the imported template:
    1. Specify the `Host` where Genesys Pulse will be deployed.
    2. Specify the `Working Directory` as a path to the Genesys Pulse installation folder.
    3. Specify the `Command Line` as pulse_startup.bat if Genesys Pulse is deployed on Windows or ./pulse_startup.sh if Genesys Pulse is deployed on Linux.
3. Install Genesys Pulse:
    1. Copy the Genesys Pulse IP to the Genesys Pulse host.
    2. Run the `setup.exe` (Windows) or `install.sh` (Linux) installation file.
    3. Follow installer prompts to install Genesys Pulse.

Install Genesys Pulse via GAX IP Deployment Wizard (Deprecated)

Begin with the following steps if your GAX is prior to 9.0.1 release and then continue with More Installation Steps.

Configure the necessary objects required by Genesys Pulse using GAX.

1. Upload Genesys Pulse Installation Package and Template:

1. In GAX, go to **Administration > Installation Packages** and click on the plus sign.
2. Select **Installation Package Upload (template uploaded separately)** and click **Next**.
   For **Upload a Package**, select the zipped file that contains the Genesys Pulse Installation Package (Genesys Pulse IP). The zip file should have in its root the files from the IP folder (such as ip_description.xml and read_me.html).
3. For **Upload an XML template**, select the XML Template file (Pulse.xml from the Templates Installation Package directory).
4. For **Upload an APD template**, select the APD Template file (Pulse.apd from Templates Installation Package directory).
5. Click **Finish**.

2. Create Genesys Pulse Application from Template:
   1. In GAX, go to **Environment > Application templates> New** and Import Genesys Pulse Application Template. (The `Pulse.apd` file from the Templates Installation Package directory).
   2. In GAX, go to **Environment> Applications > New** and create Genesys Pulse Application from the imported template:
      - specify the `Host` where Genesys Pulse will be deployed
      - specify the `Working Directory` as a path to the Genesys Pulse installation folder
      - specify the `Command Line` as `pulse_startup.bat` if Genesys Pulse is deployed on Windows or `./pulse_startup.sh` if Genesys Pulse is deployed on Linux.
3. Deploy the Genesys Pulse Installation Package:
   1. Click on the Genesys Pulse Installation Package to open the **Properties** tab.

      > **Important**
      > The Genesys Pulse Installation Package status should be complete.

   2. Click on the related icon and choose **Install** to open the IP Deployment Wizard.
   3. Fill in required fields:
      - Select a host where Genesys Pulse is deployed and the Genesys Pulse `Application` object.
      - Set the `InstallPath` to the same value as the `Working Directory` specified in the Genesys Pulse `Application` object.

More Installation Steps

> **Important**

Begin with Install Genesys Pulse from the Installation Package (for GAX starting with release 9.0.1) or Install Genesys Pulse via GAX IP Deployment Wizard (for GAX prior to release 9.0.1) and then proceed with the following steps.

1. In the Genesys Pulse installation folder, create the `conf` folder with the pulse.properties file:
   - `host= <Configuration Server host>`
   - `backup_host=<Backup Configuration Server host>(optional)`
   - `port= <Configuration Server port>`
   - `backup_port=<Backup Configuration Server port> (optional)`
   - `app=<Genesys Pulse application name>`
   - `http_port = <http port for Genesys Pulse to start on>`
   - `root_url = < the root URL to access Genesys Pulse> (optional, default value is /pulse)`
2. Navigate to **Configuration > Environment > Application**, select the Genesys Pulse Application, **Permissions** tab and configure the SYSTEM account to have Read, Update, and Execute permissions.
3. Optional: Configure options in the [pulse] section on the `Application Options` tab for the Genesys Pulse `Application` object:

   Important

   During startup, Genesys Pulse looks for all options that are required for its operation and adds them if they are not explicitly configured. If a required option is either not configured or specifies an invalid option value, Genesys Pulse uses the option's default value.

4. Create a new Database Access Point (DAP), which is necessary for connectivity to the Genesys Pulse database:
   1. Enter a **Database Name**.
   2. On the **General** tab, set the **Connection Type** to **JDBC** and specify the following field values:
      - Role = Main
      - Debug = false
      - JDBC Query Timeout: 15
      - Case Conversion: any
   3. On the **Ports** tab, change the value of the default port to the value of your RDBMS port.
   4. (Optional). A connection to the Genesys Pulse database can be specified as the JDBC URL in the jdbc_url option.

> **Tip**
>
> If you need to use virtual ip/host(s) to access Genesys Pulse Database and you do not want to create Host objects for them, then use any existing host on the Server Info tab but configure the correct ip/host through the jdbc_url option in your DAP object.

5. Add the Database Access Point to the connections of your Genesys Pulse `Application` object.
6. Add the Genesys Pulse Collector `Application` object to the connections of your Genesys Pulse `Application` object.
7. For a High Availability (HA) deployment:
    1. For each Genesys Pulse `Application` object:
        - Complete Steps 2 and 3 from Install Genesys Pulse from the Installation Package (for GAX starting with release 9.0.1) or Install Genesys Pulse via GAX IP Deployment Wizard (for GAX prior to release 9.0.1).
        - Complete More Installation Steps 1, 2, 3, 5, and 6 above.
    2. Each Genesys Pulse application should have the other Genesys Pulse application in its connections and there must be no relation between the Genesys Pulse applications as Backup Servers.

> **Important**
>
> All Genesys Pulse instances must use the same database.

8. From the user account created when you prepared the Genesys Pulse database, execute the SQL statements in the appropriate initialization script deployed during installation (scripts folder)—either:
    - pulse_init_mssql.sql
    - pulse_init_oracle.sql
    - pulse_init_postgres.sql
9. Start all Genesys Pulse instances.
10. To access Genesys Pulse (see the Logging In page for more information), enter the following URL in the address bar of the browser:

    ```
    http://<Host name>:<http_port>/<root_url>/
    ```

    where:
    - <Host name> is the name of the computer on which you installed Genesys Pulse.
    - <http_port> is the port number defined in the the pulse.properties file.
    - <root_url> is the URL defined in the the `pulse.properties` file.

Optional: Deploy Genesys Pulse Pull Collector

Genesys Pulse Pull Collector is installed during the Genesys Pulse installation into the <Genesys Pulse installation folder>/pull-collector directory. It is used to push third-party applications' (such as Genesys Knowledge Center, Genesys Web Engagement, Universal Contact Server) data into Genesys Pulse for visualization. Genesys Pulse Pull Collector processes widgets which are created from third-party templates and stores snapshots into the Genesys Pulse Collector output folder. Genesys Pulse Pull Collector should be running on the same host as Genesys Pulse Collector.

Configuration

1. Make sure that Genesys Pulse Collector is configured correctly.
2. Find the `application.conf` file in the `<Genesys Pulse installation folder>/pull-collector` directory and configure the following parameters:
   - **database**
     - **type** – type of Genesys Pulse database; valid values: `postgres`, `mssql`, and `oracle`
     - **username** – username to connect to Pulse DB
     - **password** – password to connect to Pulse DB
     - **database.url** – URL to connect to Pulse DB:
       - For Oracle it should be like `jdbc:oracle:thin:@//<host>:<port>/<dbname>`
       - For PostgreSQL it should be like `jdbc:postgresql://<host>:<port>/<dbname>`
       - For MSSQL it should be like `jdbc:sqlserver://<host>:<port>;databaseName=<dbname>`
     - **database.poller.interval** – database polling interval in seconds
       Default value: 15
   - **output**
     - **directory** – output directory for snapshots. Must point to the same directory as configured for regular Genesys Pulse Collector, started on the same host.
     - **file.ext** – snapshot files extension. Must be the same as the value of the **[transport-file]/output** option in Genesys Pulse Collector application object.
     - **compression.type** – file compression method. Must be the same as the value of the **[transport-file]/compression-method** option in Genesys Pulse Collector application object.
   - **layout**

  - **type** – type of layouts for which Genesys Pulse Pull Collector collects data.
    Default value: "ltOTHER"
    Valid values: "ltGENERIC", "ltOTHER"
  - **other_layout_type** – sub-type of layouts, effective together with layout.type = ltOTHER
    Valid values: any string
- **application.name** – name of the Genesys Pulse application object

HA and Cluster Configuration

Genesys Pulse Pull Collector instance should be running per each Genesys Pulse Collector instance in the HA pair or in the Cluster.

> **Important**
> There is no load distribution between Genesys Pulse Pull Collectors in the cluster configuration. Each Genesys Pulse Pull Collector instance processes the same amount of data.

How To Run Genesys Pulse Pull Collector

Run as a Service on Windows

To create a Windows service, perform the following steps:

1. Navigate to the `pull-collector` installation directory, which contains the `pull_collector_service.ini` and `pull_collector_service.exe` files.
2. Edit the `pull_collector_service.ini` service configuration file by replacing 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 pull-collector start=auto
   binPath="\"<path_to_pull_collector_service.exe>\" -service pull-collector -immediate"
   ```

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

   ```
   sc.exe start pull-collector
   sc.exe stop pull-collector
   ```

Run as a Service on Linux

Create a separate `systemd` service configuration file for Genesys Pulse Pull Collector.

For example, create the systemd service configuration file /etc/systemd/system/pulse-pullcollector.service with the following content:

```
[Unit]
Description=Genesys Pulse Pull Collector

[Service]
ExecStart=<absolute path to java executable> -jar path/to/installation/pulse-pull-
collector.jar
WorkingDirectory=<absolute path to the Genesys Pulse Pull Collector directory>

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

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

Optional: Deploy Genesys Pulse Cluster Configuration

For high availability (HA), you must enable HA for each Genesys Pulse Collector instance in the cluster. They are the Primary Genesys Pulse Collectors in HA configuration. To have cluster configuration, perform the following steps:

1. For each Primary Genesys Pulse Collector, configure a new Genesys Pulse Collector Application object and set it as the Backup Server in the General tab in the options of the Primary Genesys Pulse Collector application. Choose Hot Standby for the Redundancy Type.
2. For each configured Primary Genesys Pulse Collector, install another collector on different host. This host can be in different data center than the Primary Genesys Pulse Collector if the data centers are connected using a low latency high bandwidth network that uses the Genesys Pulse Collector configuration.
3. For Genesys Pulse Collector HA configuration starting with release 8.5.104.xx, all Genesys Pulse instances in the cluster must be connected to the Primary Genesys Pulse Collectors.

> **Important**
>
> You must restart all Genesys Pulse Collector instances and Genesys Pulse instances after changing the configuration.

4. Review the cluster architecture and sizing information in the Sizing Guide.
5. Install Genesys Pulse on one node.
6. Set up two or more Genesys Pulse Collector instances. All instances of Genesys Pulse Collector can be installed on the same node with Genesys Pulse or on any number of remote nodes.
   - If one or more Genesys Pulse Collectors are installed on remote nodes then WebDAV HTTP server should be installed on these nodes too. See the WebDAV server configuration instructions.

     > **Important**
     >
     > **Remotes nodes and Genesys Pulse HA**—If there is a second Genesys Pulse node and it is installed on a separate machine then all nodes with Genesys Pulse Collectors are considered remote, because for the second Genesys Pulse node, they are remote.

   - Configure options for Genesys Pulse Collector instances installed on a remote nodes and then restart every Genesys Pulse Collector instance:
     - Set the **output-url** option in the **[transport-webdav]** section to

       ```
       http://<WebDAV host>/<path to folder with snapshots>
       ```

     - Set the **heartbeat-url** option in the **[transport-webdav]** section to

```
http://<WebDAV host>/<path to heartbeat folder>
```

Configuring System Security

Genesys Pulse has features that enhance your system security. This section discusses Genesys Pulse security features and describes how to configure them.

TLS: Configuring the Genesys Pulse Database

You must configure your Oracle, Microsoft SQL, or PostgreSQL server to use TLS. In addition to the appropriate procedure below, refer to the documentation that came with your database for information on how to use TLS security.

Oracle

1. Set up the Genesys Pulse database (for Oracle).
2. Configure Oracle as described in the related database guides, and configure a TCPS listener. See Management Framework documentation for more information.
3. Configure the **jdbc_url** option in the **[pulse]** section of your Genesys Pulse DAP application object:
   `jdbc_url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=<Database host>)(PORT=<Database port>))(CONNECT_DATA=(SERVICE_NAME=<Database Service name>)))`

SSL connection using TLS v1.2
JDK 7 and JDK 8 releases support TLS v1.2 protocol. Other protocols, such as TLS v1.1, TLS v1, SSL v3, and SSL v2 have security vulnerabilities. Genesys recommends to use the latest standard TLS v1.2 version and use more secure SSL cipher suites.

The correct JDBC Thin driver is required in order to use TLS v1.2.

> **Important**
> If you are using the `ojdbc8.jar` from 12.2.0.1 version then you are all set.
> If you are using the 12.1.0.2 JDBC driver, you need to either download the 12.1.0.2 patched driver or apply the patch (that allows TLS v1.2) for the bug 19030178. The patch allows TLS v1.2 but does not enable it by default. So, you must set the `oracle.net.ssl_version=1.2` property. This property can be set either as the system property (using -D) or through the datasource properties.

MS SQL

1. Set up the Genesys Pulse database (for MS SQL).

2. Configure Microsoft SQL Server as described in the related database guides. See Management Framework documentation for more information.
3. Configure the **jdbc_url** option in the **[pulse]** section of your Genesys Pulse DAP application object:
   `jdbc_url=jdbc:sqlserver://<Database host>:<Database port>;databaseName=<Database name>;encrypt=true;trustServerCertificate=false`

PostgreSQL

1. Set up the Genesys Pulse database (for PostgreSQL).
2. Configure PostgreSQL as described in the related database guides. See Management Framework documentation for more information.
3. Configure the **jdbc_url** option in the **[pulse]** section of your Genesys Pulse DAP application object:
   `jdbc_url=jdbc:postgresql://<Database host>:<Database port>/<Database name>?ssl=true&sslcert=<path to certificate>&sslkey=<path to key>&sslrootcert=<path to root cerificate>&sslmode=verify-full`

> **Important**
>
> The certificate key must be in the `pkcs8` format. You can use the `openssl` utility to convert the key:
> `openssl pkcs8 -topk8 -nocrypt -inform PEM -outform DER -in <server.key> -out <server-key.pk8>`

TLS: Configuring Genesys Pulse

Genesys Pulse supports Transport Layer Security (TLS) communications between Genesys Pulse server and client-side connections using the web browser interface.

Genesys Pulse can support connections through HTTP or HTTPS simultaneously. This is controlled by the **supported_protocol** parameter (valid values are `http`, `https`, or `both`) in the `pulse.properties` file, located in the `conf` directory of your Genesys Pulse installation.

> **Important**
>
> Starting with release 9.0.006, Genesys Pulse does not support hard-coded encryption keys for passwords:
>
> - Genesys Pulse no longer supports the encrypted form of the `keystore_password` property for the unofficial HTTPS activation workaround. The `keystore_password` property is not encrypted and can be passed as an environment variable or command line argument.

- Genesys Pulse no longer sends user passwords in encoded form. Genesys Pulse must be used with HTTPS enabled or behind HTTPS-enabled proxy or load balancer to protect users credentials.

Starting with Genesys Pulse release 9.0.006, use the following steps to set up HTTPS:

1. Create a keystore file (PKCS12 and JKS keystore types are supported) with the private key and certificate for the Genesys Pulse server using one of the following ways:
   - Create a keystore with a self-signed certificate by executing the following command:

     ```
     keytool -keystore <full keystore path> -alias pulse -genkey -keyalg RSA
     ```

     and enter the required information as prompted.
   - Export the existing certificate to the PKCS12 format and import into an existing keystore:

     ```
     openssl pkcs12 -export -in <src certificate> -inkey <src key> -out <pkcs12 keystore
     path> -name pulse
     keytool -importkeystore -srckeystore <pkcs12 keystore path> -srcstoretype pkcs12
     -destkeystore <full keystore path>
     ```
2. Define the **https_port**, **supported_protocol**, and **keystore_path** parameters in the `pulse.properties` file. The default **https_port** is 443.
   - `https_port=8443`
   - `supported_protocol=https` or `both`
   - `keystore_path=`full path to the location of the keystore
3. Choose how the keystore password is provided to Genesys Pulse. Genesys Pulse application supports the following options:
   - Add the command line argument to java command (by changing the `pulse_startup.sh` (on Linux) or `pulse_startup.bat` (on Windows) file or by adding this option to the `JAVA_OPTS` environment variable):
     `-Dcom.genesys.pulse.keystore.password=<password>`
   - Set the password as a value of the `KEYSTORE_PASSWORD` environment variable.
   - Add the `keystore_password` property with your password to the `pulse.properties` file.

   [+] Example. How to obscure password with base64 encoding

   Linux:
   I. Obscure the password:

   ```
   echo mypassword | base64 > <path_to_key>/key.txt
   ```

   II. Use obscured password in the startup script:

   ```
   java -Dcom.genesys.pulse.keystore.password=$(base64 -d <path_to_key>/key.txt) -jar
   ./pulse.war
   ```

   Or set the environment variable that Genesys Pulse can read:

```
export KEYSTORE_PASSWORD=$(base64 -d <path_to_key>/key.txt)
```

Windows:
I. Obscure the password:

```
echo mypassword > key.txt
certutil -encode key.txt tmp.b64 && findstr /v /c:- tmp.b64 > key.txt
del tmp.b64
```

II. Use obscured password in the startup script:

```
certutil -decode key.txt tmp.b64
set /P password=<tmp.b64
del tmp.b64
java -Dcom.genesys.pulse.keystore.password=%password% -jar ./pulse.war
```

Or set the environment variable that Genesys Pulse can read:

```
certutil -decode key.txt tmp.b64
set /P password=<tmp.b64
del tmp.b64
setx KEYSTORE_PASSWORD %password%
```

4. (optional) Choose how the individual key password is provided to Genesys Pulse. If not provided, keystore password will be used.
   Genesys Pulse application supports the following options:
   - Add the command line argument to java command (by changing the `pulse_startup.sh` (on Linux) or `pulse_startup.bat` (on Windows) file or by adding this option to the `JAVA_OPTS` environment variable): `-Dcom.genesys.pulse.key.password=<password>`
   - Set the password as a value of the `KEY_PASSWORD` environment variable.
   - Add the `key_password` property with your password to the `pulse.properties` file.
5. Start Genesys Pulse.

Cipher Lists

> **Important**
> Genesys does not support the tools mentioned in this section. They are used here only to illustrate.

The Java Runtime Environment (JRE) provides the cipher suites that Pulse uses. Before enabling TLS, please ensure JRE supports actual cipher suites that your browser can accept. Otherwise, refer to your JRE distribution documentation to enable ciphers accepted by your browser.

- You can access a list of ciphers accepted by your browser here.
- For a list ciphers provided by JRE:
  1. Download and extract Ciphers.zip to a temporary location.
  2. Execute the following command from that location:

```
java Ciphers
```

- If a cipher suite or protocol that you require is provided by JRE but not enabled by default in Pulse, there is a mechanism that lets you enable the cipher suite or protocol during startup. Be aware that you must specify in preference order.
- If a vulnerability is discovered in a cipher or protocol, or if it is considered too weak to use, you can exclude it during Pulse startup.
- If a cipher suite or protocol is both included and excluded as part of the same configuration, it is disabled (excludes will always win).

> **Important**
>
> By default, the following weak cipher suites are disabled in Pulse: SSL_*, TLS_RSA_*, MD5, SHA, and SHA1-based. You can still accept the security risk and enable them by removing them from the excluded cipher suites. Once a weak cipher is enabled, you will need to restart Pulse and you will see a warning in the Pulse log:
> `Weak cipher suite <suite name> enabled for Server.`

The table below lists the available parameters in pulse.properties (regex values supported).

Parameter	Description	Default (if not specified)
setIncludeProtocols	Specifies one or more protocols to be supported.	NA
setIncludeCipherSuites	Specifies one or more cipher suites to be used for encoding data.	NA
setExcludeProtocols	Specifies one or more protocols that are not to be supported.	`"SSL", "SSLv2", "SSLv2Hello", "SSLv3"`
setExcludeCipherSuites	Specifies one or more cipher suites that are not to be used for encoding data.	`"^.*_(MD5|SHA|SHA1)$","^TLS_RSA_.*$","^SSL_.*$","^.*_NULL_`
addExcludeProtocols	Specifies additional protocols that are not to be supported.	NA
addExcludeCipherSuites	Specifies additional cipher suites that are not to be used for encoding data.	NA

Example:

```
host=pulse-host
port=8080
app=pulse
...
setExcludeCipherSuites=^.*_(MD5|SHA|SHA1)$
setIncludeCipherSuites=TLS_ECDHE_.*,TLS_RSA_.*
```

HTTP Strict Transport Security (HSTS)

HSTS is disabled by default and you can enable it by setting the **enable_hsts** option in the `pulse.properties` file to `true`. Once HSTS is enabled, Genesys Pulse prevents downgrading of encrypted HTTPS connection to unencrypted HTTP. It is implemented by sending a response header record from the server indicating that compliant Web browsers or other HTTP client programs must use HTTPS and they must display the appropriate confirmation message or an error message in the browser console.

Securely embed Pulse into iframe

Starting with 9.0.008.03 Pulse allows embedding into iframe. Use the following step to allow embedding:

- Make sure Pulse is configured over HTTPS
- Set appropriate options in `pulse.properties` file:
  - `disable_xframe_options=true`
  - `session_samesite=none` (or `lax` if the iframe source is on the same domain as Pulse)
  - `supported_protocol=https`, or `supported_protocol=both` and `enable_hsts=true`
  - edit `content_security_policy` property to contain `frame-ancestors` directive with valid parents that may embed, for example: `content_security_policy=frame-ancestors 'self' https://example.com;`

Configure the Stat Server `Application` object

> **Important**
>
> Stat Server requires Java Environment configuration for several templates to function properly. See the Stat Server Deployment Guide for details. Without this additional configuration, statistics in some Queue templates (Email Queue Activity, eServices Queue KPIs, IWD Queue Activity) will not work.

1. Set the `accept-clients-in-backup-mode` option in the [statserver] section on **Application Options** tab to yes for both the primary and backup Stat Server `Application` objects.

   > **Important**
   >
   > This option is required even if there is no backup application specified for the Stat Server application.

2. Update Stat Types specified in the pulse_statistics.cfg file within the scripts folder in Genesys Pulse installation to use the particular social media that is configured in your eServices solution (`facebook` is used in default file version). See eServices documentation for more details.
3. You must import pulse_statistics.cfg file to both the primary and backup Stat Server `Application` objects to create the stattypes that Genesys Pulse should monitor. To import the file:
   a. Click **More** on the **Application Options** tab.
   b. Select **Import**, uncheck **Override**, and browse the file.

   > **Important**
   >
   > - To calculate the **% Ready Time** in the Queue KPIs template, set the `queue-use-pseudo-actions` option in the `[statserver]` section of Stat Server `Application` object to `false`.
   > - Some Stat Server filters used in Genesys Pulse templates rely on certain user data or reasons attached to the call (for example, `VoiceCall_No_Wait`, `ReasonLunch`, and others that have `PairExists` in their definition).
   >
   >   You may need to adjust definitions of these filters to use `Attached Data` or `Reasons` according to your environment or adjust your routing strategies or desktop application to attach data used by those statistics. Otherwise, statistics that rely on these filters will show 0 (zero).
   > - The StatServer `subscribe-for-all-ixn-server-events` configuration option default value is `no`. To use the Chat Agent Activity template, you must set the option value to `yes`.
   > - The StatServer `suppress-agent-status-updates-for-ixn-server` configuration option default value is `no`. You must set the option value to `yes` to avoid unnecessary `EventCurrentAgentStatus` notifications

> from being sent to Interaction (IXN) Servers.

4. To use the Chat Agent Activity template, set the Stat Server option subscribe-for-all-ixn-server-events to yes.
5. Restart both Stat Server applications.

Quick Widget Updates

Genesys Pulse supports quick updates only for CurrentStatus and ExtendedCurrentState statistics to prevent a high performance load this causes on Stat Server and Genesys Pulse.

You are responsible to validate that your environment can handle the load in production caused by quick updates.

You can do Quick Updates using one of the following modalities:

- Aeron transport. See the section below for more information.
- RabbitMQ. See the Optional: Deploy RabbitMQ for quick widget updates page for more information.

> **Tip**
>
> - Quick Widget Updates and Advanced Alerting Capabilities require independent instances of Aeron Media Driver. Make sure that values of the Genesys Pulse Collector options **driver-directory** and **endpoints** do not interfere with settings of `Aeron Media Driver` used by `Advanced Alerting` services.
> - Advanced Alerting services support Aeron transport only. **Note:** For the successful usage of the embedded Aeron transport you do not need to implement `Advanced Alerting` services.

Aeron Transport

Starting with release 9.0, Genesys Pulse Collector includes Aeron transport.
Starting with release 9.0.001, Aeron transport is supported on Windows.

Aeron transport can be used, when Genesys Pulse and Genesys Pulse Collector are installed on the same host, by specifying options in the transport-aeron section of your Genesys Pulse Collector application. You will need to restart Genesys Pulse Collector and Genesys Pulse to apply configuration changes.

> **Important**
>
> - Aeron transport can be enabled only if each Genesys Pulse instance in your configuration is connected to Genesys Pulse Collector installed on the same host. No additional software installation is needed.
> - Genesys strongly recommends to put the Aeron driver folder on the local drive. The

Aeron driver folder on the network share can cause Genesys Pulse Collector to crash.

Optional: Deploy RabbitMQ for quick widget updates

Use RabbitMQ for quick widget updates.

> **Important**
>
> Genesys Pulse supports quick updates for CurrentStatus and ExtendedCurrentState statistics only to prevent a high performance load this causes on Stat Server and Genesys Pulse.
>
> You are responsible to validate that your environment can handle the load in production caused by quick updates.

Decide on Configuration

1. Determine whether to use RabbitMQ with Genesys Pulse using Cluster or Single-Node configurations. Genesys Pulse supports using these configurations only with RabbitMQ.
    1. **Cluster configuration** — Use at least the same number of RabbitMQ instances as the number of Genesys Pulse Collector applications. RabbitMQ can run on any host: either the one where Genesys Pulse runs or any other host accessible over reliable network.
        The default configuration should be one RabbitMQ instance running on every host where Genesys Pulse Collector runs. For example, Primary host (`host1`) and Backup host (`host2`).
    2. **Single node configuration** — This simple configuration uses a single RabbitMQ instance running either on a host with Genesys Pulse Collector or any other host accessible over reliable network.
        Note: If this RabbitMQ instance fails or the whole host fails, quick widget Updates stop working. If you choose this configuration, you still need to go through all steps to deploy RabbitMQ unless clearly stated otherwise. The host with RabbitMQ is called host1 in the remainder of this deployment.

All Configurations

1. To use RabbitMQ to work with Genesys Pulse, use the following software versions:
    - RabbitMQ server version 3.11.5.
    - The version of Erlang compatible with that version of RabbitMQ.
    - Use identical versions of Erlang on all hosts running RabbitMQ.

> **Tip**
>
> We specify the lowest acceptable version here, but any later versions should work unless their RNs specify that there are some backward compatibility changes.

2. On every host, install Erlang and RabbitMQ. Follow the instructions at the RabbitMQ site, to install on your brand of Linux or on Windows.
3. After installing, change RabbitMQ server's option, channel_max, to `0`.
4. On every host, start RabbitMQ server as described on the page of installation instructions you used. This will generate an Erlang cookie file, required for using RabbitMQ in a cluster.

Cluster Configuration

Refer to RabbitMQ's Clustering Guide to set up the RabbitMQ cluster on your hosts.

Configuration for Pulse

1. Create a vhost with a name `'/pulse'` for Genesys Pulse. If you create a vhost with a different name, you must specify it in Genesys Pulse Collector configuration options. On any of the hosts run the following command:
   ```
   rabbitmqctl add_vhost /pulse
   ```
2. Create a user for Genesys Pulse with name **'pulse'** and password **'pulse'**. If you create a user with a different name and password, you must specify them in Genesys Pulse Collector configuration options. On any of the hosts run the following command:
   ```
   rabbitmqctl add_user pulse pulse
   ```
3. Grant user access to vhost. Here is how to grant user **'pulse'** access to vhost '**/pulse'** with permissions to create exchanges with names starting with **'pulse'**. On any of the hosts run the following command:
   ```
   rabbitmqctl set_permissions -p /pulse pulse "^pulse.*" ".*" ".*"
   ```
4. To configure Genesys Pulse Collector to work with RabbitMQ you need to have `transport-rabbitmq` section configured in options of Genesys Pulse Collector application object.
   1. Add or update configuration options in the [`transport-rabbitmq`] section.
   2. Restart Genesys Pulse Collector and Genesys Pulse to apply changes.

> **Important**
>
> **RabbitMQ memory and disc usage**
>
> RabbitMQ instances should not store any Genesys Pulse application data on disc, so the disc usage is insignificant unless the message queue for any of the Genesys Pulse applications grows too large. To ensure that the message queue does not grow too big in some exceptional cases there is a limit on queue length in

Genesys Pulse, which is controlled by option `max-queue-length` in section `transport-rabbitmq` of Genesys Pulse Collector application options.

Use the value of this option to estimate possible RabbitMQ memory usage. To roughly estimate upper limit of possible memory usage use this formula:

```
<Max memory usage> = <RabbitMQ idle memory usage> + 3 * ( max-queue-length * <Average size of a delta snapshot> * 4 )
```

For example, for an average change in message size of 10KB, RabbitMQ idle memory usage of 100MB, and `max-queue-length` of 1000, we obtain 220MB of memory usage.

Tip

If Genesys Pulse and Genesys Pulse Collector are installed on the same host, instead of RabbitMQ you can use embedded Aeron transport by specifying options in the transport-aeron section of your Genesys Pulse Collector application. **Note:** For the successful usage of the embedded Aeron transport you do not need to implement Advanced Alerting services.
Starting with release 9.0.001, Genesys Pulse and Genesys Pulse Collector support Aeron transport configured on Windows.

Optional: Enable OAuth SSO using GWS

> **Warning**
>
> SSO using on-premise GWS is not available.

You can set up Genesys Pulse to use OAuth 2.0 protocol for user authorization. OAuth, short for *open authorization*, is an open standard protocol that allows secure API authorization without requiring the user to provide their credentials to a third party. You can read more about OAuth here.

When OAuth is enabled, users can log in to Genesys Pulse with accounts from Genesys Web Services (GWS).

To enable the OAuth 2.0 authentication mechanism follow these steps:

1. Enable token-based authentication between Genesys Configuration Server and Genesys Pulse:
    1. Configure the following configuration options in the `[system]` section of Configuration Server to which Pulse is connected:
        - **token-authentication-mode** - Set this option to enable token-based authentication on all ports.
        - **token-preambula** - (optional) Specifies the preamble tag that is affixed to the start of the password token. Default value is {PXZ}. Genesys recommends that you do not configure this option and use the default value, unless you have a specific reason to override the default value.
        - **token-uuid** - (optional) Specifies a UUID to be used to generate a symmetric key. If not specified, Configuration Server uses a value generated internally by the primary master Configuration Server for the particular Configuration Database.

        For detailed information about these options, refer to the *Configuration Server Configuration Options* chapter of the Framework Configuration Options Reference Manual.
    2. Configure the following configuration options in the `[general]` section of every Pulse application object:
        - **confserv_trusted** - Set this option to `true` to enable token-based authentication.
        - **token_life_in_minutes** - (optional) Specifies the length of time for which the token will be valid; once the token has expired, connection requests with this token will be rejected. Genesys recommends that you use the default value for this option, unless you have a specific reason to override it.
2. Configure the following configuration options in the `[oauth]` section of every Pulse application object:
    - **client_id** - client ID registered in GWS with authorization code grant type.
    - **password** - client secret registered in GWS.
    - externally available GWS endpoints for connection from the browser:

- **user_logout_url** - endpoint for logout, for example, http://gws-usw1.genhtcc.com/auth/v3/sign-out.
- **user_auth_url** - endpoint for user authorization, for example, http://gws-usw1.genhtcc.com/auth/v3/oauth/authorize.
- In case of multi site configuration, you may need to add additional prefixed ***_user_auth_url** and ***_user_logout_url** options, for example:

  [sites]

  pulse-usw1.genhtcc.com=site1

  pulse-use1.genhtcc.com=site2

  [oauth]

  site1_user_auth_url=https://api-g1-usw1.genhtcc.com/auth/v3/oauth/authorize

  site2_user_auth_url=https://api-g1-use1.genhtcc.com/auth/v3/oauth/authorize

  site1_user_logout_url=https://api-g1-usw1.genhtcc.com/auth/v3/sign-out

  site2_user_logout_url=https://api-g1-use1.genhtcc.com/auth/v3/sign-out

- internally available GWS endpoints for connection from the Pulse server:
  - **access_token_url** - endpoint for retrieval of the access token, for example, http://gws-usw1.genhtcc.com/auth/v3/oauth/token.
  - **user_info_url** - endpoint for retrieval of the user info, for example, http://gws-usw1.genhtcc.com/auth/v3/userinfo.
- In a case where Pulse is running behind a proxy with HTTPS termination you may need to set the `[oauth]\force_https_for_redirect` option to `true`.

3. Enable OAuth for every Pulse application object: `[security]\auth_type = oauth`.

Optional: Enable SAML SSO

> **Important**
>
> - The feature is available for Pulse version 9.0.009 and later. If you are using Pulse 8.5, please refer to Using Single Sign On (SSO).
> - SLO (Single Logout) feature is not supported, therefore please use the `saml_landing_page` property to specify page for redirect upon logout from Pulse.

You can set up Genesys Pulse to use SAML2 protocol for user authorization.

To enable the SAML2 authentication mechanism follow these steps:

1. Enable token-based authentication between Genesys Configuration Server and Genesys Pulse:
   1. Configure the following configuration options in the **system** section of Configuration Server to which Pulse is connected:
      - **token-authentication-mode** - Set this option to enable token-based authentication on all ports.
      - **token-preambula** - (optional) Specifies the preamble tag that is affixed to the start of the password token. The default value is {PXZ}. Genesys recommends that you do not configure this option and use the default value, unless you have a specific reason to override the default value.
      - **token-uuid** - (optional) Specifies a UUID to be used to generate a symmetric key. If not specified, Configuration Server uses a value generated internally by the primary master Configuration Server for the particular Configuration Database.

      For detailed information about these options, refer to the *Configuration Server Configuration Options* chapter of the Framework Configuration Options Reference Manual.
   2. Configure the following configuration options in the **general** section of every Pulse application object:
      - **confserv_trusted** - Set this option to true to enable token-based authentication.
      - **token_life_in_minutes** - (optional) Specifies the length of time for which the token will be valid; once the token has expired, connection requests with this token will be rejected. Genesys recommends that you use the default value for this option, unless you have a specific reason to override it.
2. Make sure HTTPS is configured in Pulse, see Configuring System Security for more details.
3. Edit pulse.properties file, located in the conf directory of your Genesys Pulse installation to enable SAML2 authentication by specifying the following properties:

- `saml=true`
- `session_securecookies=true`
- `session_samesite=none`
- `saml_entityid` - Your unique ID for IdP.
- `saml_idp_metadata` - Location of path to Identity Provider (IdP) metadata.
- `saml_landing_page` - URL for redirect upon logout from Pulse.
- `saml_keystore` - Location of path to Keystore containing signing key certificate.

- `saml_keystore_password` - Password for Keystore containing signing key certificate.
- `saml_key_name` - The name of signing key certificate in the Keystore.
- `saml_key_password` - (optional) Password for singing key certificate. Pulse will be using password for Keystore if not specified.
- `saml_external_userid` - (optional) SAML attribute name used to extract username from the assertions. Pulse will be using “uid“ if not specified.

Important

You can use following environment variables instead of passwords in pulse.properties:

- SAML_KEYSTORE_PASSWORD
- SAML_KEY_PASSWORD

It’s also possible to use java command line arguments:

- `-Dcom.genesys.pulse.saml.keystore.password`
- `-Dcom.genesys.pulse.saml.key.password`

4. Start Pulse and navigate to https://<pulse_host>:<pulse_https_port>/<pulse_context>/saml2/service-provider-metadata/<entityid> to download Service Provider metadata (SP).
5. Register downloaded Service Provider metadata in your Identity Provider (IdP).

Optional: Configure links between GAX and Genesys Pulse

For proper links between GAX and Genesys Pulse perform the following steps:

1. Copy the `link-to-pulse.jar` file located in the Genesys Pulse installation folder to `<GAX dir>/plug-ins` and `<GAX dir>/webapp/WEB-INF/lib` folders.
2. In GAX, navigate to `Configuration > Environment > Applications`.
3. Select the GAX `Application` object.
4. Open the `Application Options` tab.
5. Create the new **[link-to-pulse]** section.
6. Configure the **$default** option in the **[link-to-pulse]** section of your GAX application with the URL to Genesys Pulse as the value. See Genesys Pulse Configuration Options for more details.
7. Save application.
8. Select the Genesys Pulse `Application` object.
9. Open the `Application Options` tab.
10. Create the new **[link-to-gax]** section.
11. Configure the **$default** option in the **[link-to-gax]** section of your Genesys Pulse application with the URL to GAX as the value. See Genesys Pulse Configuration Options for more details.
12. Save application.
13. Restart GAX and Genesys Pulse applications.

Configure User Access

Roles

In GAX, navigate to **Configuration > Accounts > Roles** and create a new Role object to provide access to Genesys Pulse functionality.

> **Important**
>
> When creating a new Role, a dialog box with Role Template Selection appears. Do not select a template (leave the selection empty). Press **OK**.

Define the privileges granted by the Role on the `Assigned Privileges` tab in the Genesys Pulse section.

Privilege Details		
Name	**Description**	**Prerequisite**
Pulse View Dashboard	User has read-only access to launched dashboards without the ability to expand widgets to tab (includes stay on dashboard ability, which means that user's sessions will not expire and users will not be automatically logged out due to inactivity timeout).	
Pulse View Dashboard Restricted	User has read-only access to launched dashboards without the ability to expand widgets to tab (excludes stay on dashboard ability).	
Pulse Manage Tabs	User can launch, close or customize dashboards and wallboards and expand widgets to tab.	Pulse View Dashboard or Pulse View Dashboard Restricted
Pulse Edit Widget Display	User can modify display options of widgets.	Pulse Manage Tabs
Pulse Manage Widgets	User can create, remove, or modify all widget options.	Pulse Edit Widget Display
Pulse Manage Shared Dashboards	User can create, remove, import, export, or modify shared dashboards.	Pulse Manage Widgets
Pulse Manage Templates	User can create, remove, import, export, or modify templates.	Pulse Manage Widgets

Privilege Details		
Pulse Add Dashboards Without Limit	User can add any number of dashboards. Applicable if the **[pulse]\max_tabs_per_user** option is set in your Genesys Pulse application object or the **max_tabs_per_user** is set for the Access Groups Object.	Pulse Manage Tabs
Pulse Add Widgets Without Limit	User can add any number of widgets. Applicable if the **[pulse]\max_widgets_per_user** option is set in your Genesys Pulse application object or the **max_widgets_per_user** is set for the Access Groups Object.	Pulse Manage Widgets
Pulse Manage Users	User can manage other users' widgets and dashboards.	Pulse Manage Tabs, Pulse Edit Widget Display, Pulse Manage Widgets
Pulse Manually Bind Collectors	User can manually select Pulse Collectors to process particular widgets or templates.	Pulse Manage Tabs, Pulse Edit Widget Display, Pulse Manage Widgets

The following privileges are for users or third-party applications that connect to Genesys Pulse using a Web API.

- **Pulse Read All Layouts**—View all Genesys Pulse layouts using a Web API and switch off filtration of rows by access in snapshot.
- **Pulse Write Snapshot**—Upload layout snapshots using a Web API.

Genesys Pulse privileges use GAX logic—the high-level privileges do not include lower-level privileges. For example, to configure role with full control access, you have to assign all privileges to it.

For role members to create Widgets, you must assign **Pulse View Dashboard** (or **Pulse View Dashboard Restricted**), **Pulse Manage Tabs**, and **Pulse Edit Widget Display** in addition to **Pulse Manage Widgets**.

> **Important**
>
> You must assign at least the **Pulse View Dashboard** or **Pulse View Dashboard Restricted** privilege to each Role object.

Assign the Role to Persons and Access Groups in the Role Members section as required.

Permissions

Provide appropriate permissions (**Read** and **Execute**) on the following objects:

1. Genesys Pulse client application object (usually called `default`. The name is controlled by the Genesys Pulse option **client_app_name** in the **[general]** section):
    1. Select **Configuration**.
    2. On the **Environment** pane, click **Applications**.
    3. From the **Applications** list, open the required application (usually called **default**) to view its properties.
    4. Select the **Permissions** tab.
    5. Add the **Person** or **Access Group** containing this user.
    6. Add required permissions: **Read** and **Execute** are required to log in to Genesys Pulse.
2. Tenant Environment:
    1. Select **Configuration**.
    2. On the **Environment** pane, click **Tenants**.
    3. From the **Tenants** list, open the Environment tenant to view its properties.
    4. Select the **Permissions** tab.
    5. Add the **Person** or **Access Group** containing this user.
    6. Add required permissions: **Read** and **Execute** are required to log in to Genesys Pulse.
3. User's own tenant:
    1. Select **Configuration**.
    2. On the **Environment** pane, click **Tenants**.
    3. From the **Tenants** list, open the User's own tenant to view its properties.
    4. Select the **Permissions** tab.
    5. Add the **Person** or **Access Group** containing this user.
    6. Add required permissions: **Read** and **Execute** on tenant are required to log in to Genesys Pulse.

Logging In

The Genesys Pulse web-based interface runs on a web application server. It is loaded into your browser each time when you open the website where you installed Genesys Pulse. You then log in.

> **Important**
>
> Genesys Pulse supports the use of blank passwords only if Configuration Server is configured to allow blank passwords. Refer to the *Genesys Security Deployment Guide* for information about using blank passwords.

Procedure: Logging in to Genesys Pulse

Prerequisites

- Configuration Server is installed and running.
- An instance of a Genesys Pulse Application object is connected to Configuration Server and running.
- Your browser and its windows are set to a resolution of 1024x768 or greater. If you are working in 1024x768, maximize the browser.
- The user logging in must have Read permission to their own `User` object and `Read` and `Execute` permissions on the Genesys Pulse client application object (usually called `default`, controlled by the Genesys Pulse option **client_app_name** in the **[general]** section). Refer to the *Genesys Security Deployment Guide* for information about permissions. Genesys Pulse respects read-write permissions that are set for Environments and Tenants. You can only access those objects that you have permission to see.

> **Tip**
>
> - Genesys Pulse supports the two latest releases of Google Chrome, Apple Safari, Microsoft Edge, the latest release of Firefox ESR.

Steps

1. Start Genesys Pulse.

2. Open a web browser.
3. Enter the following URL in the address bar of the browser:
   `http://<Host name>:<http_port>/<root_url>/`
   where:
   - <Host name> is the name of the computer on which you installed Genesys Pulse.
   - <http_port> is the port number defined in the the pulse.properties file.
   - <root_url> is the URL defined in the the `pulse.properties` file.
4. Log in to Genesys Pulse with your assigned user name and password, and click **Log in**.

> **Important**
>
> Each instance of Genesys Pulse is associated with a single instance of Management Framework. Configuration Server and Port selection is not required during login, nor is it possible to select it.

Access to Genesys Pulse and its functionality is protected by user permissions and Role-Based Access Control. If you get a permissions error when you try to log in to Genesys Pulse or use any of its functionality, you probably do not have the appropriate permissions or role privileges. Refer to the *Genesys Security Deployment Guide* for more information about permissions and Role-Based Access Control, including how to set up appropriate permissions and role privileges.

Deploy Language Packs

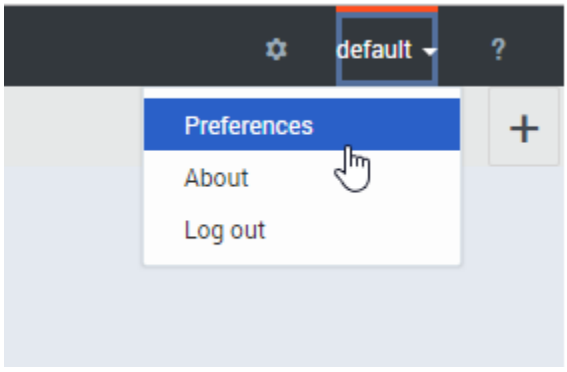
You have to install the Genesys Pulse Language Pack directly on the server by executing the Genesys Pulse Language Pack installation procedure from the Genesys Pulse Language Pack installation package.

> **Important**
> Do not deploy Language Packs using **Administration > Installation Packages**.

1. Copy the Genesys Pulse Language Pack IP to the Genesys Pulse host.
2. Run the **setup.exe** (Windows) or **install.sh** (Linux) installation file.
3. Follow the installer prompts to install the Genesys Pulse Language Pack.
4. If the installation procedure fails to find the Genesys Pulse installation, you can manually copy the Genesys Pulse Language Pack file.
5. After installation, find **pulse-<locale>** (<locale> is a language identificator) jar file in the root directory of the Genesys Pulse installation and copy to:
   - Linux: <GAX root>/plug-ins (if the directory exists) and <GAX root>/webapp/WEB-INF/lib; **<Pulse root>/plug-ins (if the directory exists) and <Pulse root>/webapp/WEB-INF/lib**
   - Windows:<GAX root>\plug-ins (if the directory exists) and <GAX root>\webapp\WEB-INF\lib; **<Pulse root>\plug-ins (if the directory exists) and <Pulse root>\webapp\WEB-INF\lib**
6. Restart GAX and Genesys Pulse.

To use the installed language in Genesys Pulse:

- For release 9.0.003 and prior, select the preferred language in GAX preferences menu. See the Genesys Administrator Extension Help for details.
- Starting with release 9.0.004, select the preferred language in Genesys Pulse preferences menu. See image below.



Preferred Language

For release 9.0.003, Genesys Pulse respects the preferred language, specified in browsers settings. However, the language specified in `User` or `System` preferences via GAX has higher priority than browser settings in all Genesys Pulse interfaces except the login page, where only browser's settings are effective.

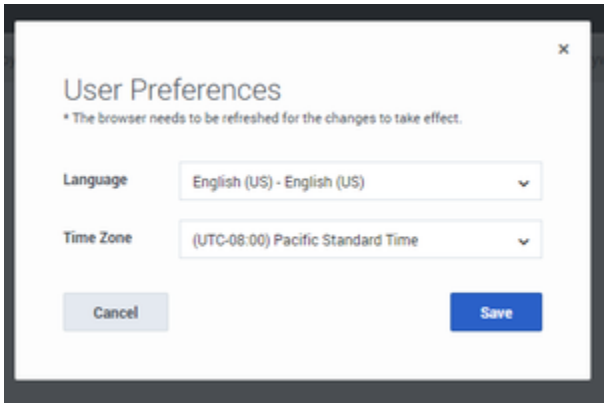
Starting with release 9.0.004, Genesys Pulse respects the preferred language, specified in the browser settings, if the language is set to `Auto` in the Genesys Pulse user preferences menu.

Browser settings are always effective on the Genesys Pulse login page. When the Language Pack that corresponds to the language specified in the browser settings is not available, English language is used.

For backward compatibility purposes Genesys Pulse uses language and time zone preferences that are specified via GAX:

- User Preferences
- Or System Preferences, in case User Preferences have never been set for the user.

Once user preferences are changed via Genesys Pulse, user preferences specified via GAX are no longer respected.



See also Genesys Pulse User Preferences.

{{NoteFormat| Prior to release 9.0.006.00, Genesys Pulse is not capable of detecting the Internet Explorer 11 language, specified in the "Settings / Internet Options / Languages" settings window. Instead, the Windows display language is used. Internet Explorer 11 support is discontinued in Pulse release 9.0.008.02

Genesys Pulse Configuration Options

Configuration options for Genesys Pulse are set in Application objects in GAX and in the pulse.properties file under the `conf` folder inside the Genesys Pulse installation folder.

Application Objects Options

The application templates for Genesys Pulse might contain other configuration options that are not described in this chapter. These options must remain set to the default values that are provided in the application templates. Changing these values might cause unexpected behavior.

You are not required to configure any options to start Genesys Pulse. Genesys Pulse supplies default values for all options that it requires to function.

GAX Application Object

[link-to-pulse] Section

$default
Default Value: No default value
Valid Values: string url
Changes Take Effect: After restart
Specifies the correct link for a Genesys Pulse menu item in the GAX navigation bar. If specified, the Genesys Pulse menu item is added to the Dashboard section of the GAX navigation bar.
Example: http://<pulse host>:<pulse port>/<pulse context>

Genesys Pulse Application Object

[general] Section

client_app_name
Default Value: default
Valid Values: A valid name of an application object of the Configuration Manager type .
Changes Take Effect: After restart
Specifies the name of the client application. Genesys Pulse requires a client application object to enable the access control of a browser-based interface.

inactivity_timeout
Default Value: 600
Valid Values: any positive number
Changes Take Effect: Immediately, for users that logged into Genesys Pulse after the option's value was changed
Specifies the time, in seconds, after the latest user's activity before Genesys Pulse forces logout of this user. A negative value deactivates this timer.
Note: This option has no effect on users with the Pulse View Dashboard privilege.

session_timeout
Default Value: 900
Valid Values: any number
Changes Take Effect: Immediately, for users that logged into Genesys Pulse after the option's value was changed
Specifies the time, in seconds, after the latest request from a user before Genesys Pulse closes the session. The value of this option must be greater than the value of the **inactivity_timeout** option.

confserv_trusted
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
Enables token-based authentication with Configuration Server.

token_life_in_minutes
Default Value: 1440
Valid Values: Any positive integer value
Changes Take Effect: Immediately
Specifies the life of the password token used in token-based authentication. Genesys recommends that you use the default value, unless you have a specific reason to override it.

[link-to-gax] Section

$default
Default Value: No default value
Valid Values: string url
Changes Take Effect: After restart
Specifies the correct link for a GAX menu item in the Genesys Pulse navigation bar. If not specified, the GAX menu item in the navigation bar leads to the same Genesys Pulse instance.
Example: http://<gax host>:<gax port>/<gax context>

[log] Section

Most of the Genesys Pulse log options are inherited from the Common Log Options section of the *Framework Configuration Options Reference Manual*. For more information about logging, see Log4j2 Configuration with the Application Template Application Block in the *Platform SDK Developer's Guide*.

all
Default Value: No default value
Valid Values:

- stdout—Log events are sent to the Standard output (stdout).
- stderr—Log events are sent to the Standard error output (stderr).
- memory—Log events are sent to the memory output on the local disk. This output is the safest in terms of the application performance.

- network—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores log events in the Log Database.

  Setting the all log-level option to network enables Data Sourcer to send log events of Standard, Interaction, and Trace levels to Message Server. Log events of Debug level are neither sent to Message Server nor stored in the Log Database.

- [filename]—Log events are stored in a file with the specified name. If you do not specify a path, the file is created in the application's working directory.

Changes Take Effect: Immediately
Specifies the outputs to which an application sends all log events. You must separate log-output types with commas when you configure more than one output type.
For example, all = stdout, logfile
Notes:

- To ease the troubleshooting process, consider using unique names for log files that different applications generate.
- Relative file paths must begin with `./`

buffering
Default Value: false
Valid Values: true, false
Changes Take Effect: Immediately
Specifies whether the operating system file buffering is on or off. This option applies only to stderr and stdout output. Setting this option to `true` increases output performance.
Note: When you enable buffering, log messages might appear in the log after a delay.

custom-message-format
Default Value: [%-11.11thread] %-20.20logger{1} <%X{requestId}> %m %exception{full}%n
Valid Values: `Log4j` or `Log4j2` pattern formats
Changes Take Effect: Immediately
When the **message-format** option is set to `custom`, the value of the **custom-message-format** option is used to format the log message pattern. The **custom-message-format** option provides the LMS-style log level and the predefined messages prefix containing timestamp per the **time-format/time-convert** option settings.

debug
Default Value: No default value
Valid Values (log output types):

- stdout—Log events are sent to the Standard output (stdout).
- stderr—Log events are sent to the Standard error output (stderr).
- memory—Log events are sent to the memory output on the local disk. This output is the safest in terms of the application performance.
- network—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores log events in the Log Database.
- [filename]—Log events are stored in a file with the specified name. If you do not specify a path, the file is created in the application's working directory.

Changes Take Effect: Immediately

Specifies the outputs to which the log events of the Debug level and higher (that is, log events of the Standard, Interaction, Trace, and Debug levels) are sent. The log output types must be separated by a comma when more than one output is configured.
For example, debug= stdout, logfile

enable_audit_trail
Default Value: false
Valid Values: false, true
Changes Take Effect: After restart
Specifies whether audit trail records are added to the log file. When this option is enabled, the log captures user operations with Dashboards, Wallboards, Widgets and Widget Templates. User operations are logged with the [AUDIT] marker.

expire
Default Value: 10
Valid Values:

- false—No expiration; all generated segments are stored.
- <number>[file]—Sets the maximum number of log files to store. Specify a number from 1–1000.
- <number> day—Sets the maximum number of days before log files are deleted. Specify a number from 1–100.

Changes Take Effect: Immediately
Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number of files (segments) or days before the files are removed. This option is ignored if log output is not configured to be sent to a log file.

interaction
Default Value: No default value
Valid Values (log output types):

- stdout—Log events are sent to the Standard output (stdout).
- stderr—Log events are sent to the Standard error output (stderr).
- memory—Log events are sent to the memory output on the local disk. This output is the safest in terms of the application performance.
- network—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores log events in the Log Database.
- [filename]—Log events are stored in a file with the specified name. If you do not specify a path, the file is created in the application's working directory.

Changes Take Effect: Immediately
Specifies the outputs to which the log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are sent. The log outputs must be separated by a comma when more than one output is configured.
For example, interaction= stdout, logfile

message-format

Default Value: medium
Valid Values:

- short—An application uses compressed headers when writing log records into its log file.
- medium—An application uses medium size headers when writing log records into its log file.
- full—An application uses complete headers when writing log records into its log file.
- shortcsv—An application uses compressed headers with comma delimiter when writing log records into its log file.
- shorttsv—An application uses compressed headers with tab char delimiter when writing log records into its log file.
- shortdsv—An application uses compressed headers with delimiter, which is specified by the value of the message-header-delimiter option.
- custom—An application uses custom log messages format, which is specified by the value of the output-pattern or the custom-message-format option.

Changes Take Effect: Immediately
Specifies the format of log record headers that an application uses when writing logs in the log file. Using compressed log record headers improves application performance and reduces the log file size. With the value set to `short`:

- A header of the log file or the log file segment contains information about the application (such as the application name, application type, host type, and time zone), whereas single log records within the file or segment omit this information.
- A log message priority is abbreviated to Std, Int, Trc, or Dbg, for Standard, Interaction, Trace, or Debug messages, respectively.
- The message ID does not contain the prefix GCTI or the application type ID.

message-header-delimiter
Default Value: |
Valid Values: String
Changes Take Effect: Immediately
When the **message-format** option is set to `shortdsv`, the value of the **message-header-delimiter** option is used as the log message delimiter.

output-pattern
Default Value: %d{dd.MM.yyyy HH:mm:ss}| %-5.5p | %-45.80t | %-30.1000c{1} %m %ex%n
Valid Values: `Log4j` or `Log4j2` pattern formats
Changes Take Effect: Immediately
When the **message-format** option is set to `custom`, the value of the **output-pattern** option is used as the log message pattern.

segment
Default Value: 100 MB
Valid Values:

- false—No segmentation allowed.

- <number> KB or <number>—Sets the maximum segment size in kilobytes. The minimum segment size is 100 KB.
- <number> MB—Sets the maximum segment size, in megabytes.
- <number> hr—Sets the number of hours for which the segment stays open. The minimum number is 1 hour.

Changes Take Effect: Immediately
Specifies if there is a segmentation limit for a log file. If there is, this option sets the unit of measurement along with the maximum size. If the current log segment exceeds the size set by this option, the current file is closed and a new file is created. This option is ignored if the log output is not configured to be sent to a log file.

standard
Default Value: No default value
Valid Values (log output types):

- stdout—Log events are sent to the Standard output (stdout).
- stderr—Log events are sent to the Standard error output (stderr).
- memory—Log events are sent to the memory output on the local disk. This output is the safest in terms of the application performance.
- network—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores log events in the Log Database.
- [filename]—Log events are stored in a file with the specified name. If you do not specify a path, the file is created in the application's working directory.

Changes Take Effect: Immediately
Specifies the outputs to which the log events of the Standard level are sent. The log output types must be separated by a comma when more than one output is configured.
For example, standard = stdout, logfile

time-convert
Default Value: local
Valid Values:

- local—The time of log record generation is expressed as a local time, based on the time zone and any seasonal adjustments. Time zone information of the application's host computer is used.
- utc—The time of the log record generation is expressed as Coordinated Universal Time (UTC).

Changes Take Effect: Immediately
Specifies the system in which an application calculates the log record time when generating a log file. The time is converted from the time in seconds since 00:00:00 UTC, January 1, 1970.

time-format
Default Value: time
Valid Values:

- iso8601—The date in the time string is formatted according to the ISO 8601 format. Fractional seconds

are given in milliseconds.

- locale—The time string is formatted according to the system's locale.
- time—The time string is formatted according to "HH:MM:SS.sss" (hours, minutes, seconds, and milliseconds) format.

Changes Take Effect: Immediately
Specifies how to represent, in a log file, the time when an application generates log records. A log record's time field in the ISO 8601 format looks like this: `2001-07-24T04:58:10.123`.

trace
Default Value: No default value
Valid Values (log output types):

- stdout—Log events are sent to the Standard output (stdout).
- stderr—Log events are sent to the Standard error output (stderr).
- memory—Log events are sent to the memory output on the local disk. This output is the safest in terms of the application performance.
- network—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores log events in the Log Database.
  Setting the all log level option to the network output enables an application to send log events of the Standard, Interaction, and Trace levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database.
- [filename]—Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

Changes Take Effect: Immediately
Specifies the outputs to which the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels) are sent. The log output types must be separated by a comma when more than one output is configured.
For example, trace = stdout, logfile

verbose
Default Value: all
Valid Values:

- all—All log events (that is, log events of Standard, Trace, Interaction, and Debug levels) are generated if you set the debug-level option in the statserver section to all.
- debug—The same as all.
- trace—Log events of the Trace and higher levels (that is, log events of Standard, Interaction, and Trace levels) are generated, while log events of the Debug level are not generated.
- interaction—Log events of the Interaction and higher levels (that is, log events of Standard and Interaction levels) are generated, while log events of the Trace and Debug levels are not generated.
- standard—Log events of the Standard level are generated, while log events of the Interaction, Trace, and Debug levels are not generated.
- none—Produces no output.

Changes Take Effect: Immediately
Determines whether a log output is created. If it is, this option specifies the minimum level of log events that are generated. The log-event levels, starting with the highest-priority level, are Standard, Interaction, Trace, and Debug.
Refer to the *Framework Management Layer User's Guide* for more information on the Standard, Interaction, Trace, and Debug log levels.

[oauth] Section

access_token_url
Default Value: No default value
Changes Take Effect: After restart
Specifies the Auth server endpoint for retrieval of the access token. For example, http://gws-usw1.genhtcc.com/auth/v3/oauth/token.

client_id
Default Value: No default value
Changes Take Effect: After restart
Specifies the OAuth2 client ID. For example, `pulse_client`.

password
Default Value: No default value
Changes Take Effect: After restart
Specifies the OAuth2 client secret. For example, `45hegfsyr34rgiuwefhiua`.

user_logout_url
Default Value: No default value
Changes Take Effect: After restart
Specifies the Auth server endpoint for logout. For example, http://gws-usw1.genhtcc.com/auth/v3/sign-out.

user_auth_url
Default Value: No default value
Changes Take Effect: After restart
Specifies the Auth server endpoint for user authorization. For example, http://gws-usw1.genhtcc.com/auth/v3/oauth/authorize.

user_info_url
Default Value: No default value
Changes Take Effect: After restart
Specifies the Auth server endpoint for retrieval of the user info. For example, http://gws-usw1.genhtcc.com/auth/v3/userinfo.

user_logout_redirect_url
Default Value: No default value
Changes Take Effect: After restart
Specifies the custom url for redirecting the user after logout. By default the user will be redirected back to Pulse.

user_logout_global
Default Value: false
Valid Values: true, false

Changes Take Effect: After restart
Enables invalidation for all SSO user sessions on logout.

force_https_for_redirect
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
This option must be set to true when Pulse is running behind an HTTPS termination proxy.

[pulse] Section

alert_refresh_interval
Default Value: 3
Valid Values: any positive number
Changes Take Effect: After the browser page refresh
Specifies, in seconds, the data refresh interval of the Alert Widget.

alert_snooze_timeout
Default Value: 900
Valid Values: any positive number
Changes Take Effect: After the browser page refresh
Specifies, in seconds, the snooze alert duration on the Alert Widget.

cache_expire_timeout
Default Value: 1200
Valid Values: zero or any positive number
Changes Take Effect: After restart
Specifies how long, in seconds, that Genesys Pulse stores the results of the object accessibility check in order to reduce the load on the Configuration Server.

database_delay_on_retry
Default Value: 5
Valid Values: any number, database_retry_duration / database_delay_on_retry must be >= 1
Changes Take Effect: After restart
Introduced: 9.0.008.02
Specifies how long, in seconds, Pulse waits between retries of a failing RDBMS operation.

database_max_active_connections
Default Value: 8
Valid Values: any positive number
Changes Take Effect: After restart
Specifies the database connection pool size, which limits the maximum number of connections to the Genesys Pulse database. If you see large unexpected delays when you update dashboard or widgets, check the number of active database connections, and, if above the maximum, increase this parameter to improve performance.

> Important

> To ensure the proper Genesys Pulse behavior, Genesys recommends to set the value of the **database_max_active_connections** option to 8 or higher number.

database_max_wait_timeout
Default Value: 10
Valid Values: any positive number
Changes Take Effect: After restart
Introduced: 9.0.006.03
The amount of time, in seconds, to wait for a connection. Applicable only to PostgreSQL and Microsoft SQL.

database_query_timeout
Default Value: 300
Valid Values: any positive number
Changes Take Effect: After restart
Introduced: 9.0.006.03
The amount of time, in seconds, to wait for a query to the database to finish.

database_retry_duration
Default Value: 300
Valid Values: any number, database_retry_duration / database_delay_on_retry must be >= 1
Changes Take Effect: After restart
Introduced: 9.0.008.02
Specifies how long, in seconds, Pulse attempts to retry a failing RDBMS operation.

database_validation_query_timeout
Default Value: 10
Valid Values: any positive number
Changes Take Effect: After restart
Introduced: 9.0.006.03
The amount of time, in seconds, to wait for a query to the database to finish when checking a connection.

disable_cfg_xpath_query
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
Introduced: 9.0.007.13
Allow more efficient handling of requests to Config Server when checking user permissions. Enable this option to reduce the load on Config Server. The option works with Config Server versions 8.5.101.21 or higher.

editable_templates
Default Value: false

Valid Values: true, false
Changes Take Effect: After restart
Enables users with appropriate permissions to edit Genesys-provided templates. If false, even an Administrator cannot edit the default templates. Use this option to remove obsolete or unused templates (for example, iWD or Email). Use in conjunction with the install_templates option.

enable_advanced_alerts
Default Value: false
Valid Values: true, false
Changes Take Effect: After the browser page refresh
Enables the ability to configure Advanced Alerts in the user interface.

> **Important**
> This option must be enabled only when Genesys Pulse configuration meets requirements of microservices.

enable_manual_collector_binding
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart, also requires the refresh of your browser page
Enables the ability to manually specify which Genesys Pulse Collector is used to calculate the statistics of a predefined templates or certain widgets. Users should have the Pulse Manually Bind Collectors role assigned in configuration in order to used this ability.

enable_push_changes
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
Introduced: 9.0.008.02
Enables the ability to open Published dashboards or wallboards in new shared mode when all changes made and published by owner are reflected on the dashboard or wallboard automatically.

folder_based_access
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
When this option is enabled, Genesys Pulse does not create Script objects for Widget Templates, Dashboards, and Wallboards when saving them to Configuration Server. Instead, the association with a target directory is stored by Genesys Pulse and Permissions in GAX must be set for the target folder rather than for an individual Script object.

health_expire_timeout
Default Value: 30
Valid Values: 1-300

Changes Take Effect: After restart
Specifies how long, in seconds, Genesys Pulse stores result of previous health check, which includes the heartbeat, DB connection, and Configuration Server connection.

install_templates
Default Value: true
Valid Values: true, false
Changes Take Effect: After restart
Used in conjunction with the editable_templates option, specifies whether Genesys Pulse installs or updates the Genesys-provided templates for the current release when Genesys Pulse starts. In most cases, when editable_templates is true this option should be set to false.
For example, if you set editable_templates option to true, delete or edit some templates and the install_templates option is set to true, then Genesys Pulse restores all original Genesys-provided templates after restart for the current release. Genesys Pulse restores only templates from the current release, not earlier releases, which may be obsolete.

max_tabs_per_user
Default Value: 0
Valid Values: zero or any positive number
Changes Take Effect: After restart
Specifies the maximum number of personal dashboards. A 0 value means users can have unlimited dashboards.

max_widgets_per_user
Default Value: 0
Valid Values: zero or any positive number
Changes Take Effect: After restart
Specifies the maximum number of widgets for all dashboards. This value is the total sum of widgets for each user. A 0 value means users can have unlimited widgets.

nav_bar_items
Default Value: No default value
Valid Values: specially formed JSON string
Changes Take Effect: After the browser page refresh
Allows to embed custom links to any third-party website in the Navigation bar.
For example, to setup the Advisors menu in Genesys Pulse you need to specify the value as a single string:

```
{"id":"nav_adv","name":"Advisors","type":"main-
item","children":[{"id":"nav_adv_cca","name":"Contact Center Advisor","type":"sub-
item","route":"http://<host>/adv/"},{"id":"nav_adv_wfa","name":"Workforce
Advisor","type":"sub-
item","route":"http://<host>/adv/"},{"id":"nav_adv_fa","name":"Frontline
Advisor","type":"sub-item","route":"http://<host>/adv/"}]}
```

where the `route` property is set according to the Advisors access URL in your environment.

site
Default Value: No default value
Valid Values: any string
Changes Take Effect: After restart

Specifies the site name where Genesys Pulse is installed. If specified, Genesys Pulse reads snapshot data only from the connected Genesys Pulse Collectors with the same site option specified in the [collector] section.

snapshot_expire_timeout
Default Value: 24
Valid Values: zero or any positive number
Changes Take Effect: After restart
Specifies how long, in hours, that the snapshot file remains before Genesys Pulse automatically removes it. This setting should be no more than 24 hours, unless you plan to provide enough disk space to store additional snapshots. Set it to 0 to disable automatic removal.

[security] Section

auth_type
Default Value: No default value
Valid Value: oauth
Changes Take Effect: After restart
Enables oauth2 authorization via GWS.

Genesys Pulse Collector Application Object

[collector] Section

dbthread
Default value: no
Valid values: yes, no
Change takes effect: After restart
Enables or disables running of embedded DB Server in Genesys Pulse Collector internal thread.

hostname
Default Value: Empty value
Valid Values: Valid host name
Changes Take Effect: After restart
Specifies the Simple Network Management Protocol (SNMP) host name.

management-port
Default Value: No default value
Valid Values: Positive integers
Warning! No other application should use this port.
Changes Take Effect: After restart
Specifies the TCP/IP port that Genesys Pulse Collector reserves for SNMP Option Management Client connections. If this option is absent or null, a server for Management Client is not created.

Warning! You must specify a value for this option if you are using an SNMP connection. Do not change the value for this option while Genesys Pulse Collector is running.

site
Default Value: No default value
Valid Values: any string
Changes Take Effect: After Genesys Pulse restart
Specifies the site name where Genesys Pulse Collector is installed. If specified, only the Genesys Pulse with the same site option specified in the [pulse] section reads the snapshot data from this Genesys Pulse Collector.

[configuration-monitoring] Section

always-get-vag-members-from-statserver
Default Value: no
Valid Values: yes, no
Changes Take Effect: After restart
Introduced: Genesys Pulse Collector 9.0.006.04
Enables or disables the ability to get members of Virtual Agent Groups (VAGs), for both Stat Server and Configuration Server (skill-only), from Stat Server.

check-layout-presence-timeout
Default Value: 900
Valid Values: 0-3600
Changes Take Effect: After restart
Specifies how often, in seconds, Genesys Pulse Collector checks for the deleted layouts. A zero (0) value completely disables the check.
Note: This defines the minimum timeout between two checks. The actual timeout depends on the database polling cycle, because the check is conducted after finishing subsequent database polling cycle.

db-poll-period
Default Value: 30
Valid Values: 3-3600
Changes Take Effect: After restart
Specifies how often, in seconds, Genesys Pulse Collector obtains updates from the Genesys Pulse database.
Note: Genesys recommends that you set this option to no less than 15 seconds.

excluded-objects-propagation-delay
Default value: 60
Valid Values: 0...3600
Changes Take Effect: Immediately
Specifies the delay in seconds before Genesys Pulse Collector attempts to propagate excluded objects in the affected layouts after an object is deleted. A zero value eliminates the timeout and Genesys Pulse Collector attempts to propagate excluded objects immediately after an object is deleted.

metagroup-contents-recheck-delay
Default Value: 60
Valid Values: 0...3600
Changes Take Effect: Immediately
Specifies the delay in seconds between when Genesys Pulse Collector verifies metagroup contents (such as Agent Group, Place Group, and DN Group) after notification from Configuration Server notifies Genesys Pulse Collector about changes in the contents of the metagroup object. Zero value of this option eliminates timeout and metagroup change is processed immediately.
Note: This configuration option impacts ANY changes in the metagroup (for example, both adding and deleting objects), so new objects added to the metagroup appear in the layout after the specified delay.

new-object-delay
Default Value: 0
Valid Values: 0-86400
Changes Take Effect: Immediately
Specifies the delay, in seconds, between when Genesys Pulse Collector receives notification of a new object in Configuration Server, and when it starts to process this notification. Setting this option to 0 enables Genesys Pulse Collector to process new objects without delays.

ods-wait-timeout
Default Value: 300
Valid Values: 10–3600
Changes Take Effect: After restart
Specifies the time, in seconds, that Genesys Pulse Collector waits before re-checking the Genesys Pulse database for proper initialization.

remove-dynamic-object-delay
Default Value: 60
Valid Values: 0-600
Changes Take Effect: After restart
Specifies the time, in seconds, that Genesys Pulse Collector waits before excluding and closing statistics for the Agent from affected layout that was excluded from the Stat Server-based VAG. A zero value turns off the delay and Genesys Pulse Collector processes changes immediately.

[heartbeat] Section

Important

- Genesys Pulse uses the **output-folder**, **heartbeat-folder**, and **latest-snapshot-output-folder** options when it is configured on the same host as Genesys Pulse Collector or if there is no corresponding external option configured.
- Genesys Pulse uses the **external-*-folder** options (if configured) when it is configured on a different host from Genesys Pulse Collector and the **external-output-folder**

option is specified.

- Genesys Pulse uses the **external-*-url** folder when it is configured on a different host from Genesys Pulse Collector and no **external-output-folder** option is specified.
- Starting with release 9.0.000, the heartbeat file is always updated, therefore the **heartbeat/heartbeat-success-condition** option is discontinued. New command line option **-startup_heartbeat_file** that specifies which heartbeat file to use before Genesys Pulse Collector can read regular heartbeat configuration options from its Application. This option has only one parameter that provides a path to the heartbeat file.

external-heartbeat-folder
Default Value: No default value
Valid Values: Network or locally mounted path to the heartbeat-folder accessible from a remote Genesys Pulse instance.
Changes Take Effect: After restart
Enables a remote Genesys Pulse instance to access the Genesys Pulse Collector heartbeat. This option must be used together with the heartbeat-folder.
Note: Relative file paths must begin with `./`
Deprecated since release 9.0.000, configure Genesys Pulse with WebDAV instead.

external-heartbeat-url
Default Value: No default value
Valid Values: valid HTTP URL to a heartbeat's folder on WebDAV server. For example, http://host1:8080/heartbeat
Changes Take Effect: After restart
Description: Enables a remote Genesys Pulse instance to access heartbeat through WebDAV server. This option must be used together with the heartbeat-folder. The value of this option is ignored by Genesys Pulse in case when Genesys Pulse Collector and Genesys Pulse are configured on the same Host or external-output-folder option is specified.

heartbeat-folder
Default Value: ./output/heartbeat (as provided by template file)
Valid Values: Valid folder path
Changes Take Effect: After restart
Specifies the path in which Genesys Pulse Collector writes the heartbeat file.
Note: Relative file paths must begin with `./`

heartbeat-period
Default Value: 120
Valid Values: 3-3600
Changes Take Effect: After restart
Specifies the period of a heartbeat update, in seconds.

heartbeat-success-condition
Default Value: statserver, snapshot-writer

Valid Values:
One or any combination of the following conditions:

- statserver—Stat Server connection should be available.
- snapshot-writer—The snapshot writer should be producing outputs without error.
- db-poller—The polling of Genesys Pulse database should not fail.
- collector-db—The connection to Genesys Pulse database should be established.

Changes Take Effect: After restart
Discontinued: as of 9.0.000
Specifies a comma-separated list of conditions to be checked for criteria of heartbeat success, which means the heartbeat is updated only if this criteria is met.

[layout-validation] Section

enable-layout-validation
Default Value: yes
Valid Values: yes, no
Changes Take Effect: After restart
Enables or disables layout validation in Genesys Pulse Collector.
When this option is set to no, Genesys Pulse Collector ignores fatal layout definition errors and collects data to the possible extent. Messages of the Warning or Error levels are still being written to the snapshot.

[limits] Section

max-formulas-per-layout
Default Value: 50
Valid Values: 1-100000
Changes Take Effect: After restart
Specifies the maximum number of formula-based statistics for each layout.

max-metagroups-per-layout
Default Value: 50
Valid Values: 1-10000
Changes Take Effect: After restart
Specifies the maximum number of metagroups for each layout.

max-objects-per-layout
Default Value: 100
Valid Values: 1-100000
Changes Take Effect: After restart
Specifies the maximum number of objects for each layout.

max-statistics-per-layout
Default Value: 100

Valid Values: 1-100000
Changes Take Effect: After restart
Specifies the maximum number of statistics for each layout.

> **Important**
>
> Genesys Pulse adds three or more auxiliary statistics to each layout. Make sure to adjust your value accordingly.

[log] Section

Most of the Genesys Pulse Collector log options are inherited from the Common Log Options section of the *Framework Configuration Options Reference Manual*.

all
Default Value: No default value
Valid Values:

- stdout—Log events are sent to the Standard output (stdout).
- stderr—Log events are sent to the Standard error output (stderr).
- network—Log events are sent to Message Server, which can reside anywhere on the network. Message Server stores log events in the Log Database.

  Setting the all log-level option to network enables Data Sourcer to send log events of Standard, Interaction, and Trace levels to Message Server. Log events of Debug level are neither sent to Message Server nor stored in the Log Database.
- memory—Log events are sent to the memory output on the local disk. This output is the safest in terms of the application performance.
- [filename]—Log events are stored in a file with the specified name. If you do not specify a path, the file is created in the application's working directory.

Changes Take Effect: Immediately
Specifies the outputs to which an application sends all log events. You must separate log-output types with commas when you configure more than one output type.
For example, all = stdout, logfile
Notes:

- To ease the troubleshooting process, consider using unique names for log files that different applications generate.
- Relative file paths must begin with `./`

buffering
Default Value: false
Valid Values: true, false
Changes Take Effect: Immediately
Specifies whether the operating system file buffering is on or off. This option applies only to stderr

and stdout output. Setting this option to true increases output performance.
Note: When you enable buffering, log messages might appear in the log after a delay.

collector-log-level
Default Value: Info
Valid Values: Debug, Trace, Info, Warning, Error, Fatal (case insensitive)
Changes Take Effect: Immediately
Defines log level for Genesys Pulse Collector log messages. Messages with severity below this level will not be logged.
Message severity levels:

- Debug—detailed debug messages.
- Trace—detailed informational and progress messages.
- Info—brief informational and progress messages.
- Warning—minor recoverable errors or situations.
- Error—severe, but recoverable errors.
- Fatal—severe, unrecoverable errors.

Note: This option was introduced in Genesys Pulse Collector release 8.5.106. Log output of earlier Genesys Pulse Collector releases corresponds to the current `Trace` log level. For release 8.5.106 or later, the **[log]/verbose** option must be set to `all` in order to see log messages configured in the **collector-log-level** option.

expire
Default Value: false
Valid Values:

- false—No expiration; all generated segments are stored.
- <number>[file]—Sets the maximum number of log files to store. Specify a number from 1–1000.
- <number> day—Sets the maximum number of days before log files are deleted. Specify a number from 1–100.

Changes Take Effect: Immediately
Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number of files (segments) or days before the files are removed. This option is ignored if log output is not configured to be sent to a log file.

segment
Default Value: 10 MB
Valid Values:

- false—No segmentation allowed.
- <number> KB or <number>—Sets the maximum segment size in kilobytes. The minimum segment size is 100 KB.
- <number> MB—Sets the maximum segment size, in megabytes.

- <number> hr—Sets the number of hours for which the segment stays open. The minimum number is 1 hour.

Changes Take Effect: Immediately
Specifies if there is a segmentation limit for a log file. If there is, this option sets the unit of measurement along with the maximum size. If the current log segment exceeds the size set by this option, the current file is closed and a new file is created.

verbose
Default Value: all
Valid Values:

- all—All log events (that is, log events of Standard, Trace, Interaction, and Debug levels) are generated if you set the debug-level option in the statserver section to all.
- debug—The same as all.
- trace—Log events of the Trace and higher levels (that is, log events of Standard, Interaction, and Trace levels) are generated, while log events of the Debug level are not generated.
- interaction—Log events of the Interaction and higher levels (that is, log events of Standard and Interaction levels) are generated, while log events of the Trace and Debug levels are not generated.
- standard—Log events of the Standard level are generated, while log events of the Interaction, Trace, and Debug levels are not generated.
- none—Produces no output.

Changes Take Effect: Immediately
Determines whether a log output is created. If it is, this option specifies the minimum level of log events that are generated. The log-event levels, starting with the highest-priority level, are Standard, Interaction, Trace, and Debug.
Refer to the *Framework Management Layer User's Guide* for more information on the Standard, Trace, Interaction, and Debug log levels.

[object-name-format] Section

You can use a custom format for an object name, which can include a mix of predefined text and additions to the object properties within their actual values with optional width and trimming rules. For details, see "Valid object name format string" and "Object Information".

AccessResource
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Access Resource. For details, see "Object Properties" at the bottom of this section.

ACDPosition
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart

Specifies the object name formatting rule for an object of type DN with subtype ACD Position. For details, see “Object Properties” at the bottom of this section.

ACDQueue
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for the object of the of type DN with subtype ACD Queue. For details, see “Object Properties” at the bottom of this section.

Agent
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for the object of the of type Agent. For details, see “Object Properties” at the bottom of this section.

AgentGroup
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for the object of the of type Agent Group. For details, see “Object Properties” at the bottom of this section.

CallingList
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Calling List. For details, see “Object Properties” at the bottom of this section.

Campaign
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Campaign. For details, see “Object Properties” at the bottom of this section.

CampaignCallingList
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Campaign Calling List. For details, see “Object Properties” at the bottom of this section.

CampaignGroup
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.

Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Campaign Group. For details, see “Object Properties” at the bottom of this section.

Cellular
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Cellular. For details, see “Object Properties” at the bottom of this section.

Chat
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Chat. For details, see “Object Properties” at the bottom of this section.

CoBrowse
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype CoBrowse. For details, see “Object Properties” at the bottom of this section.

CP
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype CP. For details, see “Object Properties” at the bottom of this section.

Data
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Data. For details, see “Object Properties” at the bottom of this section.

DN
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN. For details, see “Object Properties” at the bottom of this section.

DNGroup
Default Value: %ObjectName%

Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN Group. For details, see “Object Properties” at the bottom of this section.

EAPort
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype EA Port. For details, see “Object Properties” at the bottom of this section.

Email
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Email. For details, see “Object Properties” at the bottom of this section.

Extension
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Extension. For details, see “Object Properties” at the bottom of this section.

ExtRoutingPoint
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Ext Routing Point. For details, see “Object Properties” at the bottom of this section.

FAX
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Fax. For details, see “Object Properties” at the bottom of this section.

Mixed
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Mixed. For details, see “Object Properties” at the bottom of this section.

Music

Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Music. For details, see “Object Properties” at the bottom of this section.

Place
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Place. For details, see “Object Properties” at the bottom of this section.

PlaceGroup
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Place Group. For details, see “Object Properties” at the bottom of this section.

RoutingPoint
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Routing Point. For details, see “Object Properties” at the bottom of this section.

RoutingQueue
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Routing Queue. For details, see “Object Properties” at the bottom of this section.

RoutingStrategy
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Routing Strategy. For details, see “Object Properties” at the bottom of this section.

Script
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Script. For details, see “Object Properties” at the bottom of this section.

ServiceNumber
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Service Number. For details, see “Object Properties” at the bottom of this section.

StagingArea
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Script with subtype Staging Area. For details, see “Object Properties” at the bottom of this section.

Switch
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Switch. For details, see “Object Properties” at the bottom of this section.

Tenant
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Tenant. For details, see “Object Properties” at the bottom of this section.

Video
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Video. For details, see “Object Properties” at the bottom of this section.

VirtACDQueue
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Virt ACD Queue. For details, see “Object Properties” at the bottom of this section.

VirtRoutingPoint
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Virt Routing Point. For details, see “Object Properties” at the bottom of this section.

VoiceMail
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Voicemail. For details, see “Object Properties” at the bottom of this section.

VoIP
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype VoIP.

Workbin
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type Script with subtype Workbin. For details, see “Object Properties” at the bottom of this section.

Workflow
Default Value: %ObjectName%
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Changes Take Effect: After restart
Specifies the object name formatting rule for an object of type DN with subtype Workflow For details, see “Object Properties” at the bottom of this section.

Valid object name format string

The valid object name format string is free text string, which allows values of object properties:
%<PropertyName>:<side><padding><length>%

- PropertyName—Specifies the property name.
  Note: Property names are case sensitive.
- side—Specifies the side, L (left) or R (right) , from where the length must be counted. If you do not specify a side, Genesys Pulse uses L by default.
  L is commonly used for string or text properties.
  R is commonly used for numbers.
- padding—Specifies the padding when the property value length is less than the specified custom length
  . (dot) to pad with space characters
  0 (zero) to pad with zero characters.
- length—Specifies the maximum number of characters for the property name.

Valid Object Name Format String	Description
• %EmployeeID:10% • %EmployeeID:L10%	Both specify 10 characters of property EmployeeID from the left side.
• %EmployeeID:.10% • %EmployeeID:L.10%	Specifies 10 characters of property EmployeeID from left, but if length was less than 4 symbols, pad it with spaces.
• %DBID:R4%	Specifies 4 characters of property DBID from the right side.
• %DBID:R04%	Specifies 4 characters of property DBID from the right side, but if the length is less than 4 characters, pads it with zeros.

Object Properties

The table below lists the object properties available for use in the format strings.

Object Type	Property	Description
All Object Types	CustomName	You can specify an object name to use only within Genesys Pulse that is different than the actual object name in the Configuration Server. When this is configured you can see the custom object name in Genesys Pulse widgets, although you still see the original object name when choosing objects for your widget in Genesys Pulse. This is the useful for Virtual Queues when their Display Name is used in strategies, but not recommended for Genesys Pulse. To configure a custom Genesys Pulse object name for Configuration Server object: 1. Set the Object Annex option named `display_name`, placed in the `[pulse]` section for one or many objects. 2. Configure Genesys Pulse Collector to use custom Genesys Pulse Object Name by configuring object name format with format string `%CustomName%`. 3. The Object Name format is

Object Type	Property	Description
		configured in Genesys Pulse Collector options section [object-name-format]. If the custom name is not defined, the `%ObjectName%` value is used. **Special case:** For the Campaign Calling List (assignment of a Calling List to a Campaign rather than a native Configuration Server object), the custom name is a combination of the custom names of the corresponding Calling List and Campaign name. If some part of this name is not defined, the `%ObjectName%` value is used.
All Object Types	DBID	Specifies the ID of the object. For example, Campaign Group is in group DBID, Campaign Calling List is in CallingList ID.

All Object Types	ObjectID	Specifies the ID number of the object in Genesys Pulse Collector.
		
		Notes:
		
		• This is the typically the configuration layer DBID, but for some types of objects (for example, Campaign Group, Campaign Calling List) this is a composite 64-bit ID.
		• The composite 64-bit ID is an unsigned 64-bit number with the following composition:
		  • higher 32 bits: DBID of Campaign
		  • lower 32 bits: DBID of Calling List or Agent/Place group

All Object Types	ObjectName	Specifies the name of the object, which is written to the snapshot file.
All Object Types	ObjectType	Specifies the type of the object.
All Object Types	TenantID	Specifies the Tenant ID of the object.
All Object Types	type	Specifies the type of the object.
Agent	EmailAddress	Specifies the email address of the agent (person).
Agent	EmployeeID	Specifies the employee ID of the agent (person).

Object Type	Property	Description
Agent	ExternalID	Specifies the external ID of the agent (person).
Agent	FirstName	Specifies the first name of the agent (person).
Agent	LastName	Specifies the last name of the agent (person).
Agent	UserName	Specifies the user name of the agent (person).
Calling List	Description	Describes the calling list.
Campaign	Description	Describes the campaign.
Campaign Calling List	CallingListDescription	Describes the underlying Calling List object.
Campaign Calling List	CallingListName	Specifies the DBID of the Calling List object.
Campaign Calling List	CampaignDBID	Specifies the DBID of the Campaign object.
Campaign Group	CampaignDBID	Specifies the DBID of the Campaign object.
Campaign Group	GroupDBID	Specifies the DBID of the group object.
Campaign Group	GroupType	Specifies the numeric Type of the group object (CFGAgentGroup or CFGPlaceGroup).
DN, Routing Queue, Routing Point	Alias	Specifies the DN Alias.
DN, Routing Queue, Routing Point	AliasOrNumber	Populated with the DN Alias if available, otherwise populated with the DN number.
DN, Routing Queue, Routing Point	Number	Specifies the DN number.
DN, Routing Queue, Routing Point	SwitchDBID	Specifies the switch DBID.
DN, Routing Queue, Routing Point	SwitchID	Specifies the switch name.
Routing Strategy, Staging Area, Workbin	ScriptType	Specifies the script type ID.
Switch	DNRange	Specifies the switch DN Range.
Switch	SwitchType	Specifies the switch type.

[output] Section

collector-snapshot-log-level
Default Value: Warning
Valid Values: Debug, Info, Warning, Error, Fatal, Unknown, None (case-insensitive)
Changes Take Effect: After restart
Determines minimum log level for snapshot message that is written to Genesys Pulse Collector log.

max-output-interval

Default Value: 3600
Valid Values: 3–3600
Changes Take Effect: After restart
Specifies the maximum allowed output interval for all report layouts. Users can independently set output frequencies by layout within the Genesys Pulse user interface. If the set frequency, however, is greater than the value of this option, Genesys Pulse uses the value of this option instead.
Note: The value of this option must be greater than the value of the min-output-interval option or Genesys Pulse Collector logs an appropriate error.

min-output-interval
Default Value: 3
Valid Values: 3–3600
Changes Take Effect: After restart
Specifies the minimum allowed output interval for all report layouts. Users can independently set output frequencies by layout within the Genesys Pulse user interface. If the set frequency, however, is less than the value of this option, Genesys Pulse uses the value of this option instead.

snapshot-log-level
Default Value: Info
Valid Values: Debug, Info, Warning, Error, Fatal, Unknown, None (case-insensitive)
Changes Take Effect: After restart
Specifies the minimum log level for a snapshot message that is put into a layout snapshot.

[parallel-processing] Section

inactive-stat-data-processing-thread-gc-threshold
Default Value: 1800
Valid Values: 1-86400
Changes Take Effect: After restart
Defines, how much time, in seconds, Genesys Pulse Collector keeps previously allocated statistical data processing thread, which is currently inactive, before it is collected as garbage object.
Note: This parameter defines the minimum timeout. Actual garbage collection happens the next time Genesys Pulse Collector attempts to activate the layout or after a full configuration recheck cycle for a changed layout.

snapshot-builder-worker-thread-count
Default Value: 2
Valid Values: 1-128
Changes Take Effect: After restart
Defines the number of concurrent threads used to build snapshots.

stat-data-processing-thread-max-load-factor
Default value: 300
Valid Values: 1-300
Changes Take Effect: After restart
Specifies that a dynamic stat data processing pool is used and that the number of threads in that pool have no more layouts than specified in this option for each processing thread.
For example, if you have 1000 widget layouts and this option set to 100 then 10 stat data processing threads are created.

stat-data-processing-thread-pool-size
Default Value: 4
Valid Values: 0...256
Changes Take Effect: After restart
Specifies the number of threads in stat data processing thread pool. If this number is non-zero, then a fixed-size pool with the specified number of threads is used: otherwise; a dynamic-size pool is used, which is controlled by option `stat-data-processing-thread-max-load-factor`.
Setting this option to `0` or any value >=2 only takes effect when the `use-multiple-stat-data-processing-threads` option is set to `yes`.

use-multiple-stat-data-processing-threads
Default Value: yes
Valid values: yes, no
Changes Take Effect: After restart
Enables multi-threaded statistic data processing.

[scripting] Section

definition-script-execution-timeout
Default Value: 45
Valid Values: 1-900
Changes Take Effect: After restart
Time in seconds allowed for definition script execution.

formula-script-execution-timeout
Default Value: 5
Valid Values: 1-900
Changes Take Effect: After restart
Time in seconds allowed for single formula evaluation script execution.

init-script-execution-timeout
Default Value: 60
Valid Values: 1-900
Changes Take Effect: After restart
Time in seconds allowed for initialization script execution.

js-lib-path
Default Value: ./jslib/standard
Valid Values: Valid folder paths
Changes Take Effect: After restart
Comma-separated list of locations of the directories that contain additional JavaScript libraries to be used within the formula scripting engine.
Note: Relative file paths must begin with `./`

js-modules
Default Value: collector.js,cfglib.js,statlib.js,gts.js

Valid Values: Comma-separated list of JavaScript files
Changes Take Effect: After restart
Comma-separated list of modules to preload into the scripting engine.

stop-compute-formula-threshold-for-snapshot
Default Value: 3
Valid Values: 0-100
Change Take Effect: After restart
Specifies the maximum allowed number of `timeout expired` errors during formula computation, after which, Genesys Pulse Collector assigns the particular formula-based statistic the ERR0R value for the current snapshot. A zero value of this option suppresses the limit of the `timeout expired` failures.

[statistic-request-handling] Section

always-use-statserver-newapi
Default Value: yes
Valid Values: yes, no
Changes Take Effect: After restart
Determines whether to force Genesys Pulse Collector to request all statistics through the Stat Server New API that uses the proper parameter set.

default-insensitivity-percentage
Default Value: 0
Valid Values: 0-10000
Change takes effect: After restart
The amount of relative change (in percentage) in the value of floating-point statistics required to generate a data change notification. The insensitivity value is added to all floating-point statistic requests. The default value of 0 (zero) means no insensitivity value will be added.

default-insensitivity-absolute-value
Default Value: 0
Valid Values: 0-1000000
Change takes effect: After restart
The amount of absolute change in the value of long integer statistics required to generate a data change notification. The insensitivity value is added to all long integer statistic requests. The default value of 0 (zero) means no insensitivity value will be added.

data-source-choice-strategy
Default Value: PrimaryRunMode
Valid Values: PrimaryRunMode, LastGood, MostUpToDate, PrimaryInCME, Local
Note: All values are case-insensitive.
Changes Take Effect: After restart
Specifies which Stat Server configured in the configuration layer that Genesys Pulse Collector uses as a data source for a snapshot:

- PrimaryRunMode—Genesys Pulse Collector uses the Stat Server running in the primary mode. If both Stat Servers appear to be backup, Genesys Pulse Collector attempts the next strategy.
- LastGood—Genesys Pulse Collector uses the last good Stat Server if available. Otherwise, Genesys Pulse

Collector attempts the next strategy.

- MostUpToDate—Genesys Pulse Collector uses Stat Server that sent statistic data with most recent Server Time. Otherwise, Genesys Pulse Collector attempts the next strategy.
- PrimaryInCME—Genesys Pulse Collector uses the Primary Stat Server if available. If the Primary Stat Server is unavailable, Genesys Pulse Collector uses the Backup Stat Server. Otherwise, Genesys Pulse Collector attempts the next strategy.
- Local—Genesys Pulse Collector uses Stat Server installed on the same host. If both Primary and Backup Stat Servers are local or none of them is installed on the same host, then the PrimaryRunMode strategy is applied.

max-stat-data-queue-size
Default Value: 2147483647
Valid Values: 1-2147483647
Changes Take Effect: After restart
Specifies the limit for the internal Genesys Pulse Collector statistical data queue, which stores unprocessed data from Stat Server. Genesys Pulse drops incoming data exceeds this limit. Genesys recommends you set the value of this option to an at least **six times** the expected maximum total number of statistics.

max-stat-reopen-count
Default Value: 5
Valid Values: 0-1000
Changes Take Effect: After restart
Introduced: Genesys Pulse Collector 9.0.006.04
Specifies the number of attempts to reopen statistics after generally unrecoverable error. A timeout between reopen attempts is controlled by the **open-stat-retry-timeout** option.

min-notification-interval
Takes effect: After restart
Valid values: 1...86400
Default value: 10
Description: Minimum allowed notification interval for the time-based statistics. If a statistic is configured with notification interval that is lower than the one configured here, Pulse Collector uses the value from this option.

open-stat-retry-timeout
Default Value: 30
Valid Values: 1-3600
Changes Take Effect: After restart
Specifies the timeout in seconds, that Genesys Pulse Collector waits before attempting to re-open a failed statistic if Stat Server indicates that the error is recoverable.

optimize-statistic-requests
Default Value: yes
Valid Values: yes, no
Changes Take Effect: After restart
Specifies whether to enable statistic request optimization.

statserver-batch-size
Default Value: 500
Valid Values: 1-10000
Changes Take Effect: Immediately
Specifies the number of statistic requests sent to Stat Server in a single packet. The recommended value depends on the number of statistic requests you plan to run, network bandwidth, and processing capabilities of the Stat Server and Genesys Pulse Collector servers. If Stat Server disconnects Genesys Pulse Collector when it is opening statistics with error message Client too slow, decrease value of this option.

statserver-profiles-timeout
Default Value: 600
Valid Values: 1-86400
Changes Take Effect: Immediately
Specifies the timeout, in seconds, to receive and process server profiles from Stat Server. If profiles are not received and processed within the given timeout, Genesys Pulse Collector closes the current connection to Stat Server and attempts to reconnect.

suspend-notifications-from-secondary-server
Default value: yes
Valid values: yes, no
Change takes effect: After restart
Specifies whether Genesys Pulse Collector should attempt to use Stat Server's capability to quickly suspend and resume notifications for all request, when data from a particular Stat Server is not used at the moment to generate snapshots. To use this option, you must have Stat Server release 8.5.102 or later.

suspend-statistic-notifications-for-paused-layouts
Default Value: no
Valid Values: yes, no
Changes Take Effect: After restart
Determines whether to suspend the statistic notifications for paused layouts.
Note: You must have Stat Server version 8.1.200.17 or higher for this functionality to work correctly, if you set the value of suspend-statistic-notifications-for-paused-layouts to yes.

verbose-request-statistics
Default Value: false
Valid Values: true, false
Changes Take Effect: Immediately
Determines whether to enable Genesys Pulse Collector to log verbose messages about the objects for which it requests statistics.

[transport-aeron] Section

disabled
Default Value: no
Valid Values: yes, no
Changes Take Effect: After restart

Controls whether the Aeron transport is enabled or disabled.

> **Important**
> Prior to release 9.0.001, Genesys Pulse and Genesys Pulse Collector do not support Aeron transport configured on Windows.

driver-directory
Default Value: No default value
Valid Values: Any valid folder path
Changes Take Effect: After restart
A path to the Aeron Media Driver directory.

endpoints
Default Value: localhost: 40123
Valid Values: A comma-separated list of host and port pairs.
Changes Take Effect: After restart
A comma-separated list of host and port pairs. Host can be a multicast group IP address, a network host IP or a host name. Note, that if the host name is used then an attempt to resolve it to an IP address should yield the identical results on both the target host and the host where Genesys Pulse Collector is installed. Do not use localhost in the multihost installation.

reconnect-interval
Default Value: 10
Valid Values: 0-3600
Changes Take Effect: After restart
A timeout, in seconds, to wait before attempting a reconnect to Aeron Media Driver.

stream-id
Default Value: 10
Valid Values: 1-2147483647
Changes Take Effect: After restart
Specifies the id of an Aeron stream to publish snapshots to.

[transport-file] Section

> **Important**
> See the important note in the [heartbeat] section to see how Genesys Pulse uses the folder options.

compression-method
Default Value: None
Valid Values: None, LZ4 (case-insensitive)
Changes Take Effect: After restart
Specifies whether compression should be used and which type of compression to use for the snapshot files.

enable-latest-snapshot-output
Default Value: no
Valid Values: yes, no
Changes Take Effect: After restart
Specifies whether separate output of the latest full snapshot is enabled.

external-latest-snapshot-output-folder
Default Value: No default value
Valid Values: Network or locally mounted path to the latest-snapshot-output-folder that is accessible from a remote Genesys Pulse instance.
Changes Take Effect: After restart
Enables a remote Genesys Pulse instance to access the latest Genesys Pulse Collector output files. This option must be used together with the latest-snapshot-output-folder. The value of this option is ignored by Genesys Pulse in cases when both Genesys Pulse Collector and Genesys Pulse are configured on the same host.
Deprecated since release 9.0.000, configure Genesys Pulse with WebDAV instead.

external-latest-snapshot-output-url
Default Value: No default value
Valid Values: Valid HTTP URL to a latest-snapshot-output-folder through WebDAV server. For example, http://host1:8080/latest_snapshot
Changes Take Effect: After restart
Enables a remote Genesys Pulse instance to access latest snapshot through WebDAV server. This option must be used together with the latest-snapshot-output-folder. The value of this option is ignored by Genesys Pulse in cases when both Genesys Pulse Collector and Genesys Pulse are configured on the same host or the external-output-folder option is specified.

external-output-folder
Default Value: No default value
Valid Values: Network or locally mounted path to the heartbeat-folder accessible from a remote Genesys Pulse instance
Changes Take Effect: After restart
Enables a remote Genesys Pulse instance to access the Genesys Pulse Collector output files. This option must be used together with the output-folder.
Note: Relative file paths must begin with `./`
Deprecated since release 9.0.000, configure Genesys Pulse with WebDAV instead.

external-output-url
Default Value: No default value
Valid Values: valid HTTP URL to a snapshot's folder on WebDAV server. For example, http://host1:8080/output
Changes Take Effect: After restart

Enables a remote Genesys Pulse instance to access snapshots through WebDAV server. This option must be used together with the output-folder. The value of this option is ignored by Genesys Pulse in case when Genesys Pulse Collector and Genesys Pulse are configured on the same Host or external-output-folder option is specified.

latest-snapshot-output-folder
Default Value: ./output/latest
Valid Values: Valid file path
Changes Take Effect: After restart
Specifies the path to store the output of the latest full snapshot.
Note: Relative file paths must begin with `./`

lz4-compression-level
Default Value: 1
Valid Values: 1-16
Changes Take Effect: After restart
Specifies the compression level for the LZ4 compression method.

output-file-ext
Default Value: gpb
Valid Values: Valid file extensions for your operating system
Changes Take Effect: After restart
Specifies the file extension for the full output file format.

output-file-mode
Default Value: 0664
Valid Values: 0-0777
Changes Take Effect: After restart
On Linux, specifies the UNIX mode for each output file created by this transport.
Important! This option respects umask set at OS level.

output-folder
Default Value: No default (the Collector.apd file supplies the value of a sample output directory)
Valid Values: Valid folder paths
Changes Take Effect: After restart
Specifies the path in which Genesys Pulse Collector writes output files. If you specify a folder that does not exist, Genesys Pulse Collector creates it for you.
Note: Relative file paths must begin with `./`

worker-thread-count
Default Value: 2
Valid Values: 1-128
Changes Take Effect: After restart
Defines the number of concurrent threads used to write snapshots.

[transport-rabbitmq] Section

disabled
Default Value: no
Valid Values: yes, no
Changes Take Effect: After restart
Disables RabbitMQ messaging.

exchange
Default Value: <None>
Valid Values: A non-empty sequence of these characters: letters, digits, hyphen, underscore, dot, or colon.
Changes Take Effect: After Genesys Pulse Collector restart
Specifies the full name of a RabbitMQ exchange where Genesys Pulse Collector writes delta messages. The value of exchange option should be same for the primary and backup Genesys Pulse Collectors if you want to enable HA for quick updates. This, however, might significantly increase traffic between RabbitMQ nodes. If you use different values for this option for Primary and Backup Genesys Pulse Collectors then the network traffic will be reduced significantly but Quick updates might not work if one of the Genesys Pulse Collectors is down

max-queue-length
Default Value: 1000
Valid Values: 1-10000
Changes Take Effect: After restart
Specifies the maximum number of messages allowed in the queue.

nodes
Default Value: localhost:5672
Valid Values: hostname1:port1, hostnameN:portN
You must specify the port value: 0-65535.
Changes Take Effect: After restart
Lists a single host name with a port for a single node configuration, and all cluster nodes host names and ports for the cluster configuration. **Note:** If Genesys Pulse Backend and Genesys Pulse Collector it connects to run on one of the hosts in the list, put this host as the first entry in the list.

password
Default Value: pulse
Valid Values: String
Changes Take Effect: After restart
Specifies the password for the username.

reconnect-interval
Default Value: 10
Valid Values: 0-3600
Changes Take Effect: After restart
Specifies the time interval, in seconds, between reconnection attempts.

username

Default Value: pulse
Valid Values: String
Changes Take Effect: After restart
Specifies the username.

vhost
Default Value: /pulse
Valid Values: String
Changes Take Effect: After restart
Specifies the path to the virtual host.

[transport-webdav] Section

output-url
Default Value: No default value
Valid Values: Valid URL
Changes Take Effect: After Genesys Pulse restart
WebDAV URL to access snapshots generated by Genesys Pulse Collector. If this option is specified, values of the **[transport-file]/external-output-url** and **[transport-file]/external-output-folder** options are ignored.

heartbeat-url
Default Value: No default value
Valid Values: Valid URL
Changes Take Effect: After Genesys Pulse restart
WebDAV URL to access the heartbeat file generated by Genesys Pulse Collector. If this option is specified, values of the **[heartbeat]/external-heartbeat-url** and **[heartbeat]/external-heartbeat-folder** options are ignored.

latest-snapshot-output-url
Default Value: No default value
Valid Values: Valid URL
Changes Take Effect: After Genesys Pulse restart
WebDAV URL to access the latest snapshots generated by Genesys Pulse Collector. If this option is specified, values of the **[transport-file]/external-latest-snapshot-output-url** and **[transport-file]/external-latest-snapshot-output-folder** options are ignored.

username
Default Value: No default value
Valid Values: Any string
Changes Take Effect: After Genesys Pulse restart
The username for authentication on WebDAV Server.

password
Default Value: No default value
Valid Values: Any string
Changes Take Effect: After Genesys Pulse restart

The password for authentication on WebDAV Server.

connection-timeout
Default Value: 5000
Valid Values: 0 - 2147483647
Changes Take Effect: After Genesys Pulse restart
Time, in milliseconds, to wait until the connection to WebDAV Server is established.

[embedded-dbserver] section

msql_name
Valid values: Path to Genesys DB client executable for Microsoft SQL Server.
Default value: empty
Takes effect: After restart
Description: Specified path to Genesys DB client executable for Microsoft SQL Server. Empty value or missing option will result in using platform-specific default, which targets to platform-specific DB client executable located under path `./dbclients`

oracle_name
Valid values: Path to Genesys DB client executable for Oracle.
Default value: empty
Takes effect: After restart
Description: Specified path to Genesys DB client executable for Oracle. Empty value or missing option will result in using platform-specific default, which targets to platform-specific DB client executable located under path `./dbclients` .

postgre_name
Valid values: Path to Genesys DB client executable for PostgreSQL.
Default value: empty
Takes effect: After restart
Description: Specified path to Genesys DB client executable for PostgreSQL. Empty value or missing option will result in using platform-specific default, which targets to platform-specific DB client executable located under path `./dbclients` .

[log-db] section

Options in this section control embedded DB Server log and take effect if **[collector]/dbthread** option is set to **yes**. For more information, see Configure Genesys Pulse Collector with an Embedded DB Server. Log options are inherited from the Common Log Options section of the *Framework Configuration Options Reference Manual*.

all
Default Value: No default value
Valid Values:

- stdout—Log events are sent to the Standard output (stdout).
- stderr—Log events are sent to the Standard error output (stderr).
- network—Log events are sent to Message Server, which can reside anywhere on the network. Message

Server stores log events in the Log Database.

Setting the all log-level option to network enables Data Sourcer to send log events of Standard, Interaction, and Trace levels to Message Server. Log events of Debug level are neither sent to Message Server nor stored in the Log Database.

- memory—Log events are sent to the memory output on the local disk. This output is the safest in terms of the application performance.
- [filename]—Log events are stored in a file with the specified name. If you do not specify a path, the file is created in the application's working directory.

Changes Take Effect: after restart
Specifies the outputs to which an application sends all log events. You must separate log-output types with commas when you configure more than one output type.
For example, all = stdout, logfile
Notes:

- To ease the troubleshooting process, consider using unique names for log files that different applications generate.
- Relative file paths must begin with `./`

buffering
Default Value: false
Valid Values: true, false
Changes Take Effect: after restart
Specifies whether the operating system file buffering is on or off. This option applies only to stderr and stdout output. Setting this option to true increases output performance.
Note: When you enable buffering, log messages might appear in the log after a delay.

expire
Default Value: false
Valid Values:

- false—No expiration; all generated segments are stored.
- <number>[file]—Sets the maximum number of log files to store. Specify a number from 1–1000.
- <number> day—Sets the maximum number of days before log files are deleted. Specify a number from 1–100.

Changes Take Effect: after restart
Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number of files (segments) or days before the files are removed. This option is ignored if log output is not configured to be sent to a log file.

segment
Default Value: 10 MB
Valid Values:

- false—No segmentation allowed.
- <number> KB or <number>—Sets the maximum segment size in kilobytes. The minimum segment size is 100 KB.

- <number> MB—Sets the maximum segment size, in megabytes.
- <number> hr—Sets the number of hours for which the segment stays open. The minimum number is 1 hour.

Changes Take Effect: after restart
Specifies if there is a segmentation limit for a log file. If there is, this option sets the unit of measurement along with the maximum size. If the current log segment exceeds the size set by this option, the current file is closed and a new file is created.

verbose
Default Value: all
Valid Values:

- all—All log events (that is, log events of Standard, Trace, Interaction, and Debug levels) are generated if you set the debug-level option in the statserver section to all.
- debug—The same as all.
- trace—Log events of the Trace and higher levels (that is, log events of Standard, Interaction, and Trace levels) are generated, while log events of the Debug level are not generated.
- interaction—Log events of the Interaction and higher levels (that is, log events of Standard and Interaction levels) are generated, while log events of the Trace and Debug levels are not generated.
- standard—Log events of the Standard level are generated, while log events of the Interaction, Trace, and Debug levels are not generated.
- none—Produces no output.

Changes Take Effect: after restart
Determines whether a log output is created. If it is, this option specifies the minimum level of log events that are generated. The log-event levels, starting with the highest-priority level, are Standard, Interaction, Trace, and Debug.
Refer to the *Framework Management Layer User's Guide* for more information on the Standard, Trace, Interaction, and Debug log levels.

Access Groups Object

To optimize resources utilization by Genesys Pulse, the following options can be configured on the **Annex** tab of Access Groups in the **[pulse]** section.

max_tabs_per_user
Default Value: No default value
Valid Values: zero or any positive number
Changes Take Effect: After user re-login
Limits the number of dashboards for each member of a group, 0 means no limit.

The SYSTEM account (or account under which Genesys Pulse runs) must have the Read access to the appropriate Access Group with the specified option.

If a person, who needs to be limited by the number of dashboards, is the member of two or more

Access Groups with the `max_tabs_per_user` option specified, the effective limit is the biggest (less restrictive) of all limits. If the option is not specified or set to 0 (no limit) on the Access Group level, the application level `max_tabs_per_user` option is applied.

> **Tip**
> You can use the `Pulse Add Dashboards Without Limit` privilege to set no limits for particular users.

max_widgets_per_user
Default Value: No default value
Valid Values: zero or any positive number
Changes Take Effect: After user re-login
Limits the number of widgets for each member of a group, 0 means no limit.

The SYSTEM account (or account under which Genesys Pulse runs) must have the Read access to the appropriate Access Group with the specified option.

If a person, who needs to be limited by the number of widgets, is the member of two or more Access Groups with the `max_widgets_per_user` option specified, the effective limit is the biggest (less restrictive) of all limits. If the option is not specified or set to 0 (no limit) on the Access Group level, the application level `max_widgets_per_user` option is applied.

> **Tip**
> You can use the `Pulse Add Widgets Without Limit` privilege to set no limits for particular users.

Data Access Point Application Object

To specify the database access, configure the **[Pulse]** section on the `Application Options` tab of your Genesys Pulse DAP with the following option:

jdbc_url
Default Value: No default value
Valid Values: string jdbc url
Changes Take Effect: After restart
Genesys Pulse connects to the database using customized JDBC URL specified in DAP. If this option is not configured, Genesys Pulse uses the DAP `Application` object values for establishing the connection.

A sample URL for the MSSQL database:
jdbc:sqlserver://<host>:<port, usually 1433>;Database=<database name>;username=<username>;password=<password>;authentication=SqlPassword

A sample URL for the PostgreSQL database:
jdbc:postgresql://<host>:<port, usually 5432>/<database name>;username=<username>;password=<password>;ssl=true

A sample URL for the Oracle database:
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=<host>)(PORT=<port, usually 1521>))(CONNECT_DATA=(SERVICE_NAME=<database name>)(FAILOVER_MODE=(TYPE=SELECT)(METHOD=BASIC)(RETRIES=10)(DELAY=5))))

> **Tip**
> The username and password, in the URL, are optional for security reasons, and may not be specified. If the username and password are specified, Genesys Pulse considers them to be the access credentials to the Genesys Pulse Database, and makes the connection. If not specified, Genesys Pulse takes the username and password from the `General` tab of the DAP `Application` object.

> **Important**
> Each DBMS configures URL in different ways. You must provide the URL in the correct format and syntax as required for your DBMS. Genesys Pulse cannot establish connection to the Database if the URL is incorrect or improperly formed.

connect_timeout
Default Value: 30
Valid Values: 0 (zero) or any positive number
Changes Take Effect: After restart
Introduced: Genesys Pulse 9.0.006.03 for PostgreSQL; Genesys Pulse 9.0.008.02 for all supported databases
The amount of time, in seconds, to wait for a socket connection. A 0 value means no timeout. This option is ignored for PostgreSQL and MS SQL Server when jdbc_url is specified, and the value is taken from custom jdbc_url.

login_timeout
Default Value: 30
Valid Values: any positive number
Changes Take Effect: After restart
Introduced: Genesys Pulse 9.0.006.03 for PostgreSQL; Genesys Pulse 9.0.008.02 for all supported databases
The amount of time, in seconds, to wait for a database connection once the driver has been identified. This option is ignored for PostgreSQL and MS SQL Server when jdbc_url is specified, and

the value is taken from custom jdbc_url.

socket_timeout
Default Value: 30
Valid Values: 0 (zero) or any positive number
Changes Take Effect: After restart
Introduced: Genesys Pulse 9.0.006.03 for PostgreSQL; Genesys Pulse 9.0.008.02 for all supported databases
The amount of time, in seconds, to wait for a network socket operations. A 0 value means no timeout. This option is ignored for PostgreSQL and MS SQL Server when jdbc_url is specified, and the value is taken from custom jdbc_url.

> **Important**
>
> Long-running operations with the database (such as importing a large number of Dashboards, Wallboards or Widget Templates) can fail with a `Could not connect to the database` error in UI and the exception message `The connection is closed.` in the Pulse log when the configured values used for connect_timeout, login_timeout and socket_timeout options for Pulse DAP are too small. Workaround: Increase the connect_timeout, login_timeout and socket_timeout option values for Pulse DAP.

tls_mode
Default Value: off
Valid Values: off, request, require, authenticate
Changes Take Effect: After restart
Introduced: Genesys Pulse 9.0.008.02
The option allows to configure secured connection between Pulse and DBMS. Choose appropriate mode according to your DBMS:

Option value / DBMS	PostgresSQL JDBC URL and Connection property	MS SQL Server JDBC URL	ORACLE JDBC URL and Connection property
`off`, or no option entered	ssl=false Connection property `sslmode` is set to `disable`	authentication=NotSpecified	protocol=TCP
`request`	ssl=true	authentication=SqlPassword	protocol=TCP
`require`	ssl=true	authentication=SqlPassword	protocol=TCPS Connection property `oracle.net.ssl_cipher_suites` is set to certain cipher suites
`authenticate`	ssl=true	authentication=SqlPassword trustServerCertificate=false	protocol=TCPS

pulse.properties File

The following options are configured in the `pulse.properties` file under the `conf` folder inside the Genesys Pulse installation folder.

accesslog_enabled
Default Value: `true`
Valid Values: `true`, `false`
Changes Take Effect: After restart
Enables HTTP access logging.

accesslog_filename
Default Value: ./logs/http-yyyy_mm_dd.log
Valid Values: valid filename
Changes Take Effect: After restart
File name for the HTTP access log.

accesslog_timezone
Default Value: GMT
Valid Values: valid timezone
Changes Take Effect: After restart
Time zone for the HTTP access log.
Note: It is recommended to specify the time zone value in canonical format instead of three-letter time zone IDs. For example, Australia/Sydney, Asia/Kolkata, America/Los_Angeles.

accesslog_append
Default Value: `true`
Valid Values: `true`, `false`
Changes Take Effect: After restart
After Genesys Pulse is restarted, specifies whether to append to the existing HTTP access log.

accesslog_extended
Default Value: `false`
Valid Values: `true`, `false`
Changes Take Effect: After restart
Specifies whether to include the extended information in the HTTP access log.

accesslog_cookies
Default Value: `false`
Valid Values: `true`, `false`
Changes Take Effect: After restart
Specifies whether to include cookies in the HTTP access log.

accesslog_retaindays
Default Value: 90

Valid Values: integer
Changes Take Effect: After restart
Specifies the number of days to retain the HTTP access log.

app
Default Value: No default value
Valid Values: Application object name
Changes Take Effect: After restart
Specifies the Genesys Pulse Application object.
This option is mandatory.

backup_port
Default Value: No default value
Valid Values: integer
Changes Take Effect: After restart
Specifies the backup Configuration Server port.

backup_host
Default Value: No default value
Valid Values: Valid FQDN or IP address
Changes Take Effect: After restart
Specifies the backup Configuration Server host, written as a Fully Qualified Domain Name (FQDN) or an IP address.

content_security_policy
Default Value: No default value
Valid Values: delimited list of directives (see the full list of valid directives at Mozilla Developer Network)
Changes Take Effect: After restart
Allows to control the Content Security Policy header for an additional protection against XSS attacks.
Example: default-src 'none'; script-src * 'unsafe-inline' 'unsafe-eval'; style-src * 'unsafe-inline'; img-src * data:; connect-src 'self'; font-src *; frame-src *; form-action 'self'; worker-src 'self'; base-uri 'self'; block-all-mixed-content; frame-ancestors 'none';

> ### Important
> This option should be used with caution as wrong values could prevent Genesys Pulse UI from loading. Carefully review the following notes before usage:
>
> - The value of the `script-src` directive must include `'unsafe-inline' 'unsafe-eval'`
> - The value of the `style-src` directive must include `'unsafe-inline'`
> - In cases using external resources or extensions, the following directives should contain * (asterisk) or exact hostname values (more secure):
>   - `script-src 'unsafe-inline' 'unsafe-eval' *` or `script-src self 'unsafe-inline' 'unsafe-eval' <space separated list of sources>` for external

scripts

- `style-src 'unsafe-inline' *` or `style-src self 'unsafe-inline'` *<space separated list of sources>* for external styles
- `font-src *` or `font-src self` *<space separated list of sources>* for external fonts
- `img-src * data:` or `font-src self data:` *<space separated list of sources>* for external images
- `frame-src *` or `frame-src self` *<space separated list of sources>* for embedding 3rd party sites in an iframe widget
- if no external resources or extensions are used then * (asterisk) should be replaced with 'self'

- If embedding Pulse into an external iframe, the `frame-ancestors` directive should contain valid parents that may embed: `frame-ancestors ‘self' https://example.com;` or contain `'none'` if prohibited to embed Pulse into external iframe
- If not specified, the value of the `sandbox` directive will be set to default `'allow-scripts' 'allow-same-origin' 'allow-popups' 'allow-forms'` value. Restrictions that are weaker than the default value will have no effect.

disable_xframe_options
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
Disables X-Frame-Options security header. See Configuring System Security for more information.

enable_hsts
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
Enables or disables HSTS. See Configuring System Security for more information.

host
Default Value: No default value
Valid Values: Valid FQDN or IP address
Changes Take Effect: After restart
Specifies the primary Configuration Server host, written as a Fully Qualified Domain Name (FQDN) or an IP address.
This option is mandatory.

http_port
Default Value: 8080
Valid Values: integer
Changes Take Effect: After restart

Defines the HTTP port.

https_port
Default Value: No default Value
Valid Values: integer
Changes Take Effect: After restart
Defines the HTTPS port. See Configuring System Security for more information.

keystore_password
Default Value: No default Value
Valid Values: string
Changes Take Effect: After restart
Specifies the keystore password.

keystore_path
Default Value: No default Value
Valid Values: Full path to the location of the keystore
Changes Take Effect: After restart
Defines the keystore path. See Configuring System Security for more information.

max_cfg_connection
Default Value: 200
Valid Values: -1 or any positive integer
Changes Take Effect: After restart
Specifies the maximum number of allowed connections from Genesys Pulse to Configuration Server. To allow unlimited connections, set the value to -1.

max_idle_time
Default Value: 1000*60*60
Valid Values: integer
Changes Take Effect: After restart
Specifies the maximum idle time, in milliseconds, for HTTP connection

port
Default Value: No default value
Valid Values: integer
Changes Take Effect: After restart
Specifies the primary Configuration Server port.
This option is mandatory.

protocol_timeout
Default Value: 30000
Valid Values: Any positive integer
Changes Take Effect: After restart
Introduced: 9.0.006.03
Specifies timeout, in milliseconds, for a connection with Configuration Server.

root_url
Default Value: /pulse
Valid Values: root URL
Changes Take Effect: After restart
Specifies the root URL (host:port/rootURL).

saml
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
Enables SSO login to Pulse using SAML

saml_entityid
Default Value: No default value
Valid Values: string
Changes Take Effect: After restart
Specifies entity ID used by Pulse metadata

saml_external_userid
Default Value: No default value
Valid Values: string
Changes Take Effect: After restart
Defines the protocol to correlate to the External User ID in Genesys Person object.
Pulse must extract DSID from SAML assertion and query the user by using the external ID, then use the username to enable the user to log in.

saml_idp_metadata
Default Value: No default value
Valid Values: Valid path (or) URL
Changes Take Effect: After restart
Specifies the path or URL of the IDP's metadata XML file

saml_keystore
Default Value: No default value
Valid Values: Valid path (or) URL
Changes Take Effect: After restart
Specifies the location or path to Keystore.

saml_keystore_password
Default Value: No default value
Valid Values: string
Changes Take Effect: After restart
Specifies the Keystore's password.

saml_landing_page
Default Value: No default value
Valid Values: Valid URL
Changes Take Effect: After restart
URL to redirect users on log out of Pulse.

saml_key_name
Default Value: No default value
Valid Values: string
Changes Take Effect: After restart

Specifies the signing key certificate name.

saml_key_password
Default Value: No default value
Valid Values: string
Changes Take Effect: After restart
Specifies the signing key certificate password.

session_samesite
Default Value: strict
Valid Values: none, strict, lax
Changes Take Effect: After restart
Specifies the value for SameSite cookie attribute. See Configuring System Security for more information.

session_securecookies
Default Value: false
Valid Values: true, false
Changes Take Effect: After restart
Specifies whether the **secure** flag is set on session cookie.

supported_protocol
Default Value: http
Valid Values: http, https, both
Changes Take Effect: After restart
Defines supported protocol(s). See Configuring System Security for more information.

WebDAV Server Configuration

Use WebDAV server for sharing snapshots between Genesys Pulse and Genesys Pulse Collector when they are installed on different hosts.

WebDAV server requirements

Genesys Pulse requires the following set of WebDAV methods:

- HEAD—POST for snapshot and internal snapshot cleanup functionality
- MKCOL—POST for snapshot
- PUT—POST for snapshot
- GET—GET for snapshot, GET for snapshots and for heartbeat check during GET for healthcheck
- PROPFIND—GET for snapshot, GET for snapshots and internal snapshot cleanup functionality
- DELETE—internal snapshot cleanup functionality

The recommended web server with WebDAV support is lighttpd (the latest version) with WebDAV module.

Genesys Pulse configuration with WebDAV

Before configuring WebDAV-related options, make sure that you can access the snapshot and heartbeat folders using a web browser from your Genesys Pulse hosts.

- Set the **output-folder** option in the **[transport-file]** section and the **heartbeat-folder** option in the **[heartbeat]** section in the Genesys Pulse Collector application object to local folders, because the WebDAV snapshot files are stored on the host where Genesys Pulse Collector is installed.

To enable Genesys Pulse to pull snapshots from a remote Genesys Pulse Collector instead of the local host, you need to configure the following Genesys Pulse Collector application options:

- Set the **output-url** option in the **[transport-webdav]** section to **http://<WebDAV host>/<path to snapshots folder>**
- Set the **heartbeat-url** in the **[transport-webdav]** section to **http://<WebDAV host>/<path to heartbeat folder>**
- Set the **latest-snapshot-output-url** option in the **[transport-webdav]** section to **http://<WebDAV host>/<path to the latest snapshots folder>**
- Set the **username** option in the **[transport-webdav]** section to **<username for WebDAV>**
- Set the **password** in the **[transport-webdav]** section to **<password for WebDAV>**

Ensure the **path to snapshots folder** and **path to heartbeat folder** are relative to the Document Root of the WebDAV server. For example, if the Document Root is **/var/www/** and the snapshots folder is **/var/www/snapshots**, then the URL is **http://<WebDAV host>/snapshots**.

> **Important**
>
> The Genesys Pulse Collector configuration must not contain the **external-output-folder** option in the **[transport-file]** section nor the **external-heartbeat-folder** option in the **[heartbeat]** section.

Installation and Configuration of lighttpd on Linux

> **Warning**
>
> Packages, specified below, must be installed along with all their required dependencies, using standard OS tools and practices. Genesys does not provide specific instructions on how to install third-party dependencies. The below steps are just an example of an installation on RHEL.

Installation of lighttpd with WebDAV module

1. Enable EPEL by following instructions on http://fedoraproject.org/wiki/EPEL.
2. Install `lighttpd`: As **root** issue the following command:
   - `yum install lighttpd`

Configuration of lighttpd

1. If IPv6 is not supported or should not be used, then edit the file `/etc/lighttpd/lighttpd.conf` and change `server.use-ipv6` to disable:

   ```
   server.use-ipv6 = "disable"
   ```

2. To disable returning errors when the `Expect` header is in requests and to increase the number of supported parallel requests add the following lines to the `/etc/lighttpd/lighttpd.conf` file:

   ```
   server.reject-expect-100-with-417 = "disable"
   server.max-fds = 2048
   server.max-connections = 1024
   ```

3. Make sure it runs under same user that Genesys Pulse uses or that a user that has read write access to Genesys Pulse Collector directories
   For this, you might need to adjust **server.username** and **server.groupname** in `/etc/lighttpd/lighttpd.conf`
   Make sure that this user had access to all directories for lighttpd (for example, var.log_root,var.state_dir, var.home_dir) and others mentioned in `/etc/init.d/lighttpd` and in `/etc/`

lighttpd/lighttpd.conf

4. Enable WebDAV module by editing the /etc/lighttpd/modules.conf file and uncommenting the following line:

```
include "conf.d/webdav.conf"
```

5. Change the configuration of the WebDAV module by editing the file /etc/lighttpd/conf.d/webdav.conf.

Example of configuration:

```
## This configuration assumes that
  ## transport-file/output-folder = /genesys/collector_output/snapshots
  ## transport-file/external-output-url = http://<host>/snapshots
  ## heartbeat/heartbeat-folder = /genesys/collector_output/heartbeat
  ## heartbeat/external-heartbeat-url = http://<host>/heartbeat
  server.modules += ( "mod_webdav" )
  $HTTP["url"] =~ "^($|/)" {
    ## Specify full path to parent folder for snapshots and heartbeat folders
    server.document-root = "/genesys/collector_output/"
    webdav.activate = "enable"
    dir-listing.activate = "enable"
  }
```

You can also have webdav.activate = "enable" on the top level (not inside of $HTTP) if you want all files and directories in your server.document-root to be accessible through WebDAV.

6. Make sure that the webdav.sqlite-db-name parameter is commented out in webdav.conf.
7. Restart the lighttpd server:

```
/etc/init.d/lighttpd restart
```

Installation and configuration of WebDAV extension for IIS 8 on Windows Server 2012

> **Important**
> Microsoft IIS configuration is complex and requires a competent Windows IIS administrator. Genesys does not provide support for IIS configuration.

Genesys Pulse Configuration Limitations

Anonymous Authentication is the IIS authentication method that can be used for Genesys Pulse.

Limitations when using IIS Anonymous Authentication

1. Only Genesys Pulse Collector is supported as a data collector for Genesys Pulse.
2. You must use **cleanuptool** from the Genesys Pulse Collector installation to cleanup old snapshots.

> **Important**
>
> To avoid issues with the cleaning process, disable the automatic removal by setting the Genesys Pulse option `snapshot_expire_timeout` to `0` (zero) in the `[pulse]` section.

Enhance Authentication Security

You can improve the security of Anonymous Authentication by restricting access to WebDAV to explicitly specified IP addresses of hosts where Genesys Pulse is installed:

1. Install the IP and Domain Restrictions feature of IIS.
2. Using the IIS Manager, you can select the website that is configured to share Genesys Pulse snapshots in the **Sites** section of your host.
3. In the **IP Address and Domain Restrictions** section, click **Edit Feature Settings** in the **Actions** pane.
4. Set the **Access for unspecified clients** to **Deny** and click *OK*.
5. Click **Add Allow Entry** in the actions pane and specify the IP addresses of the hosts running Genesys Pulse in the **Specific IP address** field.

Installing the WebDAV extension for IIS 8

Install the WebDAV extension and enable it for your IIS instance according to the guide provided by Microsoft.

Configure the IIS website with WebDAV extension to work with Genesys Pulse

1. Open the IIS Manager.
2. In the **Sites** section of your host, select the website that shares Genesys Pulse snapshots with the Genesys Pulse components.
3. Go to the **Authentication** section and enable the **Anonymous Authentication** method. Disable all other authentication methods.
4. Go to the **MIME Types** section and add the following MIME types:
    1. Extension "`.*`" and MIME type "`application/octet-stream`"
    2. Extension "`.gpb`" and MIME type "`application/octet-stream`"
    3. Extension "`.LZ4`" and MIME type "`application/octet-stream`"
5. Go to the **WebDAV Authoring Rules** section, click the **WebDAV Settings...** link in **Actions** pane and change the value for **Allow Anonymous Property Queries** option to **True**.
6. In the **WebDAV Authoring Rules** section, add **Read** and **Source** permissions for all content and all users.
7. Edit your web site's **Basic Settings...** so that the **Physical path** property is configured to where Genesys Pulse Collector writes the snapshot and heartbeat files.

Genesys Pulse Web Service API

Display external data in Genesys Pulse (or to display Pulse data in an external system) by using the Genesys Pulse RESTful (Representational State Transfer) Web Service API.

> **Important**
>
> - The API is subject to change at any time without notice.
> - Make sure you are authenticated before executing API methods. See the POST method under `/api/session/login` for details.

/api/session/login

Methods:

[+] POST

Authenticates a user.

Request body:

```
{
 "username": "your_username",
 "password": "your_password"
}
```

Available response representations:

- 204 - Successfully authenticated.
- 401 - Authentication failed, application/json with details.

/api/session/logout

Methods:

[+] GET

Invalidates the user's session.

Available response representations:

- 204 - Successfully invalidated.
- 401 - Authentication failed.

/api/wbrt/templates

Methods:

[+] GET

Returns array of templates by specified parameters.

Example request URI(s):

- /api/wbrt/templates
- /api/wbrt/templates?uscn=1
- /api/wbrt/templates?uscn=1&type=ltGENERIC

Request query parameters

name	required	type	description
uscn	no	long	specifies uscn, for filtering templates changed after this USCN (non inclusive)
type	no	string, available values: ltGENERIC ltPCREGULAR ltPCPERFORMANCE ltDATADEPOT ltIFRAME ltOTHER	specifies type

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with array of template.
- 400 - Request is not valid, application/json with details:
  - `{ "message": "INVALID_URL" }` - incorrect request parameters specified.
- 403 - User does not have privileges.

[+] POST

Creates new template.

Request query parameters

name	required	type	description
overwrite	no	boolean	if true, then templates with the same name and saved to the same location will be removed

By default template is created with proxy_access_object.dbid = 0 and shared for everyone without creating proxy access object.

If you want to control access, then specify empty proxy_access_object.dbid for creating proxy access object:

```
{
    "definition": {
        ...
        "proxy_access_object": {
        }
    }
}
```

You can specify dbid of folder where proxy access object must be created by proxy_access_object.folder_dbid. For folders in other tenant you have to specify proxy_access_object.tenant_dbid as well.
Instead of folder_dbid and tenant_dbid you can just specify proxy_access_object.path, for example: "path":"\\Configuration\\Environment\\Scripts"

Available request representations:

- application/json with template configuration. See LayoutInfo example.

Required permission:

- PULSE_WRITE_TEMPLATE

Available response representations:

- 200 - New template was successfully created, redirect to newly created template.
- 400 - Provided template configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - template has invalid format
  - `{ "message": "INVALID_LAYOUT_DEFINITION" }` - template definition is invalid
  - `{ "message": "PROXY_ACCESS_OBJECT_REQUIRED" }` - no proxy access object specified
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - template with same name already exists in the folder. Retry with overwrite = true to remove duplicate templates.
- 403 - User does not have privileges.

/api/wbrt/templates/<guid>

Methods:

[+] GET

Return template with specified id.

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with requested template.
- 403 - User does not have privileges.

[+] PUT

Updates template with specified id.

> **Important**
> The request does not create new template if it does not exist.

Request query parameters

name	required	type	description
overwrite	no	boolean	if true, then templates with the same name and saved to the same location are removed

The method can be used to move template between folder. Execute with proxy_access_object.dbid = 0 to make it as shared for everyone, or specify target folder in proxy_access_object.folder_dbid. For moving between tenants also specify proxy_access_object.tenant_dbid.

Available request representations:

- application/json with template configuration.

Required permission:

- PULSE_WRITE_TEMPLATE

Available response representations:

- 200 - Template was updated

- 400 - Provided template configuration is not valid, application/json with details:
  - `{ "message": JSON_PARSE_ERROR }` - template has invalid format
  - `{ "message": INVALID_LAYOUT_DEFINITION }` - new template definition is invalid
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - template with same name already exists in the folder. Retry with overwrite = trueto remove duplicate templates.
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
- 403 - User does not have privileges.
- 404 - Template does not exist, application/json with details:
  - `{ "message": "TEMPLATE_NOT_FOUND" }` - template not exists

[+] DELETE

Delete template with specified id.

Required permission:

- PULSE_WRITE_TEMPLATE

Available response representations:

- 200 - Template was deleted
- 204 - Template was already deleted
- 403 - User does not have privileges.

/api/wbrt/layouts

Methods:

[+] GET

Returns array of layouts by specified parameters.

Example request URI(s):

- /api/wbrt/layouts
- /api/wbrt/layouts?uscn=1
- /api/wbrt/layouts?uscn=1&type=ltGENERIC

Request query parameters

name	required	type	description
uscn	no	long	specifies uscn, for

name	required	type	description
			filtering layouts changed after this USCN (non inclusive)
type	no	string, available values: ltGENERIC ltPCREGULAR ltPCPERFORMANCE ltDATADEPOT ltIFRAME ltOTHER	specifies layout type

Required permission:

- PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with array of layout.
- 400 - Request is not valid, application/json with details:
  - `{ "message": "INVALID_URL" }` - incorrect request parameters specified.
- 403 - user doesn't have privileges.

[+] POST

Creates new layout.

Available request representations:

- application/json with layout configuration. See LayoutInfo example.

Required permission:

- PULSE_WRITE_LAYOUT

Available response representations:

- 200 - New layout was successfully created, redirect to newly created layout.
- 400 - Provided layout configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - layout has invalid format
  - `{ "message": "INVALID_LAYOUT_DEFINITION" }` - layout definition is invalid
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
- 403 - User does not have privileges.

/api/wbrt/layouts/<guid>

Methods:

[+] GET

Return layout with specified id.

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with requested layout.
- 403 - User does not have privileges.
- 404 - Layout does not exist or current user doesn't have any widgets for this layout, but there are some widgets of other users (layout owned by some other user)

[+] PUT

Updates layout with specified id. If there are more than one widget for this layout, then the new layout is created and returned in response.

> **Important**
> The request does not create new template if it does not exist.

Available request representations:

- application/json with layout configuration. See LayoutInfo example.

Required permission:

- PULSE_WRITE_LAYOUT

Available response representations:

- 200 - Layout was updated, redirect to new newly created layout if any
- 400 - Provided layout configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - layout has invalid format
  - `{ "message": "INVALID_LAYOUT_DEFINITION" }` - new layout definition is invalid
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.

- 403 - User does not have privileges.
- 404 - Layout does not exist, application/json with details:
  - `{ "message": "LAYOUT_NOT_FOUND" }` - layout does not exist

[+] DELETE

Delete layout with specifiifed id.

Required permission:

- PULSE_WRITE_LAYOUT

Available response representations:

- 200 - Layout was deleted
- 204 - Layout was already deleted
- 400 - could not remove specified layout, application/json with details:
  - `{ "message": "FIRST_DELETE_RELATED_WIDGETS" }` - try to remove layout before related widgets.
- 403 -In case if user doesn't have privileges.

/api/wbrt/layouts/<guid>/snapshot

Methods:

[+] GET

Returns recent snapshot for specified layout.

NOTE: If user does not have "Pulse Read All Layouts" privilege then this method performs additional rows filtering based on user access restrictions for snapshots with layout_type = **ltPCREGULAR** and layout_type = **ltPCPERFORMANCE**. Rows with objects not accessible for the user are filtered out. It must work as follows: if there is column _Object$CfgType, then use pair (_Object$CfgType, _Object$ID) to check permissions, otherwise attempt the usual combination (_Object$Type, _Object$ID).

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json snapshot.
- 204 - no content, snapshot not exists
- 403 - User does not have privilegess.
- 404 - Related resource does not exist, application/json with details:

- `{ "message": "LAYOUT_NOT_FOUND" }` - layout does not exist

[+] POST

Saves the snapshot for the layout with the specified id. Uses the timestamp property from the LayoutSnapshot to distinguish a different snapshot. If there is already a saved snapshot with the same timestamp, then it is overwritten.

Available request representations:

- application/json with snapshot. See LayoutSnapshot example

Required permission:

- PULSE_WRITE_SNAPSHOT

Available response representations:

- 200 - snapshot was successfully saved
- 400 - snapshot is not valid, application/json with details:
  - `{ "message": "SNAPSHOT_PARSE_ERROR" }` - snapshot has invalid format
  - `{ "message": "INVALID_BODY_HASH" }` - state.body_hash_1 in layout does not match state.body_hash_1 in the snapshot
- 403 - User does not have privileges.
- 404 - Related resource does not exist, application/json with details:
  - `{ "message": "LAYOUT_NOT_FOUND" }` - layout does not exist

/api/wbrt/layouts/<guid>/snapshots

Methods:

[+] GET

Returns array of snapshots for specified layout and matched filter period. If start and end are not specified then returns only latest snapshot.

Example request URI(s):

- /api/wbrt/layouts/1/snapshots
- /api/wbrt/layouts/1/snapshots?start=1411720605
- /api/wbrt/layouts/1/snapshots?start=1411720605&columns=Inbound_Talk_Time,Internal_Talk_Time
- /api/wbrt/layouts/1/snapshots?start=140000000&frequency=10

Request query parameters

name	required	type	description
start	no	long	Unix epoch timestamp indicating the start of period for which snapshots should be returned. If not specified, then all snapshots generated before end time are returned.
end	no	long	Unix epoch timestamp indication end of period for which snapshots should be returned. If not specified, then all snapshots generated after the start time are returned.
frequency	no	integer	Minimum interval between two snapshots in history in seconds. Needed for reducing amount of snapshots returned by this request. Default value is 0, so all existing snapshots inside specified period are returned.
columns	no	Array of Strings	Array of Column.id that is to be included in the snapshot. If not specified, then all columns are included.

> **Important**
>
> This method performs additional row filtering based on user access restrictions for snapshots with layout_type = **ltPCREGULAR** and layout_type = **ltPCPERFORMANCE**. Rows with objects that are not accessible for a user are filtered out. It must work as follows: if there is column _Object$CfgType, then use pair (_Object$CfgType, _Object$ID) to check permissions; otherwise, attempt usual combination (_Object$Type, _Object$ID).

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with array of snapshots. See LayoutSnapshot example.

- 403 - User does not have privileges.

/api/wbrt/widgets

Methods:

[+] GET

Returns array of widget by specified parameters.

Example request URI(s):

- /api/wbrt/widgets
- /api/wbrt/widgets?uscn=1

Request query parameters

name	required	type	description
uscn	no	long	specifies uscn, for filtering widgets changed after this USCN (non inclusive)

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED

Available response representations:

- 200 - application/json with array of widgets. See Widget example.
- 403 - User does not have privileges.

[+] POST

Create new widget for specified layout.

Available request representations:

- application/json with widget configuration. See Widget example.

Request query parameters:

name	required	type	default	description	
brief	no	boolean	true	when set to `false`, the related layout is expected to be in the widget request;	

name	required	type	default	description	
				also allows saving server-side alerts related to the widget	

Required permission:

- PULSE_WRITE_WIDGET

Available response representations:

- 200 - New widget was successfully created, redirect to newly created widget.
- 400 - Provided widget configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - widget has invalid format
  - `{ "message": "WIDGETS_LIMIT_EXCEEDED" }` - widgets limit exceeded
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the widget request
  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the widget request
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the widget request
- 403 - User does not have privileges.
- 404 - Widget configuration data does not exist, application/json with details:
  - `{ "message": "LAYOUT_NOT_FOUND" }` - specified widget layout does not exist

/api/wbrt/widgets/<guid>

Methods:

[+] GET

Returns widget configuration for widget with specified id and belonging to user or default widget.

Request query parameters:

name	required	type	default	description	
brief	no	boolean	true	When set to `false`, the widget is returned with layout definition and server-side	

name	required	type	default	description	
				alerts related to the widget. Otherwise just widget is returned.	

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED

Available response representations:

- 200 - application/json with widget configuration. See Widget example.
- 403 - User does not have privileges.
- 404 - Widget with requested id does not exist, application/json with details:
  - `{ "message": "WIDGET_NOT_FOUND" }`

[+] PUT

Update widget configuration for specified widget.

> **Important**
> The request does not create new widget if it does not exist

Request query parameters:

name	required	type	default	description	
brief	no	boolean	true	When set to `false`, the related layout is expected to be in the widget request; also allows saving server-side alerts related to the widget	

Available request representations:

- application/json with widget configuration. See Widget example.

Required permission:

- PULSE_WRITE_WIDGET

Available response representations:

- 200 - Widget was successfully updated
- 400 - Provided widget configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - widget has invalid format
  - `{ "message": "INVALID_LAYOUT_HASH" }` - specified layout has differ body_hash than provided in snapshot
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the widget request
  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the widget request
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the widget request
- 403 - User does not have privileges.
- 404 - Widget configuration data or widget itself does not exist, application/json with details:
  - `{ "message": "WIDGET_NOT_FOUND" }` - specified widget not exist
  - `{ "message": "LAYOUT_NOT_FOUND" }` - specified widget layout does not exist

[+] DELETE

Delete widget with specified guid.

Required permission:

- PULSE_WRITE_WIDGET

Available response representations:

- 200 - Widget was deleted
- 204 - Widget was already deleted
- 403 - User does not have privileges.

/api/wbrt/tabs

Methods:

[+] GET

Returns array of tabs for current user.

Request query parameters:

name	required	type	default	description	
brief	no	boolean	true	When set to `true`, returns only the set of tabs without widgets. Otherwise, returns tabs with the `widget` field containing full widget definition, including layout and related server-side alerts for all widget's presented on the tab.	
shared	no	boolean	false	When set to `true`, returns only shared dashboards (tabs with type `ttDashboard`). Otherwise, returns user tabs. User cannot change or delete widgets placed to shared dashboard unless he is owner of this shared dashboard.	
type	no	string		Optional filter for filtering by type, possible values: `ttDashboard`, `ttWidget`, `ttWallboard`.	

Required permission:

- PULSE_READ or READ_RESTRICTED

Available response representations:

- 200 - application/json with array of tabs.

- 403 - User does not have privileges.

[+] POST

Creates new tab.

Request query parameters:

name	required	type	default	description
brief	no	boolean	false	When set to `true`, widgets are not created according to widgets definitions (only positions are saved). Otherwise creates new widgets according to widgets definitions, each widget contains the related layout definition.
shared	no	boolean	false	When set to `true`, creates shared tab, which does not belong only to the current user. Otherwise, creates the tab which is available only for current user.
overwrite	no	boolean	false	When set to `true`, removes tabs with the same name and saved to the same location (only for shared tabs).

If you are going to create shared tab, then proxy_access_object field must be specified:

- proxy_access_object.dbid = 0 for sharing for everyone without creating proxy access object.
- empty proxy_access_object.dbid for creating proxy access object to control access:

```
{
    "body": {
        ...
        "proxy_access_object": {
        }
    }
}
```

You can specify dbid of folder where proxy access object must be created by proxy_access_object.folder_dbid. For folders in other tenant you have to specify proxy_access_object.tenant_dbid as well.

Available request representations:

- application/json with tab.

Required permission:

- PULSE_WRITE_TAB - for personal tab
- PULSE_SHARE_TAB - for shared tab

Available response representations:

- 200 - Tab was successfully created. Redirect to newly created tab. See Tab example.
- 400 - Provided tabs configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - request has invalid format.
  - `{ "message": "PROXY_ACCESS_OBJECT_REQUIRED" }` - request with shared = true but has no proxy access object specified.
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - tab with same name already exists in the folder. Retry with overwrite = `true` to remove duplicate tabs.
  - `{ "message": "TABS_LIMIT_EXCEEDED" }` - tabs limit exceeded.
- 403 - User does not have privileges.

[+] PUT

> **Important**
> Only for use with personal tabs. Please read method behaviour before using.

Stores array of tabs for current user in the following way:

1. Remove all tabs belonging to current user but not presented in the request.
2. Remove all widgets related to the removed tabs.
3. Update existing tabs belonging to current user and specified in the request.
4. Remove all existing but not presented in the request widgets.
5. Create not existing tabs but specified in the request.
6. If brief != `ture` create/update widgets with widget definitions.

Request with empty array removes all personal tabs. Request query parameters:

name	required	type	default	description	
brief	no	boolean	false	When set to `true`, widgets are not created/ updated with widgets definitions (only positions are saved). Otherwise creates/ updates widgets according to widgets definitions, each widget is expected to contain related layout definition.	

Available request representations:

- application/json with tab **array**.

Required permission:

- PULSE_WRITE_TAB - for personal tab
- PULSE_SHARE_TAB - for shared tab

Available response representations:

- 200 - User's tabs were saved. Redirect to to the updated tab. See Tab example.
- 400 - Provided tabs configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - the request has an invalid format
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the request
  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the request
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the request
  - `{ "message": "WIDGETS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and widget is presented in the request
- 403 - User does not have privileges.

/api/wbrt/tabs/<guid>

Methods:

[+] GET

Returns tab with specified guid.

Request query parameters:

name	required	type	default	description	
brief	no	boolean	false	When set to `true`, returnes tab without widgets definition (only positions property filled). Otherwise, returns tab with field `widget`, containing full widget definitions (including related layout and related server-side alerts) for all widget's presented on the tab.	

Required permission:

- PULSE_READ or READ_RESTRICTED

Available response representations:

- 200 - application/json with tab. See Examples.
- 403 - User does not have privileges.
- 404 - Tab does not exist or user doesn't have access to it.

[+] PUT

Update tab with specified guid. This method removes widgets and their layouts from old tab if newer tab doesn't have references to them.

> **Important**
> The request does not create new tab if it does not exist.

name	required	type	default	description
brief	no	boolean	false	If specified and "true" then widgets will not be created\updated with widgets definitions (only positions will be saved). Otherwise will create\ update widgets according to widgets definitions.
overwrite	no	boolean	false	if true then tabs with same name and saved to same location will be removed (only for shared tabs)

The method can be used to move tab between folder. Execute with proxy_access_object.dbid = 0 to make it as shared for everyone, or specify target folder in proxy_access_object.folder_dbid. For moving between tenants also specify proxy_access_object.tenant_dbid.

Available request representations:

- application/json with tab.

Required permission:

- PULSE_WRITE_TAB - for personal tab
- PULSE_SHARE_TAB - for shared tab

Available response representations:

- 200 - Tab was updated successfully.
- 400 - Provided tabs configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - request has invalid format
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - tab with same name already exists in the folder. Retry with overwrite = `true` to remove duplicate tabs.
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the request.

  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the request.
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the request.
  - `{ "message": "WIDGETS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and widget is presented in the request.
- 403 - User does not have privileges or access to change this tab.
- 404 - Tab with specified guid does not exist or user does not have read access to this tab.

[+] DELETE

Delete tab with specified guid. This method removes related widgets and their layouts before removing the tab.

Required permission:

- PULSE_WRITE_TAB

Available response representations:

- 200 - Tab was successfully deleted.
- 204 - Tab is already deleted or user doesn't have read access to this tab.
- 403 - User does not have privileges or access to delete this tab.

/api/wbrt/users

Methods:

[+] GET

Returns all Genesys Pulse users.

Required permission:

- PULSE_WRITE_USER

Available response representations:

- 200 - Application/json with users.
- 403 - User does not have privileges.

/api/wbrt/users/<guid>

Methods:

[+] GET

Returns user with specified guid.

Request query parameters:

name	required	type	default	description	
brief	no	boolean	true	When set to `false`, the user is returned with the `tab` field containing full personal tabs definitions. When not specified or set to `true`, the user is returned without personal tabs definitions.	

Required permission:

- PULSE_WRITE_USER

Available response representations:

- 200 - Application/json with user.
- 403 - User does not have privileges.
- 404 - User does not exist.

Example

brief=`true`

```
{
    "body": {
        "guid": "5efe4b7c07b3-b636-11ea-fe84-6719e62a",
        "username": "default",
        "tab_guid": [
            "961f0facee94-a9d4-11ea-fe84-681b5081",
            "f22c1537b09b-a43b-11eb-0d5e-69cb9b37"
        ],
        "widgets_number": 1,
        "preferences": {
            "language": "en-us",
            "time_zone": ""
        }
    }
}
```

[+] DELETE

Delete a user with the specified guid. This method removes related tabs, widgets, and their layouts before removing the user.

Required permission:

- PULSE_WRITE_USER

Available response representations:

- 200 - User was successfully deleted.
- 204 - User is already deleted.
- 403 - User does not have privileges.

/api/wbrt/import

Methods:

[+] POST

Import entities provided in request body.

Example of request body for importing data exported with the export service:

```
{
    "data": <result of export>
}
```

Example of request body for importing templates:

```
{
    "data": {
                "template": [{
                        "definition": <template_definition>
                }, {
                        "definition": <template_definition>
                }, ...]
    }
}
```

Example of request body for importing tabs:

```
{
    "data": {
                "tab": [{
                        "body": <template_body>
                }, {
                        "body": <template_body>
                }, ...],
                "widget": [{
                        "body": <widget_body>
                }, {
```

```
                "body": <widget_body>
            }, ...],
            "layout": [{
                "definition": <layout_definition>
            }, {
                "definition": <layout_definition>
            }, ...]
    }
}
```

Required permission:

- PULSE_WRITE_TEMPLATE - for importing templates
- PULSE_SHARE_TAB - for importing tabs

Available response representations:

- 200 - application/json with imported entities.
- 400 - Provided import data is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - request has invalid format
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - entity with same name already exists in the folder. Retry with specified overwrite:
    opOverwrite - overwrite old entity in case of name conflict
    opSkip - skip in case of conflict
    opRename - rename new entity in case of name conflict
- 403 - User does not have privileges or access permissions to import one or more entity to the destination folder.

/api/wbrt/export

Methods:

[+] POST

Export entities according to the query provided in request body.

Example of request body for exporting all Wallboards:

```
{
    "tab": [{"type": "ttWallboard"}]
}
```

Example of query for exporting all shared Dashboards and Templates:

```
{
    "tab": [{"type": "ttWallboard", "proxy_access_object": {}}],
    "template": [{}]
}
```

Example of query for exporting Dashboards\Wallboards with given guids:

```
{
```

```
    "tab": [{"guid": "<guid_1>"}, {"guid": "<guid_2>"}]
}
```

Example of query for exporting Templates with given guids:

```
{
    "template": [{"guid": "<guid_1>"}, {"guid": "<guid_2>"}]
}
```

Example of query for exporting all templates with certain type:

```
{
    "template": [{"layout_type": "ltPCREGULAR"}]
}
```

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED

Available response representations:

- 200 - application/json with export result.
- 403 - User does not have privileges.

/api/plugins/wbrt/health

Methods:

[+] GET

Checks Genesys Pulse application's health. Genesys Pulse is determined as healthy when:

- There is a working connection to Genesys Pulse Database.
- There is a working connection to Configuration Server.

This service caches the previous check result and keeps it according to the value of the **pulse/health_expire_timeout** option.

Health of Genesys Pulse Collectors is not checked here, use use /api/plugins/wbrt/health/detail to get health information about

Required permission:

- no permissions required

Available response representations:

- 200 - Genesys Pulse application works fine.
- 503 - Genesys Pulse health check failed, please, use use /api/plugins/wbrt/health/detail service to get

more information.

/api/plugins/wbrt/health/detail

Methods:

[+] GET

Returns Genesys Pulse application health details.

Genesys Pulse application health details include:

- State of connection to Configuration Server.
- State of connection to the Genesys Pulse Database.
- State of Genesys Pulse Collectors and their health details, including:
  - State of connection to Configuration Server.
  - State of connection to the Genesys Pulse Database.
  - State of connection to Stat Servers.
  - Starting with release 9.0.004.03, state of Stat Server utilization:
    - Total Statistic Count - total number of statistic requests that are either already opened, or pending, or not sent to Stat Server yet.
    - Opened Statistic Count - the number of requests that are successfully opened.
    - Failed Statistic Count - the number of requests that failed.

This service caches the previous check result and keeps it according to the value of the **pulse/health_expire_timeout** option.

Required permission:

- no permissions required

Available response representations:

- 200 - application/json with Genesys Pulse application health details. Including:
  - **snapshotWritingStatus**—Snapshots are written to the file system. The snapshot generation and writing process itself does not depend on present/missing connections. If a connection to Stat Server is missing, the data may be all N/A, but snapshots are still generated and written.
  - **pendingLayoutStatusChangesCount**—The number of pending layouts to activate, deactivate, or recheck. It depends on the connectivity to DB Server. If the connection is present, this number changes, as layout are being activated/deactivated/rechecked. Stalled nonzero number may indicate missing connection to DB Server. Genesys Pulse Collector needs to read changed layouts and new layout definitions, which requires active connection to DB Server.
  - **maxStatisticValueDelay**—This value depends on the number of layouts, their refresh period, performance of CPU, sufficient memory availability (no swap operations caused by Genesys Pulse Collector), performance of the file system. There is no recommended value; however, most layouts

typically have the same refresh interval. If the setting of the `maxStatisticValueDelay` is higher than some part of this interval (say 25-50%), then you want to consider increasing resources. Values do not indicate that something is totally wrong, but may indicate that Genesys Pulse Collector can perform better if it runs with more elevated settings (more processing threads) and faster drives/file systems for the snapshots storage.

Example of the service response:


```
{
 "configServer":{
  "connected":true
 },
 "database":{
  "connected":true
 },
 "collectors":{
  "Collector_9.0.006":{
   "currentTimestamp": "1600369952",
   "startupTimestamp": "1600159972",
   "uptime": "209980",
   "connectionStatus": {
    "configServer": {
     "connected": true,
     "instance": "primary",
     "connectionTimestamp": "1600159973",
     "disconnectTimestamp": "0"
    },
    "statServer0": {
     "present": true,
     "connected": false,
     "connectionTimestamp": "0",
     "disconnectTimestamp": "1600159974",
     "totalStatisticCount": "60",
     "openedStatisticCount": "0",
     "failedStatisticCount": "0"
    },
    "statServer1": {   "present": true,   "connected": false,   "connectionTimestamp": "0",
"disconnectTimestamp": "1600159974",   "totalStatisticCount": "60",   "openedStatisticCount":
"0",   "failedStatisticCount": "0"
    },
    "dbServerConnection": "online",
    "dbConnection1": {
     "connected": true,
     "connectionTimestamp": "1600159975",
     "disconnectTimestamp": "0"
    },
     "dbConnection2": {
      "connected": true,
      "connectionTimestamp": "1600159975",
      "disconnectTimestamp": "0"
     }
    },
   "snapshotWritingStatus": true,
   "pendingLayoutStatusChangesCount": "0",
   "maxStatisticValueDelay": 0,
   "version": "2019.10.10.00"
  }
 }
}
```

Troubleshooting

[+] Snapshot was populated, but related Widget shows only spinner instead of data from the snapshot.

LayoutState message has special fields body_hash_1/body_hash_2.

These fields are intended to indicate compatibility between LayoutDefinition and LayoutSnapshot.

So body_hash_1/body_hash_2 are populated in LayoutInfo.state (with hash value calculated for LayoutDefinition) each time when LayoutInfo is saved (via POST/PUT or via Widget wizard).

Same values must be inserted to LayoutSnapshot.state by data producer. This lets Genesys Pulse know that data from LayoutSnapshot is legal for current state of LayoutDefinition.

Important

Starting with **release 8.5.104.05**, Genesys Pulse validates value of body_hash_1 in snapshot against value in the layout.

So when you trying to post snapshot with incorrect body_hash_1 value you will get 400 BAD REQUEST with message INVALID_BODY_HASH in the response body.

You can ommit body_hash_1 value in the snapshot and it will not be used for detecting layout change.

Then widget will continue to show data from such snapshot even something changed in the layout.

For example if you select/unselect some objects widget will show these objects until new snapshot without them is generated by Genesys Pulse Collector.

The widget is updated too infrequently or too frequently

LayoutDefinition message has special refresh_interval field indicates how often data producer must provide new data.

The same refresh_interval field is presented in the LayoutSnapshot and indicates how often data producer provides data actually.

Genesys Pulse UI uses LayoutSnapshot.refresh_interval for calculation of delay before each next data request:

Refresh Delay Calculation

```
var delay = snapshot.refresh_interval || 60, //use refresh interval from data provider or 60
sec
    gap = 2,
    genTime = snapshot.timestamp, //must be filled by data provider
        readTime = snapshot.read_timestamp; //filled by Genesys Pulse BE on each snapshot
request

var d = readTime - genTime - gap;
if ((d > 0) && (d < delay)) {
```

```
    delay = delay - d;
}
```

> **Important**
>
> LayoutSnapshot.refresh_interval field must be provided by data producer. It can be smaller/greater than in related LayoutDefinition.

Examples

[+] LayoutInfo Example

```
{
    "definition": {
        "guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
        "refresh_interval": 15,
        "layout_type": "ltGENERIC",
        "column": [
            {
                "category": "ccDIMENSION",
                "is_delta_key": true,
                "id": "_Object$ID"
            },
            {
                "category": "ccDIMENSION",
                "id": "_Object$Name",
                "format": "string"
            },
            {
                "category": "ccMEASURE",
                "vt": "vINT",
                "id": "First_column",
                "type": "ctGENERIC",
                "format": "integer",
                "label": "First"
            },
            {
                "category": "ccMEASURE",
                "vt": "vINT",
                "id": "Second_column",
                "type": "ctGENERIC",
                "format": "time",
                "label": "Second"
            },
            {
                "category": "ccMEASURE",
                "vt": "vSTR",
                "id": "Third_column",
                "type": "ctGENERIC",
                "format": "string",
                "label": "Third"
            }
        ],
        "default_widget": {
            "threshold": [
```

```
                {
                    "value": [
                        0,
                        0,
                        0
                    ],
                    "direction_up": false,
                    "column_id": "First_column"
                }
            ],
            "view": [
                {
                    "column_selector": [
                        "Second_column"
                    ],
                    "sorting": [
                        {
                            "is_asc": true
                        }
                    ],
                    "type": "BarView"
                }
            ],
            "size_x": 1,
            "size_y": 4,
            "label": "Test  Widget"
        },
        "name": "Third party source",
        "description": "Example of third party source"
    },
    "state": {
        "current_status": "stACTIVE",
        "body_hash_1": 583854940,
        "body_hash_2": 583854940
    },
    "record": {
        "timestamp": 1443695496,
        "username": "pulse"
    }
}
```

[+] LayoutSnapshot Example

```
{
    "layout_guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
    "col": [
        {
            "v": [
                101,
                102,
                103,
                104,
                105,
                106,
                107,
                108,
                109,
                110
            ],
            "col": {
                "category": "ccDIMENSION",
                "is_delta_key": true,
                "vt": "vINT",
```

```
            "id": "_Object$ID"
        }
    },
    {
        "v": [
            "Agent1",
            "Agent2",
            "Agent3",
            "Agent4",
            "Agent5",
            "Agent6",
            "Agent7",
            "Agent8",
            "Agent9",
            "Agent10"
        ],
        "col": {
            "category": "ccDIMENSION",
            "vt": "vSTR",
            "id": "_Object$Name"
        }
    },
    {
        "v": [
            60,
            120,
            180,
            234,
            123,
            531,
            521,
            231,
            541,
            634
        ],
        "col": {
            "category": "ccMEASURE",
            "vt": "vINT",
            "id": "First_column",
            "type": "ctGENERIC",
            "format": "integer",
            "label": "First"
        }
    },
    {
        "v": [
            523,
            0,
            312,
            543,
            631,
            434,
            423,
            642,
            743,
            135
        ],
        "col": {
            "category": "ccMEASURE",
            "vt": "vINT",
            "id": "Second_column",
            "type": "ctGENERIC",
            "format": "time",
```

```
                "label": "Second"
            }
        },
        {
            "v": [
                "Value1",
                "Value2",
                "Value3",
                "Value4",
                "Value5",
                "Value6",
                "Value7",
                "Value8",
                "Value9",
                "Value10"
            ],
            "col": {
                "category": "ccMEASURE",
                "vt": "vSTR",
                "id": "Third_column",
                "type": "ctGENERIC",
                "format": "string",
                "label": "Third"
            }
        }
    ],
    "state": {
        "current_status": "stACTIVE",
        "body_hash_1": 583854940,
        "body_hash_2": 583854940
    },
    "layout_type": "ltGENERIC",
    "timestamp": 1409066836
}
```

[+] Widget Example

```
{
        "body": {
                "guid": "28cb32f0-6d8c-4be9-9ee5-1e2fa5f91db2",
            "layout_guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
            "label": "MyWidget",
            "size_x": 1,
            "size_y": 4,
            "view": [{
                "type": "BarView",
                "column_selector": ["Login_Duration"],
                "sorting": [{
                    "is_asc": false
                }]
            }]
    }
}
```

[+] Tab Example

```
{
    "body": {
        "guid": "c8cb42f0-1d8c-4be9-9ee5-1e2fa5f91db2",
        "title": "Dashboard",
        "type": "ttDashboard",
        "position": [{
```

```
            "widget_guid": "c4bff0ce-d842-4070-be48-36a2cf6e0630",
            "col": 1,
            "row": 1
        }],
        "widget": [{
            "body": {
                "guid": "c4bff0ce-d842-4070-be48-36a2cf6e0630",
                "layout_guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
                "label": "MyWidget",
                "size_x": 1,
                "size_y": 4,
                "view": [{
                    "type": "BarView",
                    "column_selector": ["Login_Duration"],
                    "sorting": [{
                        "is_asc": false
                    }]
                }]
            }
        }]
    },
    "record": {
        "timestamp": 1443695496,
        "username": "pulse"
    }
}
```


[+] Tab Example (brief=true)


```
{
    "body": {
        "guid": "c8cb42f0-1d8c-4be9-9ee5-1e2fa5f91db2",
        "title": "Dashboard",
        "type": "ttDashboard",
        "position": [{
            "widget_guid": "c4bff0ce-d842-4070-be48-36a2cf6e0630",
            "col": 1,
            "row": 1
        }]
    },
    "record": {
        "timestamp": 1443695496,
        "username": "pulse"
    }
}
```

Health Check

The Genesys Pulse Web Service API gives you the ability to perform health checks.

See the following responses in **/api/plugins/wbrt/health/detail**:

- **snapshotWritingStatus** — Snapshots are written to a file system, and are generated and written even if external connections are not available. If the connection to Stat Server is not available, data can be displayed as `N/A`, but snapshots are generated and written.
- **pendingLayoutStatusChangesCount** — This is a summary number of pending layouts to activate, deactivate, or recheck. This process depends on the availability of the connection to DB Server:
  - If the connection is present, this value changes as layout activation/deactivation/recheck operations are processed.
  - If the connection is not present, this value does not change.

    Genesys Pulse Collector needs to read changed layouts and new layout definitions, which requires an active connection to DB Server.
- **maxStatisticValueDelay** — This value varies depending on the number of layouts, their refresh period, CPU performance, memory availability (no swap operations caused by Genesys Pulse Collector), and performance of the file system.

  There is no expected / recommended value for maxStatisticValueDelay; however, you can expect most layouts to have roughly the same refresh interval. If the value of maxStatisticValueDelay is significantly larger than this interval (for example 25-50% larger), consider increasing resources. Genesys Pulse Collector can perform better if it runs with a sufficient number of processing threads and fast drive/file system on which to store snapshots.

Starting and Stopping Genesys Pulse

There are several ways to start and stop Genesys Pulse, depending on your operating system.

Linux

Linux users can start and stop Genesys Pulse by using one of the following methods:

- The System Dashboard in GAX.
- Genesys Administrator
- Solution Control Interface (SCI)

Windows

Windows users can start and stop Genesys Pulse by using one of the following methods:

- The System Dashboard in GAX.
- Genesys Administrator
- Solution Control Interface (SCI)
- Restarting the Genesys Pulse Windows Service

Prerequisites

Confirm the following requirements before starting Pulse:

- See Supported Browsers in the Genesys Supported Operating Environment Reference.
- Your monitor must be set to a resolution of no less than 1024x768.
- The DB Server for Genesys Pulse Collector must be running.
- The RDBMS for the Genesys Pulse database must be running.
- Configuration Server and the DB Server for Configuration Server must be running.

> **Important**
> Users without permission to edit their own application object cannot save their

Custom dashboard configuration. Users without permission to edit both their own application object and their Tenant object cannot save the Default dashboard configuration.

For example, such a user can make changes, such as creating widgets, but cannot save the changes. The next time the page is loaded, the changes are lost.

See the steps to configure user access for details.

Genesys Pulse User Interface Extensions

Introduction

This article describes Genesys Pulse user interface (UI) extensions. The main purpose of these extensions is to support the 3rd-party widgets which are developed separately and installed as a plug-in. In addition, you can load external .js and .css files in order to customize existing widgets.

Important

- The Genesys Pulse extension API is supported on a best-efforts basis. Any customer attempting to use or develop extensions must have expertise to do so and agree to troubleshoot them on their own. This API is subject to change in future versions at Genesys' discretion without prior notice.
- Be extra cautious with extensions that are loading third-party JavaScript libraries as they can break Genesys Pulse functionality.

How to Create Your Own Plug-in

Starting with release 8.5.108, you can create the Genesys Pulse plug-in skeleton using the `plugin-generator.js` utility script from the **plugin-sdk\libs** directory of the Genesys Pulse installation package. This script is tested with the latest nodejs version 8, but should also work with the nodejs version 6.5 or higher.

For example:

```
node plugin-generator.js
Enter plugin name (full plugin name will be pulse-plugin-<name>):
> My Funnel
Enter plugin version:
> 1.0.0
Enter plugin description:
> My Description
Creating ./pulse-plugin-my-funnel/pom.xml
Creating ./pulse-plugin-my-funnel/src/main/resources/META-INF/applicationContext.xml
Creating ./pulse-plugin-my-funnel/src/main/java/com/genesyslab/wbrt/plugin/
PulsePluginMyFunnel.java
Creating ./pulse-plugin-my-funnel/src/main/resources/web/manifest.js
Creating ./pulse-plugin-my-funnel/src/main/resources/web/pulse.manifest.js
Plugin structure has been created
```

After that, you need to insert your extension implementation into a generated pulse.manifest.js file.

To build a plug-in you need to have Java and Maven installed. Maven dependencies are located in the **plugin-sdk\libs** directory of the Genesys Pulse installation package. You can install them by running the installation script from the same directory.

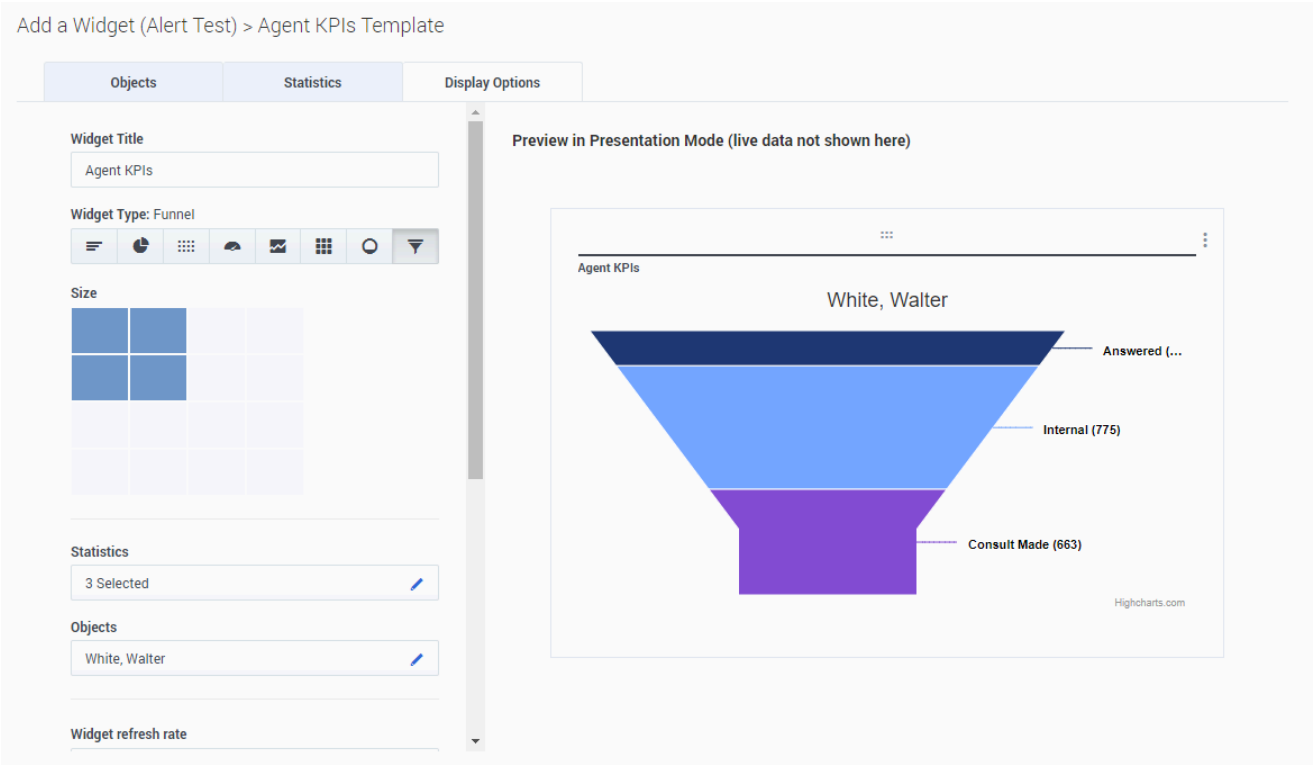
After that you are ready to build a plug-in:

```
cd pulse-plugin-my-funnel
mvn clean install
```

Now the plug-in is ready to be deployed into Genesys Pulse. Copy it to the **<PULSE_PATH>/webapp/WEB-INF/lib/** and restart Genesys Pulse.

> ### Tip
>
> To speed up the development process you do not need to rebuild and deploy your plug-in after each change. You can modify an already deployed `pulse.manifest.js` file in the **<PULSE_PATH>/webapp/plugins/<your_plugin_name>/** directory. To apply any change you just need to reload the Genesys Pulse page. When you are finished, ensure that you copy the final `pulse.manifest.js` file back to your project directory located at **<your_plugin_path>/src/main/resources/web/**, otherwise you might lose all your changes when Genesys Pulse is restarted.

Now the new Funnel chart is available among other widget types:



Genesys Pulse User Interface Extension API

Each `pulse.manifest.js` should contain one or more immediately-invoked function expression (IIFE) and use the global `window.pulse` function to register IIFEs.

The below example shows the registering of a custom widget which uses the `Highcharts` library to draw a custom chart.

```
(function() {
    pulse.extension({
        type: 'WIDGET',             // type of extension allows us to add more extension
types in the feature
        apiVersion: '9.0.0',        // version of api used in the extension
        id: 'CustomWidgetOne',      // unique extension id, should not clash with other
deployed extensions
        label: 'Custom Widget One', // label displayed in Display Options of the Widget Wizard
        icon: 'icon-app-chart',     // icon displayed in Display Options of the Widget Wizard
        require: [                  // javascript or css files to be loaded for the extension
work
                                    // use object when your library exposes global variable
                                    // no need to load d3 (version 3.5.17), jQuery,
underscore - they are already loaded by Genesys Pulse
                                    // to avoid loading library from CDN put library side by
side with pulse.manifest.js and provide URL like "../pulse-plugin-name/library.js"
             {Highcharts: "https://code.highcharts.com/highcharts.js"},
             "https://code.highcharts.com/highcharts-more.js"
        ],
        render: function (element, data, options) { // key function for rendering widget
content
            ... //put your rendering code here
            return true; // return true when widget content is rendered or false when widget
cannot be rendered
        },
        resize: function (element, data, options) { // function to be called when widget
being resized
            ... //put your rendering code here
        },
        constraints : { // constraints for widget configuration
           dashboardSupport: true, // allows to select widget on dashboard, enabled by default
           wallboardSupport: true, // allows to select widget on wallboard, disabled by
default
           size: { // define min and max size for the widget
              minX: 1, // min horizontal size
              minY: 1, // min vertical size
              maxX: 1, // max horizontal size
              maxY: 2  // max vertical size
           },
           objects: { // define the min and max objects which can be selected by the users to
adjust the real-state of the visualization
              min: 1, // require at least one object selected
              max: 1  // allow to select no more than one object
           },
           statistics: { // define the min and max stats which can be selected by the users
to adjust the real-state of the visualization
              min: 1, // require at least one statistic selected
              max: 10 // allow to select no more than ten statistic
           }
        },
        containerClass: "my-chart", // optional, specify custom css class for widget container
        containerStyle: {           // optional, overwrite widget container style
           "padding-bottom": "10px"
        }
```

```
    });
}();
```

The important **render** function accepts three arguments:

- **element**—HTMLElement, where the content of a widget should be rendered.
- **data**—the data from the snapshot formatted for easier use by new developers.
- **options**—additional options, such as current selected statistics and objects, widget size, current locale.

The **render** function is called each time the content of a widget is redrawn with new **data**. When a widget is resized the **resize** function is called with the same three arguments.

The **data** object has the following format:

```
{
  statistics: [{         // array with all statistics selected in Statistics of the Widget
Wizard
    format: "time",      // statistic format
    id: "Ready_Time",    // statistic internal name
    label: "Ready Time", // display name of statistic
    ranges: {            // describes threshold ranges for particular statistic
      green: {
        from: 100,
        color: "green"
      },
      orange: {
        to: 100,
        from: 20,
        color: "yellow"
      },
      red: {
        to: 20,
        color: "red"
      }
    },
    values: [ // array with statistic values per each object, values are ordered according to
objects order in objects array
      6470,
      6435,
      6467
    ]
  }],
  objects: [{ // array with all objects selected in Objects of the Widget Wizard
      id: 103,
      label: "Master, Yoda"
    },
    {
      id: 104,
      label: "Darth, Vader"
    },
    {
      id: 9638,
      label: "Luke, Skywalker"
    }
  ],
  history: { // object provides access to historical values of statistic, available since
Genesys Pulse version 9.0.0
    get: function(statisticId, objectId, options) { ... } // method returning array with
historical data for particular statistic and object
    // additional parameters can be passed via options object:
```

```
    // options.start - optional, allows to specify left time boundary of historical data
represented either by Date object or by milliseconds
    // example of method call result: [{time: 1517845138005, value: 1}, {time: 1517845156610,
value: 42}, ...]
  }
}
```

This format is self-describing, easy to understand and easy to transform into input for popular charting libraries like `Highcharts`.

The **options** object has the following format:

```
{
  selectedObjects: [ 103, 104 ],        // array with ids of objects selected in Display
Options to show in chart
  selectedStatistics: [ "Ready_Time" ], // array with ids of statistics selected in Display
Options to show in chart
  theme: {                              // object describes current selected color theme,
could be used to pick Genesys approved colors for charts
    backgroundColor:"#fdfdfd",
    chartColors: [ ... ]
    rangeColors: {
      green:"#4ac764"
      orange:"#f8a740"
      red:"#ea4f6b"
   }
}
```

How to Change Basic Styles

Starting with release 8.5.106, you can customize .js and .css files to apply a new set of colors for the Genesys Pulse instance.

You can place js/css files into the Genesys Pulse folder or **styles/scripts** subfolders. The options for loading custom js/css files are as follows:

- `pulse/load_css_custom=/pulse/styles/custom.css`
- `pulse/load_js_custom=/pulse/scripts/custom.js`

You can specify many options with `load_css/load_js` prefixes:

- `pulse/load_css_<style_name_1>=/pulse/styles/custom_1.css`
- `pulse/load_css_<style_name_n>=/pulse/scripts/custom_n.css`
- `pulse/load_js_<script_name_1>=/pulse/styles/custom_1.js`
- `pulse/load_js_<script_name_n>=/pulse/scripts/custom_n.js`

The options do not require Genesys Pulse restart.

> **Tip**
>
> Files placed in Genesys Pulse folders or **styles/scripts** subfolders can be overwritten during the upgrade. It is better to add these files to a different directory accessible by the http(s) protocol.

You can use a full URL when files are hosted on another web server:

- pulse/load_css_custom=http://<host>:<port>/mystyle.css
- pulse/load_js_custom=http://<host>:<port>/myscript.js

There are samples in the **examples** folder where Genesys Pulse is installed, in the **custom-css-example** and **custom-js-example** subfolders. These samples are intended for advanced users, who can do their own customization of provided examples.

The font size in the `Text` widgets is designed to scale the font according to the content. However, you can use a custom .css to customize it if you have a competent web designer or with the help of Genesys Professional Services.

Starting and Stopping Genesys Pulse Collector

You can start Genesys Pulse Collector on a Windows or UNIX platform. Invoking Genesys Pulse Collector starts a series of internal checks for proper configuration. The value of the `max-output-interval` option, for example must be greater than the value of the `min-output-interval` option or Pulse Collector exits. Verify your Collector Application object for proper configuration before starting Genesys Pulse Collector.

Prerequisites

The following must be running prior to starting Genesys Pulse Collector:

- Backup Configuration Server

  To make sure that Genesys Pulse Collector can connect to backup Configuration Server on startup (for example, when the primary Configuration Server application switched to the backup mode or stopped at the moment when Genesys Pulse Collector starts/restarts), you must specify backup Configuration Server parameters when starting Genesys Pulse Collector.

  On the command line, specify these parameters using the following two arguments:

  - -backup_host hostname
  - -backup_port port-number

- National Characters

  To use national characters in the string properties correctly, Genesys Pulse Collector determines which multibyte encoding is used by Configuration Server. You can allow Genesys Pulse Collector to use the default setting or specify the characters by editing the following configuration options:

  - **Windows**

    You can specify Configuration Server multibyte encoding using the following command-line parameter:

    - `-cs_`codepage following the Windows code page number (for example, 1251). For more information about Windows code pages and their numbers, see https://msdn.microsoft.com/en-us/library/windows/desktop/dd373814(v=vs.85).aspx If the parameter is not specified, Genesys Pulse Collector assumes Configuration Server uses the code page that corresponds to the locale of the Windows operating system where Genesys Pulse Collector is running.

      Example:

      If your Configuration Server has utf-8 encoding and your Windows server (where Collector should run) has Arabic encoding.

      To have national characters displayed correctly, you must do following:

      1. Open in browser page http://msdn.microsoft.com/en-us/library/windows/desktop/

dd317756(v=vs.85).aspx mentioned above

2. Find which Windows code page corresponds to the Configuration Server encoding. We can see that UTF-8 corresponds to the code page 65001.
3. Add following command-line option to Collector: `-cs_codepage 65001`.

- **Linux**

  You can specify Configuration Server multibyte encoding through the command-line parameter `-cs_encoding` following the iconv-compatible symbolic name of encoding (for example, UTF-8).

  To display the list of encodings supported on the your system, enter the following command in the Linux console:

  ```
  iconv --list
  ```

Start Genesys Pulse Collector

You can start Genesys Pulse Collector on any of the supported platforms.

Solution Control Interface (SCI)

You can start Genesys Pulse Collector:

[+] From SCI.

To start Genesys Pulse Collector from the Solution Control Interface (SCI):

1. From the `Applications` view, select your Genesys Pulse Collector `Application` object in the list pane.
2. Click the `Start` button on the toolbar, or select `Start` either from the Action menu or the context menu. (Right-clicking your `Application` object displays the context menu.)
3. Click Yes in the confirmation box that appears. Your Genesys Pulse Collector application starts.

Windows

On a Windows platform, you can start Genesys Pulse Collector:

[+] Manually from the Programs menu as an application.

To start Genesys Pulse Collector from the `Programs` menu as an application, select `Start Pulse Collector` from the program group created during installation. The application opens a console window and automatically issues the parameters specified during configuration to start Genesys Pulse Collector. The Genesys Pulse Collector application name, version, and connectivity parameters appear in the title bar.

[+] Manually from a console window as an application.

1. To start Genesys Pulse Collector as an application from a console window: At the command-line prompt, go to the directory where Genesys Pulse Collector has been installed. Type the name of the Genesys Pulse Collector executable followed by the appropriate command-line parameters using the following

syntax:

```
collector.exe -host hostname -port portno -app application
```

where:

- `hostname` refers to the name of the computer on which Configuration Server is running.
- `portno` refers to the communication port on which Configuration Server is running.
- `application` refers to the name of the Genesys Pulse Collector `Application` object as defined in Genesys Administrator.

> **Important**
> If the host or application name contains spaces or hyphens (–), enclose it in double quotation marks.

For example, to start Genesys Pulse Collector with parameters specifying the host as cs-host, port as 2020, and name as Pulse Coll, type:
`collector.exe -host "cs-host" -port 2020 -app "Pulse Coll"`

> **Important**
> If needed, specify the optional parameters -backup_host, -backup_port, -cs_codepage, and -cs_encoding.

[+] As a Windows service.

1. From the task bar, choose `Start - Administrative Tools > Computer Management`.
2. Open `Services and Applications > Services`.
3. Right-click your Genesys Pulse Collector service from the list and select `Start`.

> **Important**
> Since the Local Control Agent (LCA) can be installed as a Windows service with the user interface disabled, all servers started through SCI, in this case, are started without a console unless you specifically select the Allow Service to Interact with Desktop check box for both LCA and Genesys Pulse Collector.

UNIX Platforms

You can start Genesys Pulse Collector:

[+] Manually from UNIX Platforms.

1. Go to the directory where Genesys Pulse Collector has been installed.

> **Important**
> You can invoke Genesys Pulse Collector only from the directory in which it was installed.

2. Type the name of the Genesys Pulse Collector executable followed by the appropriate command-line parameters using the following syntax:
   `./collector -host hostname -port portno -app application`

   where:

   - `hostname` refers to the name of the computer on which Configuration Server is running.
   - `portno` refers to the communication port on which Configuration Server is running.
   - `application` refers to the name of the Genesys Pulse Collector Application object as defined within Genesys Administrator.

   > **Important**
   > If the host or application name contains spaces or hyphens (–), enclose it in double quotation marks.

   For example, to start Genesys Pulse Collector with parameters specifying the host as `cs-host`, port as `2020`, and name as `Pulse Coll`, type:
   `./collector -host "cs-host" -port 2020 -app "Pulse Coll"`

   > **Important**
   > If needed, specify the optional parameters -backup_host, -backup_port, and -cs_encoding.

Configure the `stdout` option in the `log` section of `collector` options to write to a log file, so that you can check for errors in its configuration if Genesys Pulse Collector fails to start. If you cannot resolve a problem, contact Genesys Customer Care and provide the entire content of the log.

You can also type the name of the Genesys Pulse Collector executable and its command-line parameters into a shell script and execute the script using the following command:
`./run.sh [Name of script]`

To redirect Genesys Pulse Collector output (on most UNIX shells), use the following syntax:
`./collector -host hostname -port portno -app appl > log_file.log`

To have both log file and console, within Genesys Administrator add the following to Genesys Pulse Collector's application properties:

- Section: Log
- Option: `all` with the following value:
  `stdout,<log_file_name.log>,network`
  Separate values with commas. Instead of `stdout`, you can also specify `stderr`.

Genesys Pulse Collector writes messages to <log_file_name.log> in the same directory where Genesys Pulse Collector is installed. To have Genesys Pulse Collector write to a different location, specify the full path for this parameter.

Stop Genesys Pulse Collector

You can stop Genesys Pulse Collector on any of the supported platforms.

> **Important**
> Be sure that the autorestart property is cleared for the Genesys Pulse Collector Application object in Genesys Administrator.

Solution Control Interface (SCI)

You can stop Genesys Pulse Collector:

[+] From SCI.

If you use LCA and Solution Control Server (SCS), you can stop Genesys Pulse Collector from SCI. To do so:

1. From the `Applications` view, select your Genesys Pulse Collector `Application` object in the list pane.
2. Click `Stop` on the toolbar, or select `Stop` either from the Action menu or the context menu.
3. Click Yes in the confirmation box that appears.

On a Windows platform

On a Windows platform, you can stop Genesys Pulse Collector:

- **[+] Manually from its console window as an application.**

  If Genesys Pulse Collector is running as an application, switch to its console window and press `Control-C` to stop it.

- **[+] As a Windows service.**

  If you are running Genesys Pulse Collector as a Windows service, you should stop it only from the Microsoft Management Console.

  To stop Genesys Pulse Collector running as a Windows service:

  1. From the task bar, choose `Start - Administrative Tools > Computer Management`.
  2. Open Services and `Applications > Services`.
  3. Right-click your Genesys Pulse Collector service from the list and select `Stop`.

- If LCA and SCS are used, you can stop Genesys Pulse Collector from SCI.

On a Unix platform

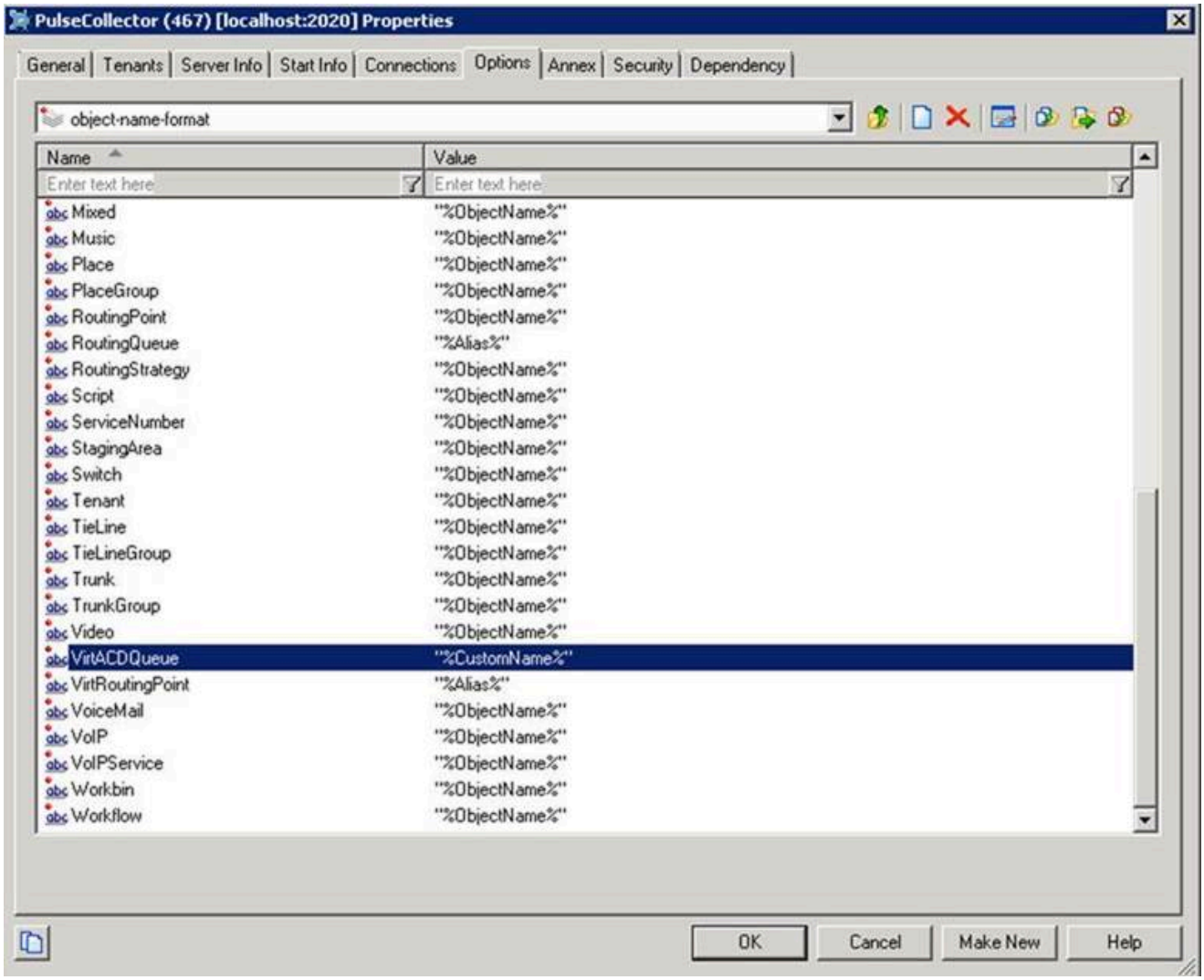
Stop Genesys Pulse Collector on UNIX using either of the following methods:

- On the command line, type `kill -9 processid`, where `processid` is Genesys Pulse Collector's UNIX process ID.
- Press `^C` from the active Genesys Pulse Collector console.
- If LCA and SCS are used, you can stop Genesys Pulse Collector from SCI.

Change Object Names

You may want a different name for an object in your Genesys Pulse dashboard that works best for your business needs. You can assign a different display name or nickname to Genesys Pulse objects, such as queues, DN groups, agents, or VAGs.

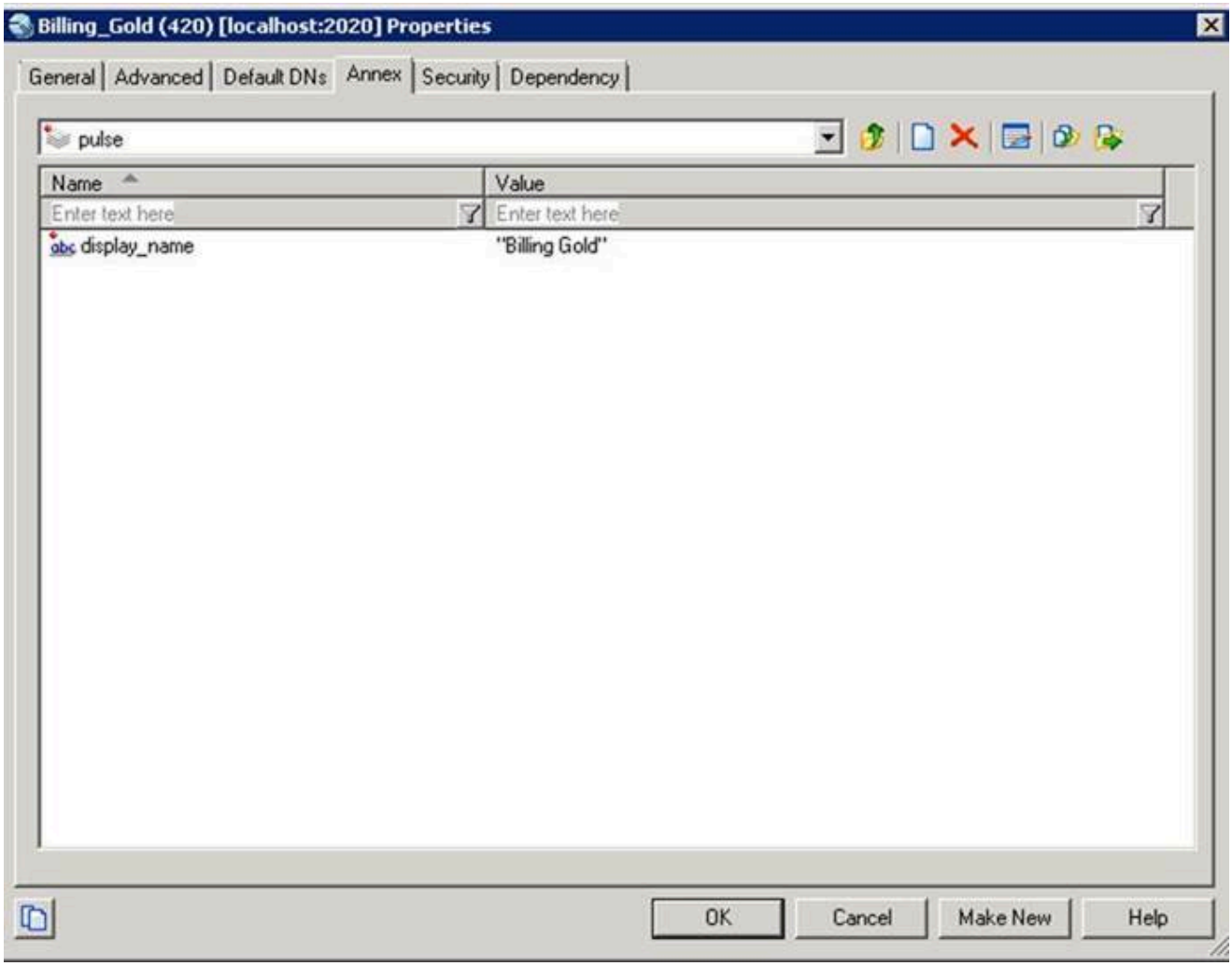
Change the default option



In order to use a custom name, you need to set the value of the specific object type option to `%CustomName%` in the Genesys Pulse Collector [object-name-format] section.

In this example, we set up a custom name for the system Virtual Queues by setting the `VirtACDQueue` option to `%CustomName%`.

Set the display name



Once you have set the object option to `%CustomName%`, you need to add the custom name in the Annex properties tab of the relevant objects.

In this example, instead of displaying the default value `Billing_Gold` (as shown in the header bar), Genesys Pulse displays `Billing  Gold` in the widgets. If you do not use the `display_name` option in the [pulse] section, the default value of the object is displayed (`Billing_Gold`).

Restart Genesys Pulse Collector

You must restart Genesys Pulse Collector to apply the changes.

Cleanup Snapshot Data

Genesys Pulse Collector can generate large amounts of layout data snapshot files each day. These files are usually useful for only a short time period.

> **Important**
>
> For the troubleshooting purposes Genesys strongly recommends to keep Genesys Pulse Collector outputs for at least recent 2-3 days.

Genesys provides a filesystem cleanup utility (cleanuptool), to simplify the removal of older files in data intensive environments. The cleanuptool is included within the Genesys Pulse Collector installation for all supported operating systems.

> **Important**
>
> Beginning with Genesys Pulse Collector release 9.0.007.00, the installation package for Linux provides three executable variants, each of which uses a distinct memory allocator:
>
> - collector.default - uses standard glibc memory allocator
> - collector.jemalloc - uses jemalloc memory allocator
> - collector.tcmalloc - uses tcmalloc memory allocator
>
> The cleanuptool executable is built only for the standard glibc memory allocator.

Deployment

After you install Genesys Pulse Collector, you can find the executable file — called **cleanuptool** in Linux or **cleanuptool.exe** in Windows — in the same folder that contains the Genesys Pulse Collector executable. You control this console application using a configuration file and command-line options.

Genesys recommends you run the cleanuptool regularly, using standard task scheduling software available within your operating system, such as Cron for Linux or Windows Task Scheduler for Windows. You must run the cleanuptool under a user account that has permissions to read, write, and remove files and directories on the filesystem where Genesys Pulse Collector writes the layout data snapshots.

Define your scheduling period based on data generation intensity, and your target file system capacity. Genesys recommends that you run the cleanuptool every hour.

How the cleanuptool works

The cleanuptool scans for files in the specified folder path (and subfolders). It checks to see how much time has elapsed between the time when the cleanuptool started and the last time each file was modified, compares the result to values you specify in the configuration file, and preserves or removes files accordingly.

The cleanuptool also gives you the option to use *granulation points* to determine whether to preserve a file. If you do not define granulation, the tool preserves all matching files that fall into one of the active intervals. Use granulation points if you want to preserve files created at specific points within the interval. For example, you might preserve files that were generated every *N* minutes; this greatly reduces the number of files stored, while retaining a representative sample that can still be useful for troubleshooting. More information and examples are given below.

Command-line options

Execute the cleanuptool as follows:

```
cleanuptool [options] <path> ...
```

Where **[options]** can be any of the values listed in the following table:

Command-line options

Option	Extended usage	Definition
-c	--config-file <file-name>	Specify new configuration file to use (default is './cleanuptool.ini', if present)
-p	--preserve-last-file	Always preserve last file, even outdated
-l	--follow-symlinks	Follow symbolic links
-m	--cross-mountpoints	Cross filesystem mount point boundaries
-s	--stop-on-error	Stop processing if error occurs
-d	--dry-run	Do not perform actual file removals
-V	--verbose	Show verbose messages
-VV	--extra-verbose	Show extra verbose messages
-q	--quiet	Do not show verbose messages
-v	--version	Show version
-h	--help	Show help message

Configuration File

The configuration file defines time intervals at which the cleanuptool preserves files. If the modification timestamp of a file matches one of the active time intervals, the file is preserved. Otherwise, it is deleted.

You can test the configuration file by running the cleanuptool in the **dry-run** mode with the **--extra-verbose** log option, which causes it to simulate a cleanup. In this mode, the tool collects and outputs information about the files that qualify to be deleted (but does not actually delete them) and displays output similar to the following:

```
Directory 'output/63/5a78424ef873-8ef1-11ec-c242-586644a1': 86 files to preserve, 6 files to
remove.
```

The configuration file has the standard INI-file format: It is divided into sections, and allows comment lines that start with semicolon. A sample configuration file is provided in the Genesys Pulse Collector installation directory, and is called **cleanuptool.ini.sample**.

General Section

The **general** section has two parameters:

- **active-intervals** — This required parameter lists the names of active intervals. There is no default value.
- **measure** — This optional parameter defines the time measurement units for interval points. Valid values: m, min, minutes—minutes, s, sec, seconds—seconds. The default value is minutes.

Intervals Section

The **intervals** section defines a set of intervals to use when evaluating whether to preserve a file. You can define multiple intervals, and then specify which intervals are used:

- Populate the **active-intervals** option in the **general** section with the names of the intervals to use.
- Define each interval as a separate INI file parameter in the **Intervals** section, using the format NAME= VALUE, where VALUE is a parameter definition string with the following format:

  ```
  [ALIGNMENT]<BEGIN>-<END>[:<GRANULARITY]
  ```

  Where:

  - **[ALIGNMENT]**— This optional parameter defines the alignment of granulation points. By default, the intervals are aligned to granularity number. Possible values: A = aligned, or U = unaligned.
    - **<BEGIN>** — This required parameter defines the beginning of the interval, inclusive.
    - **<END>** — This required parameter defines the end of the interval, not inclusive.
  - **[GRANULARITY]** — This optional parameter defines the granularity of the interval. A value of 0 (the default) means there are no granulation points. Specify a positive integer to set the spacing at which granulation points appear within the interval.

Interval examples

Interval Definition	Result
NAME=0-1	Preserve all files that were modified between 0 to 1 units before the current time.
NAME=10-20:1	From the interval between 10 and 20 units before the current time, preserve files near granulation points located at each 1 unit of time, aligned to granularity. (Alignment is not specified, so the default is used: the intervals are aligned to the granularity value). The resulting granulation points are: 10, 11, 12, 13, 14, 15, 16, 17, 18, and 19.
NAME=U10-20:3	From the interval between 10 and 20 units before the current time, preserve files near granulation points located at every 3rd unit of time. The first granulation point *is not* aligned to granularity value, so the first point is at the beginning of this interval. The resulting granulation points are: 10, 13, 16, and 19.
NAME=A20-60:7	From the interval between 10 and 60 units before the current time, preserve files near granulation points located at every 7th unit of time. The first granulation point *is* set to be aligned to the granularity value, so the first granulation point appears at the first time within the interval that is divisible by the granularity value. The resulting granulation points are: 21, 28, 35, 42, 49, and 56.

Capture Genesys Pulse Collector memory dumps

This information is useful when you need to:

- Take a memory dump from a running Genesys Pulse Collector process.
- Configure an operating system for automatically generating crash dumps when Genesys Pulse Collector crashes.

Take a Memory Dump of the Running Genesys Pulse Collector Process on Linux

1. Open a Linux terminal.
2. Confirm the **GCore** utility is installed from the gdb package by typing **gcore** in the terminal. If the **GCore** utility is not available, install it:
   a. Ubuntu: **sudo apt-get install gdb**
   b. RHEL, CentOS: **sudo yum install gdb**
3. Determine the process ID of Genesys Pulse Collector using the following command: **ps -ef | grep collector**
4. Change directories to the one to store the dump (for example: **cd ~/memory_dumps**).
5. Run commands: **gcore <PID>** where <PID> is process ID
6. You should get the file **core.<PID>**.
7. If you need to submit this core dump file to Genesys:
   a. Compress it (XZ or BZip2 are strongly recommended, as long as they give better compression ratio):
      i. with XZ: **xz -6 core.<PID>**
      ii. or with BZip2: **bzip2 -9 core.<PID>**
      iii. or with GZip: **gzip -9 core.<PID>**
   b. Submit the file **core.<PID>.xz** (or **core.<PID>.bz2** or **core.<PID>.gz**) to the location specified by the Genesys Customer Care.

Take a Memory Dump of the Running Genesys Pulse Collector Process on Windows

1. Make sure you have **ProcDump** utility. If not, complete following steps:
   a. Download the freeware **SysInternals ProcDump** utility from https://technet.microsoft.com/en-us/sysinternals/dd996900.aspx.
   b. Extract **procdump.exe** from the downloaded archive to ***C:\Windows***
2. Open Windows Task Manager (by pressing **Ctrl+Shift+Esc**, or pressing **Win+R** and typing **taskmgr** in the **Run** > dialog.)
3. In the Windows Task Manager, make sure you have a column PID. If not:
   a. Choose the menu item **View->Select Columns...**
   b. Select **PID (process identifier)**
4. In the Windows Task Manager, click **Show processes from all users**
5. In the Windows Task manager, sort processes by process name and find appropriate **collector.exe** and note its PID
6. Open the command prompt
7. Change directory to where you store the dump (for example, **cd /D D:\MemoryDumps**).
8. Type the following command: ***procdump -ma -o <PID> collector.<PID>.dmp*** where <PID> is the process ID of Genesys Pulse Collector
9. The memory dump file is created: **collector.<PID>.dmp**.
10. If you need to submit this memory dump file to Genesys:
    a. Install freeware **7-Zip** archiver of you don't already have it. You may download from this web site www.7-zip.org
    b. Open folder where memory dump resides in the Windows Explorer.
    c. Right-click on the dump file and choose menu item ***7-Zip->Add to archive...***
    d. Adjust following parameters in the **Add to Archive** dialog:
       i. Archive format: 7z
       ii. Compression level: Ultra
       iii. Compression method: LZMA
    e. Press OK and wait for compression to finish.
    f. Submit resulting file ***collector.<PID>.7z*** to the location specified by the Genesys Technical Support.

Tune a Linux Operating System to Generate a Core Dump for the Genesys Pulse Collector in the Case it has Crashed

1. As superuser, edit the file ***/etc/abrt/abrt.conf*** , set parameter **MaxCrashReportsSize** to value **0**.
2. As superuser, edit file ***/etc/abrt/abrt-action-save-package-data.conf*** , set parameter **ProcessUnpackaged** to value **yes**.
3. Restart abrtd service: ***sudo service abrtd restart***
4. Now Genesys Pulse Collector crash dumps will appear in the folder ***/var/spool/abrt*** or other folder, as configured in the ***/etc/abrt/abrt.conf*** .
5. If you need to submit this core dump file to Genesys:
   a. Locate directory with necessary crash data (named like ***ccpp-YYYY-MM-DD-HH:MM:SS-PID***) in the folder ***/var/spool/abrt*** (or other directory, as configured in the ***/etc/abrt/abrt.conf***). You are interested to provide Genesys with archive of the file called ***coredump*** located in that directory.
   b. Compress it (XZ or BZip2 are strongly recommended, as long as they give better compression ratio) - note that you must do that as superuser, because abrtd dump directory is typically not accessible to normal users:
      i. with XZ: **sudo xz -c -6 /var/spool/abrt/ *ccpp-YYYY-MM-DD-HH:MM:SS-PID*/ coredump >coredump.<PID>.xz**
      ii. or with BZip2: ***sudo bzip2 -c -9 /var/spool/abrt/ ccpp-YYYY-MM-DD-HH:MM:SS-PID/ coredump >coredump.<PID>.bz2***
      iii. or with GZip: ***sudo gzip -c -9 /var/spool/abrt/ ccpp-YYYY-MM-DD-HH:MM:SS-PID/ coredump >coredump.<PID>.gz***
   c. Change ownership of the resulting archive file so that you can access it with your regular user: ***sudo chown your-user-name:your-default-group coredump.<PID>.<Archiever-Specific-Suffix>***
   d. Submit resulting file ***coredump.<PID>.xz*** (or ***coredump.<PID>.bz2*** or ***coredump.<PID>.gz***) to the location specified by the Genesys Technical Support.

Tune Windows Operating System to Generate a Crash Dump for the Genesys Pulse Collector in the Case it has Crashed

1. Open Notepad text editor
2. Enter there below text, replacing value of the ***DumpFolder*** parameter with real path where you want to store Genesys Pulse Collector crash dumps. **IMPORTANT NOTES:** Directory path must be quoted and each directory separator (backslash) should be placed twice. Example: **"C:\\CrashDumps\\PulseCollector"**.

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows Error Reporting]\
"Disabled"=dword:0

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Windows Error Reporting\
LocalDumps\collector.exe]
"DumpFolder"="<path to folder where you store Genesys Pulse Collector dumps>"
"DumpCount"=dword:10
"DumpType"=dword:2
```

3. Save this file as ***collector-wer.reg*** .
4. Double click on it in the Windows Explorer and say yes to question of Windows Registry Editor to add data to Registry.
5. Now, if Genesys Pulse Collector crashes, full memory dump of Genesys Pulse Collector will appear in the specified folder.
6. If you need to submit this memory dump file to Genesys:
    1. Install freeware **7-Zip** archiver of you don't already have it. You may download from this web site www.7-zip.org
    2. Open folder where memory dump resides in the Windows Explorer.
    3. Right-click on the dump file and choose menu item ***7-Zip->Add to archive...***
    4. Adjust following parameters in the **Add to Archive** dialog:
        1. Archive format: 7z
        2. Compression level: Ultra
        3. Compression method: LZMA
    5. Click **OK** and wait for compression to finish.
    6. Submit resulting 7-Zip archive file to the location specified by the Genesys Customer Care.

Suggested Additional Reading

1. WER Settings (Windows) https://msdn.microsoft.com/en-us/library/windows/desktop/bb513638(v=vs.85).aspx
2. core - coredump file http://linux.die.net/man/5/core
3. limits.conf - configuration file for the pam_limits module http://linux.die.net/man/5/limits.conf
4. bash ulimit command in the bash man page http://linux.die.net/man/1/bash

Advanced Alerting Capabilities

Genesys Pulse provides the Advanced Alert capabilities. You can configure advanced alerts that depend on several statistical values and receive notifications via email. To enable this feature follow the steps below:

1. **Enable Embedded to Genesys Pulse microservices.**
   Set the following Genesys Pulse configuration options to `true`.
   - **[layout-watcher]/enabled**
   - **[widget-watcher]/enabled**
   - **[measure-watcher]/enabled**
   - **[pulse-user-permissions]/enabled**

   See Embedded Microservices for additional configuration options.
2. **Configure StatServer Data Provider service.**
   In the Genesys Pulse Collector installation folder find the `/microservices/StatServerDataProvider/StatServerDataProvider.cfg` file and specify the following options
   - **[StatServerDataProvider]**
     host = <StatServer host>
     port = <StatServer port>
   - Make sure that **[LayoutSubscriber]\target** corresponds to the host and port of the Embedded to Genesys Pulse `LayoutWatcher` service.

   See StatServer Data Provider for additional configuration options.
3. **Configure Formula Processor service.**
   In the Genesys Pulse Collector installation folder find the `/microservices/FormulaProcessor/FormulaProcessor.cfg` file and make sure that [MeasureSubscriber]\target corresponds to the host and port of the Embedded to Pulse `MeasureWatcher` service.
   See Formula Processor for additional configuration options.
4. **Configure AlertProcessor service.**
   In the Genesys Pulse installation folder find the /alert-processor/alertprocessor.properties file and specify the following options
   - aeron.driver.directory= <directory used by the aeron driver>
   - email.smtp.host = <SMTP server host>
   - email.smtp.port = <SMTP server port>
   - Make sure that the value of the `service.widgetwatcher.servers` option corresponds to the host and port of the Embedded to Pulse `WidgetWatcher` service.
   - Make sure that the value of the `service.userpermissions.servers` option corresponds to the host and port of the Embedded to Pulse `UserPermissions` service.

   See AlertProcessor for additional configuration options.
5. **Set the value of the Genesys Pulse option [pulse]/enable_advanced_alerts to true**. After enabling this feature, the `Advanced Alerts` section is shown on the `Alerts` tab of Widget and Template

Wizards. There users can specify conditions for an alert and email address where to send this alert.
Note, that the [pulse]/enable_advanced_alerts option affects Genesys Pulse user interfaces only. The availability of the Advanced Alerts section in the interface does not guarantee that required services (microservices) are deployed nor that they function properly. Already configured alerts will not be removed after this option is set to `false`.

6. If `FormulaProcessor`, `StatServer Data Provider`, and `AlertProcessor` are installed on different hosts, make sure that:
   - The proper host name or IP address (instead of the localhost) is specified in the aeron control and aeron endpoint parameters for `Formula Processor` and `StatServer Data Provider`.
   - `FormulaProcessor` and `StatServer Data Provider` hosts are properly specified in the following `AlertProcessor` options:
     - **service.formulaprocessor.aeron.channel**
     - **service.statserverdataprovider.aeron.channel**

   You can specify custom aeron control and endpoint parameters, but make sure to update corresponding options in the dependent microservice configuration file. For more details see options descriptions of AlertProcessor, Formula Processor, and StatServer Data Provider.
7. Restart Genesys Pulse
8. **Start Aeron Media Driver** on the host where `Formula Processor`, `StatServer Data Provider`, and `AlertProcessor` are deployed.
   The `Aeron Media Driver` executable is provided within `FormulaProcessor` and `StatServer Data Provider` installation folders. The `Aeron Media Driver` executable will start the aeron driver in a system-specific directory.
   Default driver_directory options values of `FormulaProcessor` and `StatServer Data Provider` are set to the same system-specific directory.
   Make sure that `FormulaProcessor`, `StatServer Data Provider` and `AlertProcessor` are configured to use the same aeron driver, when they are installed on the same host (the same directory is specified in the corresponding configuration options). `AlertProcessor` will start its own embedded `Aeron Media Driver` automatically if the external driver, which is configured to use the same directory, is not running.

   > **Tip**
   > **Quick Widget Updates and Advanced Alerting services require independent instances of `Aeron Media Driver`. Make sure that values of the Genesys Pulse Collector options driver-directory and endpoints do not interfere with settings of `Aeron Media Driver` used by Advanced Alerting services.**
9. **Start StatServer Data Provider.**
10. **Start Formula Processor.**
11. **Start AlertProcessor.**

Embedded to Genesys Pulse Microservices

These microservices are responsible for monitoring Advanced Alerts, configured in Genesys Pulse, and sending the information to Formula Processor, StatServer Data Provider, and AlertProcessor microservices for further processing.

The following microservices have to be started in the embedded mode:

- LayoutWatcher
- WidgetWatcher
- MeasureWatcher
- UserPermissions

In order to enable a particular service, the corresponding configuration options must be configured in the Genesys Pulse Application object. Changes to these options take effect after the Genesys Pulse restart.

For embedded LayoutWatcher the options are:

- **[layout-watcher]/enabled** - to enable embedded version in Genesys Pulse. The default value is **false**.
- **[layout-watcher]/port** - to specify gRPC port. The default value is **50051**.
- **[layout-watcher]/db_polling_interval** - to specify interval in ms for polling database. The default value is **5000**.

For embedded WidgetWatcher the options are:

- **[widget-watcher]/enabled** - to enable embedded version in Genesys Pulse. The default value **false**.
- **[widget-watcher]/port** - to specify gRPC port. The default value is **50052**.
- **[widget-watcher]/db_polling_interval** - to specify interval in ms for polling database. The default value is **5000**.

For embedded MeasureWatcher the options are:

- **[measure-watcher]/enabled** - to enable embedded version in Genesys Pulse. The default value **false**.
- **[measure-watcher]/port** - to specify gRPC port. The default value is **50053**.
- **[measure-watcher]/db_polling_interval** - to specify interval in ms for polling database. The default value is **5000**.

For embedded UserPermissions the options are:

- **[pulse-user-permissions]/enabled** - to enable embedded version in Genesys Pulse. The default value **false**.
- **[pulse-user-permissions]/port** - to specify gRPC port. The default value is **50054**.
- **[pulse-user-permissions]/cache_max_size** - to specify max size of cache. The default value is

10000.

- **[pulse-user-permissions]/cache_expire_timeout** - to specify expire timeout for cache in seconds. The default value is **1200**.

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.

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.

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.

Aeron Media Driver

Overview

The Aeron Media Driver is a separate process that provides buffers of data for Aeron to process from various transmission media. It decouples the means of data transmission from protocol processing.

Microservices require the driver to operate with Aeron. The Aeron Media Driver is not deployed as a separate package. It is included as a part of each microservice that has to work with the Aeron. The Aeron Media Driver executable is located inside the StatServer Data Provider directory (microservices/StatServerDataProvider/aeron-driver/bin/) or inside the Formula Processor directory (microservices/FormulaProcessor/aeron-driver/bin/). Despite the fact that the driver is a part of each service, only one driver instance is required for each host.

The Aeron Media Driver is implemented in Java and requires Java version 1.8.0 or newer.

How to Run Aeron Media Driver

To run the Aeron Media Driver as a foreground process, use the script provided with the driver. The script uses the default configuration for the driver.

[+] 9.0.001+ release

Stating with the Aeron Media Driver, included in Genesys Pulse Collector 9.0.001 release, you can provide your own configuration parameters via the AERON_DRIVER_OPTS environment variable in the *-Dparameter=value* form.

Below is the list of parameters supported by the Aeron Media Driver:

- **aeron.dir**
  The path to the directory where the Aeron Media Driver needs to store its files. On Linux, the directory inside the /dev/shm/ is recommended. If you provide your own path, make it the same for the driver and any microservice that operates with this driver. If it is not specified, then the default value provided by the Aeron Media Driver is used.
- **aeron.socket.so_sndbuf**
  aeron.socket.so_rcvbuf

  The size, in bytes, of the send and receive socket buffers. The length of the buffer must be a power of two.

  On Linux, it must not exceed the kernel configuration parameters:

  - net.core.wmem_max
  - net.core.rmem_max

- **aeron.term.buffer.length**
  aeron.publication.term.window.length

  The size, in bytes, of the Term (a section of data within a stream) buffer. The length of the buffer must be a power of two and must be the same length on both ends.

- **aeron.mtu.length**
  The length of MTU, in bytes.

For example, to specify the directory to store Aeron Media Driver files, set the AERON_DRIVER_OPTS environment variable to -Daeron.dir=path/to/directory.

[+] 9.0.000 release

You can provide your own configuration via environment variables:

- **AERON_DIR**
  The path to the directory where the Aeron Media Driver needs to store its files. On Linux, the directory inside /dev/shm/ is recommended. If you provide your own path, make it the same for the driver and any microservice that operates with this driver.

  If it is not specified then the default value provided by the Aeron is used.

- **AERON_SO_BUFFER**
  The size in bytes of the send and receive socket buffers. The length of the buffer must be a power of two. On Linux, it must not exceed the kernel configuration parameters:

  - net.core.wmem_max
  - net.core.rmem_max

  The default value is 4194304.

- **AERON_TERM_BUFFER**
  The size in bytes of the Term (a section of data within a stream) buffer. The length of the buffer must be a power of two and must be the same length on both ends.

  The default value is 67108864.

- **AERON_MTU**
  The length of MTU in bytes.

  The default value is 65504.

Run as a Service on Windows

To create a Windows service, perform the following steps:

1. Navigate to the aeron-driver installation directory, which contains the aeron_driver_service.ini and aeron_driver_service.exe files.
2. Edit the aeron_driver_service.ini service configuration file:
   - Replace the JVMPath value with the absolute path to the jvm.dll file in your host environment.
   - Replace the -Daeron.dir value with the absolute path to the folder for Aeron Media Driver files. If it

is not specified, the System-specific directory is created. For example, `C:\Windows\Temp\aeron-hostname`.

3. To start the service, run the following command in the Windows command prompt:

```
sc.exe create aeron-driver start=auto binPath="\"<path_to_aeron_driver_service.exe>\"
 -service aeron-driver -immediate"
```

   where *<path_to_aeron_driver_service.exe>* is the full path to the `aeron_driver_service.exe` file.

4. If needed, you can manage the service using the SC command in the Windows command prompt:

```
sc.exe start aeron-driver
 sc.exe stop aeron-driver
```

Run as a Service on Linux

Create a separate `systemd` service configuration file for the `Aeron Media Driver` service. For example, create `systemd` service configuration file `/etc/systemd/system/pulse-aeron-media-driver.service` with the following content:

9.0.001+ release:

```
[Unit]
Description=Pulse Aeron Media Driver
[Service]
ExecStart=/path/to/installation/aeron-driver/bin/aeron-driver
[Install]
WantedBy=multi-user.target
```

9.0.000 release:

```
[Unit]
Description=Pulse Aeron Media Driver
[Service]
ExecStart=/path/to/installation/run-aeron-driver
[Install]
WantedBy=multi-user.target
```

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