Genesys Pulse Deployment Guide

Genesys Pulse 8.5.103

12/29/2021

Table of Contents

Genesys Pulse Deployment Guide

Genesys Pulse is a Genesys Administrator Extension (GAX) plug-in application that enables at-a-glance views of real-time contact center statistics within the GAX 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.

> **Important**
>
> This Deployment Guide provides instructions for a new installation of Genesys Pulse. To migrate from an earlier version of Genesys Pulse, start with the *Genesys Pulse Deployment Procedure*.

About Genesys Pulse

Find overview information about architecture, components, and other concepts for Genesys Pulse.

Genesys Pulse Help

Architecture

Framework Components

Deployment

Plan your environment and deploy Genesys Pulse.

Migrate from an earlier release

Deploy Genesys Pulse and Genesys Pulse Collector

Configuration Options

Additional Options

Use

WebDAV Server to share snapshots

Restful Web Service API to display external data

Starting and Stopping

Start or stop the applications:

Genesys Pulse

Genesys Pulse Collector

Change History

Find information about what has changed in this release of the content.

Change History

Architecture and Components

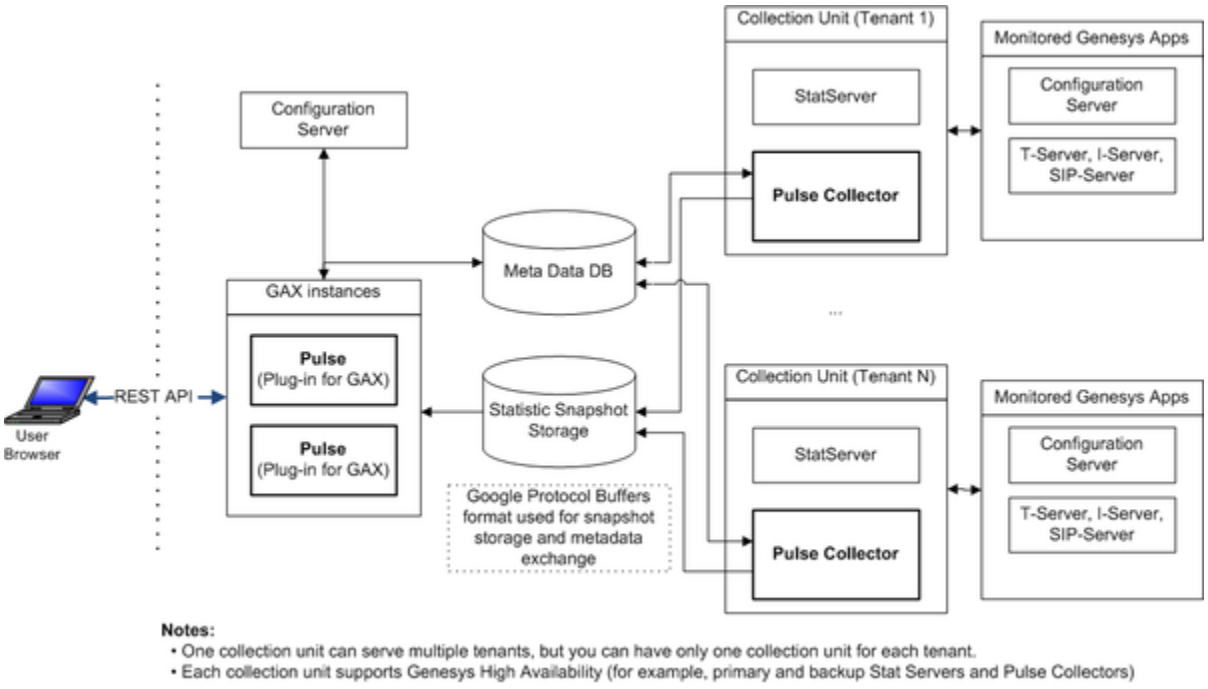
Genesys Pulse is a Genesys Administrator Extension (GAX) plug-in application that enables at-a-glance views of real-time contact center statistics within the GAX graphical user interface. Genesys Pulse uses widgets to display user-defined List, Donut, Key Performance Indicator (KPI), or Grid charts of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

You can have only one Genesys Pulse application for each GAX instance. For configuration, Genesys Pulse requires a `[pulse]` section and is added to the options of the GAX Application object. Genesys Pulse must have its own Database Access Point (DAP).

Genesys Pulse uses the existing GAX client architecture, which is described in the Genesys Administrator Extension Deployment Guide.

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.

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

When a Genesys Pulse Collector fails, GAX either cannot serve users and the load balancer routes users to the other GAX or, if you configured WebDAV, GAX 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.

GAX Instance

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

If the GAX instance fails while the supervisor is connected, the load balancer is still responsible for redirecting the user session to the other GAX instances and the login page is displayed. The second active GAX instance takes over and no statistics are lost. There are a pair of GAX 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 Guide.

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 e-mail 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.

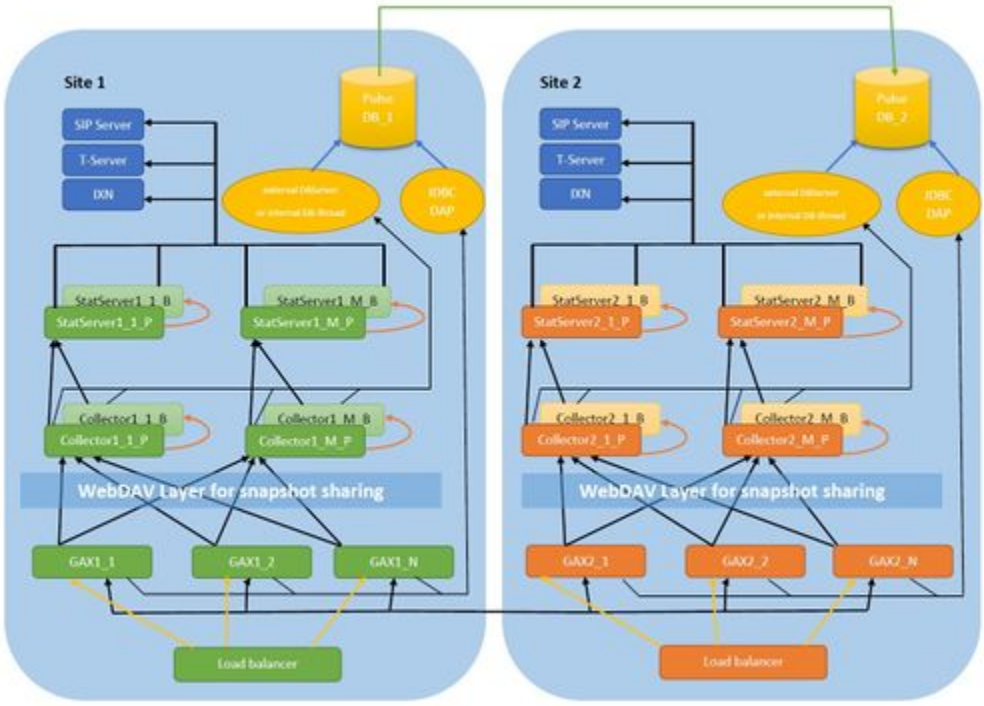
DB Server

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 the [Framework 8.1 DB Server User's Guide] 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 GAX 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

> **Important**
>
> - This Deployment Guide provides instructions for a new installation of Genesys Pulse. To migrate from an earlier version of Genesys Pulse, start with the Deployment Procedure.
> - You must deploy matching 5-digit releases of Genesys Pulse and Genesys Pulse Collector (for example, release 8.5.104.xx).
> - In these procedures, use the instructions provided in the GAX Deployment Guide or the Framework Deployment Guide, and add the object-specific configuration requirements listed here.

1. Confirm Software Requirements.

Confirm Software Requirements

You deploy Genesys Pulse as a GAX plug-in on a web application server to be accessed using a web browser through the GAX user interface.

The following are prerequisites for Genesys Pulse deployment:

- GAX release 8.5.220.20 and higher.

  > **Important**
  >
  > Genesys Pulse release 8.5.108.04 supports GAX versions 8.5.250.17 - 8.5.260.16.

- Genesys DB Server must be release 8.1.300.05 or higher. Refer to the Framework 8.0 DB Server User's Guide for details.
- Genesys Deployment Agent (GDA) must be installed on the computer on which you install Genesys Pulse and Genesys Pulse Collector if you plan to install Genesys Pulse by using GAX. GDA is required to deploy solution definitions and IPs through GAX. See the GAX Deployment Guide for details.
- Your Relational Database Management System (RDBMS) must be up and running. This release of Genesys Pulse supports the following relational database platforms:
  - Microsoft SQL Server 2012, 2012 Cluster, 2014, 2016 Enterprise Edition
  - Oracle 11g, 12c, 12c RAC

    - PostgreSQL 9.0
- The computer on which you install Genesys Pulse must be running one of the following:
    - Windows Server 2012, 2016.
      You must also install Microsoft Visual C++ Redistributable Package for Visual Studio 2013. See https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads for more details.
    - Red Hat Enterprise Linux AS 6.x (64-bit), 7.x (64-bit) with Updates from RHN enabled.
      Your System Administrator may need to install a set of compatibility packs for the 32-bit platform. These packages have generic names, for example: `compat-glibc*`, `compat-libstdc++*`, `libstdc++*`. Installer provides the information about missing packages (if any).

> **Important**
>
> Both Genesys Pulse Collector and GAX 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 8.5.103 or higher installed for basic operation.
- You must install Java 1.8 or higher.

> **Important**
>
> You cannot use compatibility mode in Internet Explorer, even if you are using a supported version.

2. Prepare the Genesys Pulse Database.

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 statement below for theGenesys 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 this statement:
   ```
   ALTER DATABASE <Pulse DB> SET READ_COMMITTED_SNAPSHOT ON;
   ```

Oracle

1. 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.

3. Configure a third-party data source.

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's 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.

4. Deploy Genesys Pulse Collector.

Deploy Genesys Pulse Collector

Genesys Pulse Collector connects directly to Stat Server to collect statistics for contact center objects. 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. Click step 11 from the tabs on the left to see details about how to **Configure Pulse Collector with an embedded DB Server**.

1. Upload Genesys Pulse Collector Installation Package and Template:
   a. In GAX, go to **Administration > Installation Packages** and click on the plus sign.
   b. Select **Installation Package Upload (template uploaded separately)** and click **Next**.
   c. 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).
   d. For **Upload an XML template**, select the XML Template file (Collector.xml from the Templates Installation Package directory).
   e. For **Upload an APD template**, select the APD Template file (Collector.apd from Templates Installation Package directory).
   f. Click **Finish**.
2. Deploy the Genesys Pulse Collector Installation Package and Template:
   a. Click on the Genesys Pulse Collector Installation Package to open the Properties tab.

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

   b. Click on the **Related** icon (configuration icon) and choose **Install** to open the IP Deployment Wizard.

c. Fill required fields and finish the installation.

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

> **Tip**
>
> You can also install Genesys Pulse Collector directly on the server by executing the Genesys Pulse Collector installation procedure from the Genesys Pulse Collector installation package, then uploading the Genesys Pulse Collector template (Collector.apd) using Configuration Manager.

3. 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.
4. For a high-availability (HA) deployment, repeat step 2 to install the backup instance to another server. You need a Genesys Pulse Collector for each instance of the Genesys Pulse application. Configure another Collector Application object for each Primary and set it as `Backup Server` in the General Tab in the options of the Primary 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 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 Collectors. If you have only two Collectors, then all widgets are processed by both Collectors.

5. In GAX, add the Genesys Pulse Collector `Application` object to the connections of your GAX `Application` object.
6. For a load-balanced environment configuration with two GAX applications and Genesys Pulse plug-ins, associate all instances of GAX with all primary instances of Genesys Pulse Collector.
7. 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.

a. Select the Default connection type.

b. 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.

c. Specify the connectivity parameters for your RDBMS.

d. To use an external DB Server, select the **Ports** tab to change the default port value to the value of your DB Server port.

8. In the connections of the Genesys Pulse Collector `Application`, add the DAP that is to be used by Genesys Pulse Collector.
9. In the connections of the Genesys Pulse Collector `Application`, add the primary Stat Server application that is to be used by Genesys Pulse Collector.
10. Add the Tenant objects to the Tenants tab for all Genesys Pulse Collectors that you plan to monitor in Genesys Pulse.

5. Optional: Deploy Cluster Configuration.

Optional: Deploy Genesys Pulse Cluster Configuration

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

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

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

4. Review the cluster architecture and sizing information in the Sizing Guide.
5. Install GAX with Genesys Pulse on one node.
6. Set up two or more Genesys Pulse Collector instances. All instances of Collector can be installed on the same node with GAX or on any number of remote nodes.
   a. If one or more 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 Collectors are considered remote, because for the second Genesys Pulse node, they are remote.

   b. Configure options for Collector instances installed on a remote nodes and then restart every Collector instance:
      - Set the `external-output-url` option in the `transport-file` section to

        `http://<WebDAV host>/<path to folder with snapshots>`
      - Set the `external-heartbeat-url` option in the `heartbeat` section to

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

6. Deploy Genesys Pulse.

Deploy Genesys Pulse

Configure the necessary objects required by Genesys Pulse using GAX.

1. Upload Genesys Pulse Installation Package and Template:
   a. In GAX, go to **Administration > Installation Packages** and click on the plus sign.
   b. 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).
   c. For **Upload an XML template**, select the XML Template file (Pulse.xml from the Templates Installation Package directory).
   d. For **Upload an APD template**, select the APD Template file (Pulse.apd from Templates Installation Package directory).
   e. Click **Finish**.

2. Deploy the Genesys Pulse Installation Package and Template:
   a. Click on the Genesys Pulse Installation Package to open the **Properties** tab.

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

   b. Click on the related icon and choose **Install** to open the IP Deployment Wizard.
   c. Fill required fields and finish the installation.

> **Important**
>
> - For "IPCommon InstallPath", use an empty folder where the Genesys Pulse plugin will be installed (for example, C:\123).
> - When installing Genesys Pulse on Linux, a second path leading to the GAX installation (the field is called GAX directory) must be specified in addition to InstallPath.

> **Tip**
>
> You can also install Genesys Pulse directly on the server by executing the Genesys Pulse installation procedure from the Genesys Pulse installation package.
>
> If the installation procedure fails to find the GAX installation, you can manually copy the Genesys Pulse plugin files to enable Pulse in GAX.
>
> After installation, find all jar files in root directory of installed Genesys Pulse plugin and copy them to:
>
> Linux: `<GAX root>/plug-ins` (if the directory exists) and `<GAX root>/webapp/WEB-INF/lib`
>
> Windows: `<GAX root>\plug-ins` (if the directory exists) and `<GAX root>\webapp\WEB-INF\lib`.

> **Important**

> Starting with the 8.5.108 release, if Pulse is configured to use Aeron, make sure that only `aeron-client-1.4.0.jar`, `aeron-driver-1.4.0.jar`, `agrona-0.9.7.jar` files are presented in the plug-ins and lib directories:
>
> Linux: `<GAX root>/plug-ins` (if the directory exists) and `<GAX root>/webapp/WEB-INF/lib`
>
> Windows: `<GAX root>\plug-ins` (if the directory exists) and `<GAX root>\webapp\WEB-INF\lib`.
>
> If there are any previous versions of aeron-client, aeron-driver, or argona jar files, please remove them manually.

3. Navigate to **Configuration > Environment > Application**, select the GAX Application, **Permissions** tab and configure the SYSTEM account to have Read, Update, and Execute permissions.
4. Optional: Configure the [Pulse] options on **Application Options** tab for the GAX `Application` object:

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

5. Create a new Database Access Point, which is necessary for connectivity to the Genesys Pulse database:
   a. Enter a **Database Name**.
   b. 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
   c. On the **Ports** tab, change the value of the default port to the value of your RDBMS port.
   d. On the **Application Options** tab, add the **role** configuration option with its value set to **pulse** in the **GAX** section to identify this Database Access Point as the database schema created for Genesys Pulse.
   e. For a PostgreSQL database, add the postgre_71_compatible configuration option with its value set to false in the GAX section on the **Application Options** tab.

Refer to the Database Access Points for additional information.

6. Add the Database Access Point to the connections of your GAX `Application` object.
7. For a High Availability (HA) deployment,
   a. Complete Steps 2-6, for each GAX `Application` object to be used with the Genesys Pulse plug-in.
   b. Each GAX application should have the other GAX application in its connections and there must be no relation between the GAX applications as Backup Servers.

> **Important**
> Both 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. Restart GAX. See the GAX Deployment Guide for information about how to start GAX.

7. Optional: Deploy RabbitMQ.

Optional: Deploy 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 StatServer and Genesys Pulse.
>
> You are responsible to validate that your environment can handle the load in production caused by quick updates.

Use RabbitMQ for quick widget Updates.

1. To use RabbitMQ to work with Genesys Pulse, use the following software versions:
   - RabbitMQ server version 3.3.5.
   - Use identical versions of Erlang on all hosts running RabbitMQ:

- Windows, use Erlang version OTP 17.3
- Linux, use the latest Erlang version available (R14B-04 or newer)

> **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. If you are using RabbitMQ 3.7.5 or newer, please make sure the RabbitMQ server's option channel_max is set to `0`.

2. Determine whether to use RabbitMQ with Genesys Pulse using Cluster or Single-Node configurations. Genesys Pulse supports using these configurations only for use with RabbitMQ.
   - **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`).
   - **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.
3. On every host, install Erlang:
   - On Windows Server:
     i. Download Erlang OTP 17.3 Windows installer. For Windows x64, use the 64-bit version.
     ii. Run installer.
     iii. Go through the installation wizard and install Erlang.
   - On Linux, as the **root** user, run the following commands:
     i. To enable EPEL:

     ```
     wget http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm

     rpm -ivh epel-release-6-8.noarch.rpm
     ```

     ii. To install Erlang:

     ```
     yum install erlang
     ```
4. On every host, install and configure RabbitMQ:
   a. Download and install RabbitMQ:
      - On Windows Server:
        i. Download RabbitMQ installer
        ii. Run the installer.

iii. Go through the installation wizard and install RabbitMQ.

- On Linux, as the **root** user, issue the following commands:

i. Download RabbitMQ package:

```
wget https://www.rabbitmq.com/releases/rabbitmq-server/v3.3.5/rabbitmq-server-3.3.5-1.noarch.rpm
```

ii. Import RabbitMQ key:

```
rpm --import http://www.rabbitmq.com/rabbitmq-signing-key-public.asc
```

iii. Install RabbitMQ server:

```
yum install rabbitmq-server-3.3.5-1.noarch.rpm
```

b. On every host, first start RabbitMQ server to generate Erlang cookie file.

- On Windows, start RabbitMQ service. If it is already started proceed to the next step.
- On Linux, from the command line as the **root** user:

i. Run: `service rabbitmq-server start`

ii. If you want RabbitMQ to start automatically when system boots, run:

```
chkconfig rabbitmq-server on
```

> **Important**
>
> To run the **rabbitmqctl** tool that is installed as a part of RabbitMQ server.
>
> - On Linux, run it as **root** right after installing the server.
> - On Windows, navigate to `${RabbitMQ installation directory}\rabbitmq_server-3.3.5\sbin` directory and run **rabbitmqctl** from there.

> **Important**
>
> When the entire cluster is brought down, the last node to go down must be the first node to be brought online. If this doesn't happen, the nodes will wait 30 seconds for the last disc node to come back online, and fail afterwards.
> -rabbitmq.com
>
> When node fails due to the above limitation RabbitMQ's `startup_log` contains message like this:
>
> ```
> BOOT FAILED
> ===========
> Error description:
>    {could_not_start,rabbit,
> ```

```
{bad_return,
    {{rabbit,start,[normal,[]]},
     {'EXIT',
         {rabbit,failure_during_boot,
             {error,
                 {timeout_waiting_for_tables,
```

c. (Cluster configuration only) Reset RabbitMQ on every RabbitMQ host using the following commands:

```
rabbitmqctl stop_app

rabbitmqctl reset
```

d. (Cluster configuration only) Stop RabbitMQ server on every host. You must stop RabbitMQ on both servers before starting any one of them. On both hosts, do the following:

- On Windows, stop RabbitMQ service.
- On Linux, as the **root** user, run:

```
service rabbitmq-server stop
```

e. (Cluster configuration only) Copy Erlang cookie from one host to all others. These files must be identical on all hosts that form RabbitMQ cluster.

- On Windows: Copy to both `C:\Users\<user account under which Erlang runs>\.erlang.cookie` and `C:\Windows\.erlang.cookie`
- On Linux: Copy to `/var/lib/rabbitmq/.erlang.cookie`

f. (Cluster configuration only) Configure RabbitMQ cluster consisting of several RabbtMQ instances running on `host1, host2, ..., hostn`.

i. Create the `rabbitmq.config` file on all of those hosts:

- On Windows, in `%APPDATA%\RabbitMQ\`.
- On Linux, in `/etc/rabbitmq`.

This how configuration might look for a 2-node cluster:

```
[
 {rabbit,
  [
    {cluster_nodes, {['rabbit@host1', 'rabbit@host2'], disc}}
  ]
  }
].
```

Change `host1` and `host2` with the names of your hosts.

> **Important**
>
> Hostnames are case sensitive. You must use uppercase

> characters for Windows hosts and lowercase characters for Linux hosts.

ii. On Windows only: In `${RabbitMQ installation directory}\rabbitmq_server-3.3.5\sbin` run the following command: `rabbitmq-service.bat install`

g. (Cluster configuration only) Start RabbitMQ server on all hosts.

- On Windows, start RabbitMQ service.
- On Linux, as the **root** user run:

```
service rabbitmq-server start
```

h. (Cluster configuration only) Confirm that RabbitMQ instances have formed a cluster. Run:

```
rabbitmqctl cluster_status
```

The output should contain the following line for the cluster consisting of two nodes:

```
running_nodes,['rabbit@host1','rabbit@host2']}
```

i. 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
```

j. 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
```

k. 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.*" ".*" ".*"
```

5. 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. Add or update configuration options in the [`transport-rabbitmq`] section.
6. Restart Genesys Pulse Collector and GAX 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.

8. Configure the Stat Server Application object.

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 monitor 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`.

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

9. Configure user access.

Configure User Access

1. 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**.

a. Define the privileges granted by the Role on the **Assigned Privileges** tab in the Genesys Pulse section.
Privilege Details:

- **Pulse View Dashboard**—User has read-only access to launched dashboards without the ability to expand widgets to tab (includes stay on dashboard ability).

- **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 and close dashboards and expand widgets to tab.
- **Pulse Edit Widget Display**—User can modify widget display options.
- **Pulse Manage Widgets**—User can create, remove, or modify all widget options.
- **Pulse Manage Shared Dashboards**—User can create, remove, or modify shared dashboards.
- **Pulse Manage Templates**—User can create, remove, or modify templates.
- **Pulse Add Widgets Without Limit**—User can add any number of widgets.
- **Pulse Manage Users**—Manage other users' widgets and dashboards.
- **Pulse Manually Bind Collectors**—Manually select Pulse Collectors to process particular widgets or templates.

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.

Pulse privileges use GAX logic, so 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.

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

2. Provide appropriate permissions on the following objects:
   a. Read and Execute on GAX client application object (usually called `default`, controlled by the GAX option `client_app_name` in the [general] section).
      i. Select **Configuration**.
      ii. From the **Environment** pane, choose **Applications**.
      iii. Click to open the required application (usually called `default`) to view its properties.
      iv. Select **Permissions** tab.
      v. Add the **Person** or **Access** group containing this user.
      vi. Add required permissions: **Read** and **Execute** are required to log in to GAX.

b. Tenant Environment
   i. Select **Configuration**.
   ii. From the **Environment** pane, choose **Tenants**.
   iii. Click to open the required tenant to view its properties.
   iv. Select **Permissions** tab.
   v. Add the **Person** or **Access** group containing this user.
   vi. Add required permissions: **Read** and **Execute** are required to log in to GAX.
c. User's own tenant
   i. Select **Configuration**.
   ii. From the **Accounts** pane, choose **Tenants**.
   iii. Click to open the required tenant to view its properties.
   iv. Select **Permissions** tab.
   v. Add the **Person** or **Access** group containing this user.
   vi. Add required permissions: **Read** and **Execute** on tenant are required to log in to GAX.

10. Optional: Configure Multi-Language Environments.

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 the Windows platform:

```
-cs_codepage 65001
```

You may need to edit the Genesys Pulse Collector service command line in the Windows registry.

11. Optional: Configure embedded DB Server.

Optional: Configure Pulse Collector with an embedded DB Server

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

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

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

Software Requirements

Pulse Collector uses Genesys DB Clients, which require DBMS clients to be installed. Follow the instructions under the **DBMS Environment Settings** chapter of the DB Server User's Guide to install and configure those clients.

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

- You had the Genesys DB Server installed on the host with Pulse Collector.
- The Genesys DB Server and Pulse Collector are of the same architecture (32- or 64-bit).

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 a **[log-db]** section to **collector** application options and add all of the options that usually go to the **[log]** section of Genesys DB Server application.

If there is no **[log-db]** section specified in **collector** application options, the embedded DB Server log uses the 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 **[log]** section, with **-db** suffix added.

Limitations

- The Linux version of Pulse Collector running with embedded DB Server is unable to connect to MS SQL Server database.
- If you are using Oracle 12c database with Genesys Pulse Collector configured with embedded DB Server, make sure that you install Oracle 11.2 client on the host that runs Genesys Pulse Collector.

Deploy Language Packs

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

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

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

To use the installed language in Pulse, you must select the preferred language in GAX preferences menu. See the Genesys Administrator Extension Help for details.

Genesys Pulse Configuration 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

[pulse] Section

cache_expire_timeout
Valid Values: zero or any positive number
Default Value: 1200
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_max_active_connections
Valid Values: any positive number
Default Value: 8
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.

editable_templates
Valid Values: true,false
Default Value: 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_manual_collector_binding
Valid Values: true,false
Default Value: 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.

health_expire_timeout
Valid Values: zero or any positive number
Default Value: 30
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
Valid Values: true,false
Default Value: true
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_widgets_per_user
Valid Values: zero or any positive number
Default Value: 0
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
Valid Values: specially formed JSON string
Default Value: not specified
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
Valid Values: any string
Default Value: not specified
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
Valid Values: zero or any positive number
Default Value: 24
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.

Genesys Pulse Collector Application Object

[collector] Section

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

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

management-port
Valid Values: Positive integers
Default Value: No default value
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
Valid Values: any string

Default Value: not specified
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

check-layout-presence-timeout
Valid Values: 0-3600
Default Value: 900
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
Valid Values: 3-3600
Default Value: 30
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
Valid Values: 0...3600
Default value: 60
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
Valid Values: 0...3600
Default Value: 60
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
Valid Values: 0-86400

Default Value: 0
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
Valid Values: 10–3600
Default Value: 300
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
Valid Values: 0-600
Default Value: 60
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**
>
> - GAX 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.
> - GAX 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.
> - GAX 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.

external-heartbeat-folder
Valid Values: Network or locally mounted path to the heartbeat-folder accessible from a remote GAX instance.
Default Value: No value
Changes Take Effect: After GAX restart
Enables a remote GAX 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 `./`

external-heartbeat-url
Valid Values: valid HTTP URL to a heartbeat's folder on WebDAV server. For example, http://host1:8080/heartbeat
Default Value: No value
Changes Take Effect: After GAX restart
Description: Enables a remote GAX 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 GAX in case when Genesys Pulse Collector and GAX are configured on the same Host or external-output-folder option is specified.

heartbeat-folder
Valid Values: Valid folder path
Default Value: ./output/heartbeat (as provided by template file)
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
Valid Values: 3-3600
Default Value: 120
Changes Take Effect: After restart
Specifies the period of a heartbeat update, in seconds.

heartbeat-success-condition
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.

Default Value: statserver, snapshot-writer
Changes Take Effect: After restart
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
Valid Values: yes,no
Default Value: yes
Changes Take Effect: After restart
Enables or disables layout validation in Genesys Pulse Collector.
Note for releases prior to 8.5.103: Genesys Pulse Collector always checks for consistency of LayoutID and TenantID. These validations apply even if options are set to a value no.

validate-strict-tenant-security
Valid Values: yes,no
Default Value: yes
Changes Take Effect: After restart
Discontinued: as of 8.5.103
Enables Genesys Pulse Collector to fail the layout validation when the option strict-tenant-security in the configuration-monitoring section is set to yes and Genesys Pulse encounters an individual object that belongs to a tenant that is different than layout's tenant.

[limits] Section

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

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

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

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

[Log] Section

all
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.

Default Value: stdout
Changes Take Effect: Immediately
Specifies the outputs to which Data Sourcer 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.

segment
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.

Default Value: 10 MB
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
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.

Default Value: all
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 Deployment Guide or Framework Solution Control Interface Help 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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%

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

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

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

Workbin
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Valid object name format string. For details, see the table at the bottom of this section.
Default Value: %ObjectName%
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
Valid Values: Debug, Info, Warning, Error, Fatal, Unknown, None (case-insensitive)
Default Value: Warning
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
Valid Values: Debug, Info, Warning, Error, Fatal, Unknown, None (case-insensitive)
Default Value: Info
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
Valid Values: 1-86400
Default Value: 1800
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
Valid Values: 1-128
Default Value: 2
Changes Take Effect: After restart
Defines the number of concurrent threads used to build snapshots.

stat-data-processing-thread-max-load-factor
Valid Values: 1-300
Default value: 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
Valid Values: 0...256
Default Value: 4
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`.

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

[scripting] Section

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

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

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

js-lib-path
Valid Values: Valid folder paths
Default Value: ./jslib/standard
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
Valid Values: Comma-separated list of JavaScript files
Default Value: collector.js,cfglib.js,statlib.js,gts.js
Changes Take Effect: After restart

Comma-separated list of modules to preload into the scripting engine.

stop-compute-formula-threshold-for-snapshot
Valid Values: 0-100
Default Value: 3
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
Valid Values: yes, no
Default Value: 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.

data-source-choice-strategy
Valid Values: PrimaryRunMode, LastGood, FirstAvailable, MostUpToDate, PrimaryInCME, Local
Note: All values are case-insensitive.
Default Value: PrimaryRunMode
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.
- FirstAvailable—Genesys Pulse Collector uses the Primary Stat Server if available. If uses the Primary Stat Server is unavailable, Genesys Pulse Collector uses the Backup Stat Server. 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 always uses the Primary Stat Server.
- 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
Valid Values: 1-2147483647
Default Value: 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.

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

statserver-batch-size
Valid Values: 1-10000
Default Value: 500
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
Valid Values: 1-86400
Default Value: 600
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
Valid values: yes, no
Default value: yes
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
Valid Values: yes, no
Default Value: 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
Valid Values: true, false
Default Value: 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-file] Section

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

compression-method
Valid Values: None, LZ4 (case-insensitive)
Default Value: None
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
Valid Values: yes, no
Default Value: no
Changes Take Effect: After GAX restart
Specifies whether separate output of the latest full snapshot is enabled.

external-latest-snapshot-output-folder
Valid Values: Network or locally mounted path to the latest-snapshot-output-folder that is accessible from a remote GAX instance.
Default Value: No value
Changes Take Effect: After GAX restart
Enables a remote GAX 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 GAX in cases when both Genesys Pulse Collector and GAX are configured on the same host.

external-latest-snapshot-output-url
Valid Values: Valid HTTP URL to a latest-snapshot-output-folder through WebDAV server. For example, http://host1:8080/latest_snapshot
Default Value: No value
Changes Take Effect: After GAX restart
Enables a remote GAX 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 GAX in cases when both Genesys Pulse Collector and GAX are configured on the same host or the external-output-folder option is specified.

external-output-folder
Valid Values: Network or locally mounted path to the heartbeat-folder accessible from a remote GAX instance

Default Value: No value
Changes Take Effect: After GAX restart
Enables a remote GAX 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 `./`

external-output-url
Valid Values: valid HTTP URL to a snapshot's folder on WebDAV server. For example, http://host1:8080/output
Default Value: No value
Changes Take Effect: After GAX restart
Enables a remote GAX 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 GAX in case when Genesys Pulse Collector and GAX are configured on the same Host or external-output-folder option is specified.

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

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

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

output-file-mode
Valid Values: 0-0777
Default Value: 0664
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
Valid Values: Valid folder paths
Default Value: No default (the Collector.apd file supplies the value of a sample output directory)
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 Valid Values: 1-128
Default Value: 2
Changes Take Effect: After restart
Defines the number of concurrent threads used to write snapshots.

[transport-rabbitmq] Section

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

exchange
Valid Values: A non-empty sequence of these characters: letters, digits, hyphen, underscore, dot, or colon.
Default Value: <None>
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
Valid Values: 1-10000
Default Value: 1000
Changes Take Effect: After restart
Specifies the maximum number of messages allowed in the queue.

nodes
Valid Values: hostname1:port1, hostnameN:portN
You must specify the port value: 0-65535.
Default Value: localhost:5672
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
Valid Values: String
Default Value: pulse
Changes Take Effect: After restart
Specifies the password for the username.

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

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

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

[transport-webdav] Section

Starting with release 8.5.106, the following new configuration options are added:

output-url
Valid Values: Valid URL
Default Value: no default value
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
Valid Values: Valid URL
Default Value: no default value
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 **[transport-file]/external-output-url** and **[transport-file]/external-output-folder** options are ignored.

latest-snapshot-output-url
Valid Values: Valid URL
Default Value: no default value
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

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

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

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

WebDAV Server Configuration

Use WebDAV server for sharing snapshots between Genesys Pulse and Genesys Pulse Collector when they are installed on different hosts and there is no other way to share the folder that contains snapshots.

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 GAX 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:

- For Genesys Pulse version 8.5.106 and higher:
  - 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>**

- For Genesys Pulse version lower than 8.5.106:
    - Set the **external-output-url** option in the **[transport-file]** section to **http://<WebDAV host>/<path to snapshots folder>**
    - Set the **external-latest-snapshot-output-url** option in the **[transport-file]** section to **http://<WebDAV host>/<path to the latest snapshots folder>**
    - Set the **external-heartbeat-url** in the **[heartbeat]** section to **http://<WebDAV host>/<path to heartbeat folder>**

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 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 RHEL 6 or 7

Installation of lighttpd with WebDAV module

1. Enable EPEL: As **root** issue the following commands:

```
wget http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm rpm -ivh epel-release-6-8.noarch.rpm
```

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 GAX by setting the GAX 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 Restful Web Service API

Display external data in Genesys Pulse by using the Genesys Pulse Restful Web Service API.

> **Important**
> The API is subject to change at any time without notice.

/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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard

ability).

or

Pulse Read All Layouts—User can access all Pulse layouts as read-only.

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.

Available request representations:

- application/json with template configuration.

Required permission:

- Pulse Manage Templates—User can create, remove, or modify templates.

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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

  or

  Pulse Read All Layouts—User can access all Pulse layouts as read-only.

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 Manage Templates—User can create, remove, or modify templates.

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 Manage Templates—User can create, remove, or modify templates.

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 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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

  or

  Pulse Read All Layouts—User can access all Pulse layouts as read-only.

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.

Required permission:

- Pulse Manage Widgets—User can create, remove, or modify all options of widgets.

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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

  or

  Pulse Read All Layouts—User can access all Pulse layouts as read-only.

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.

The request does not create new layout if it does not exist

Available request representations:

- application/json with layout configuration.

Required permission:

- Pulse Manage Widgets—User can create, remove, or modify all options of widgets.

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 specififed id.

Required permission:

- Pulse Manage Widgets—User can create, remove, or modify all options of widgets.

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 doesn't 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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

  or

  Pulse Read All Layouts—User can access all Pulse layouts as read-only.

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.

Required permission:

- Pulse Write Snapshot—User can upload snapshots for layouts.

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.

name	required	type	description
			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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

  or

  Pulse Read All Layouts—User can access all Pulse layouts as read-only.

Available response representations:

- 200 - application/json with array of snapshots.
- 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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

Available response representations:

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

[+] POST

Create new widget for specified layout.

Available request representations:

- application/json with widget configuration.

Required permission:

- Pulse Edit Widget Display—User can modify display options of widgets.

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
- 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

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

Required permission:

- Pulse View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

Available response representations:

- 200 - application/json with widget configuration.
- 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

Available request representations:

- application/json with widget configuration.

Required permission:

- Pulse Edit Widget Display—User can modify display options of widgets.

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.
- 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 Edit Widget Display—User can modify display options of widgets.

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	description
brief	no	boolean	If specified and "true" then will be returned only set of tabs without widgets. Otherwise will return tabs with field "widget" containing full widget definition for all widget's presented on tab.
shared	no	boolean	If specified and "true" then only shared dashboards (tabs with type "ttDashboard") are returned. Otherwise user tabs are returned. User cannot change or delete

name	required	type	description
			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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

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	description
shared	no	boolean	If specified and "true" then will be created shared tab, which doesn't belong only to current user. Otherwise will be created tab which is available only for current user
brief	no	boolean	If specified and "true" then widgets will not be created according to widgets definitions (only positions will be saved). Otherwise will create new widgets according to widgets definitions. Each created widget continues to use same layout as original until it (layout) has changed. See description for PUT /api/wbrt/

name	required	type	description
			layouts.
overwrite	no	boolean	if true then tabs with same name and saved to same location will be removed (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 Manage Tabs—User can launch and close dashboard.
- Pulse Manage Shared Dashboards—User can create, remove, or modify shared dashboards.

Available response representations:

- 200 - Tab was successfully created. Redirect to newly created tab
- 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": "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.

[+] 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 will remove all personal tabs.

name	required	type	description
brief	no	boolean	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. Each created widget continues to use same layout as original until it (layout) has changed. See description for PUT /api/ wbrt/layouts.

Available request representations:

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

Required permission:

- Pulse Manage Tabs—User can launch and close dashboard.
- Pulse Manage Shared Dashboards—User can create, remove, or modify shared dashboards.

Available response representations:

- 200 - User's tabs were saved..
- 400 - Provided tabs configuration is not valid, application/json with details:

  - `{ "message": "JSON_PARSE_ERROR" }` - request has invalid format
- 403 - User does not have privileges.

/api/wbrt/tabs/<guid>

Methods:

[+] GET

Returns tab with specified guid.

Request query parameters:

name	required	type	description
brief	no	boolean	If specified and "true" then will be returned tab without widgets definition (only positions property filled). Otherwise will return tab with field "widget" containing full widget definitions for all widget's presented on tab.

Required permission:

- Pulse View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

Available response representations:

- 200 - application/json with tab.
- 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	description
brief	no	boolean	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	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 Manage Tabs—User can launch and close dashboard.
- Pulse Manage Shared Dashboards—User can create, remove, or modify shared dashboards.

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.
- 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 Manage Tabs—User can launch and close dashboard.

Available response representations:

- 200 - Tab was successfully deleted.
- 204 - if 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/import

Methods:

[+] POST

Import entities provided in request body.

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

```
{
    "data": <result of</ac:plain-text-body>
        </ac:structured-macro> 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 Manage Templates—User can create, remove, or modify templates.
- Pulse Manage Shared Dashboards—User can create, remove, or modify shared dashboards.

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 View Dashboard—User can access launched Pulse dashboard as read-only (includes stay on dashboard ability).

  or

  Pulse View Dashboard Restricted—User can access launched Pulse dashboard as read-only (does not include stay on dashboard ability).

Available response representations:

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

/api/plugins/wbrt/health

Methods:

[+] GET

Checks Pulse application health.

Required permission:

- no permissions required

Available response representations:

- 200 - Pulse application works fine
- 503 - Pulse health check failed, application/json with malfunction details:
  - `{ "message": "HEALTH_CHECK_FAILED" }` - no details available
  - `{ "message": "COLLECTOR_DOES_NOT_RESPOND" }` - collector stops updating heartbeat file
  - `{ "message": "DATABASE_ERROR" }` - no connection to database
  - `{ "message": "CFG_SERVER_ERROR" }` - no connection to configuration server

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 Pulse know that data from LayoutSnapshot is legal for current state of LayoutDefinition.

> ### Important
>
> Starting with **Pulse 8.5.104.05** 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 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.

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 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"
    }
}
```

Starting and Stopping Genesys Pulse

Genesys Pulse starts and stops automatically when Genesys Administrator Extension (GAX) starts and stops.

Prerequisites

Confirm the following requirements before starting Pulse:

a. See Browser Support for Pulse in the *Genesys Supported Operating Environment Reference Guide*.
b. Your monitor must be set to a resolution of no less than 1024x768.
c. The DB Server for Genesys Pulse Collector must be running.
d. The RDBMS for the Genesys Pulse database must be running.
e. 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.

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 pulse-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.

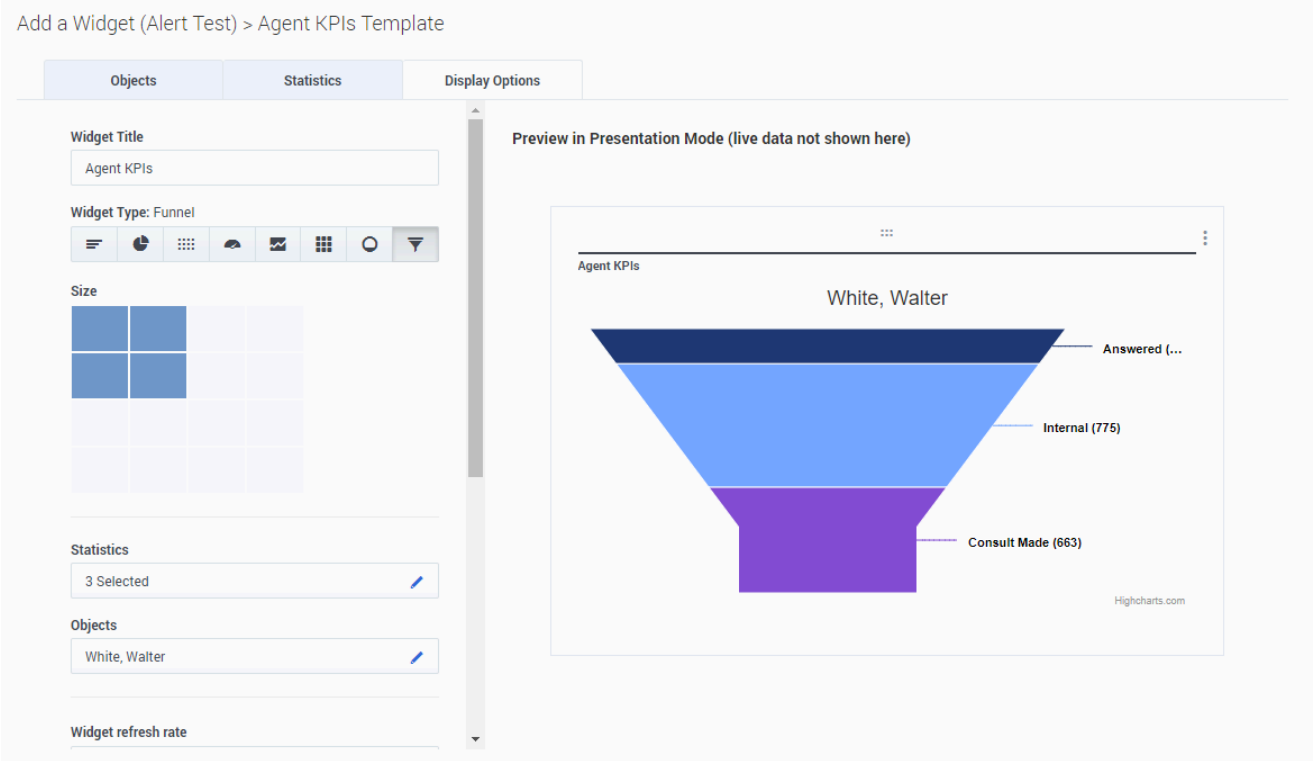
After that you are ready to build a plug-in:

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

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

> **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 **<GAX_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 GAX 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: '8.5.1',        // 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"
    }
  ]
}
```

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 GAX folder or **styles/scripts** subfolders. The options for loading custom js/css files are as follows:

- `pulse/load_css_custom=/gax/styles/custom.css`
- `pulse/load_js_custom=/gax/scripts/custom.js`

You can specify many options with `load_css/load_js` prefixes:

- `pulse/load_css_<style_name_1>=/gax/styles/custom_1.css`
- `pulse/load_css_<style_name_n>=/gax/scripts/custom_n.css`
- `pulse/load_js_<script_name_1>=/gax/styles/custom_1.js`
- `pulse/load_js_<script_name_n>=/gax/scripts/custom_n.js`

The options do not require GAX restart.

> ### Tip
> Files placed in GAX 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 restart the Genesys Pulse Collector application when the backup Configuration Server switches to the primary mode, you must specify the 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

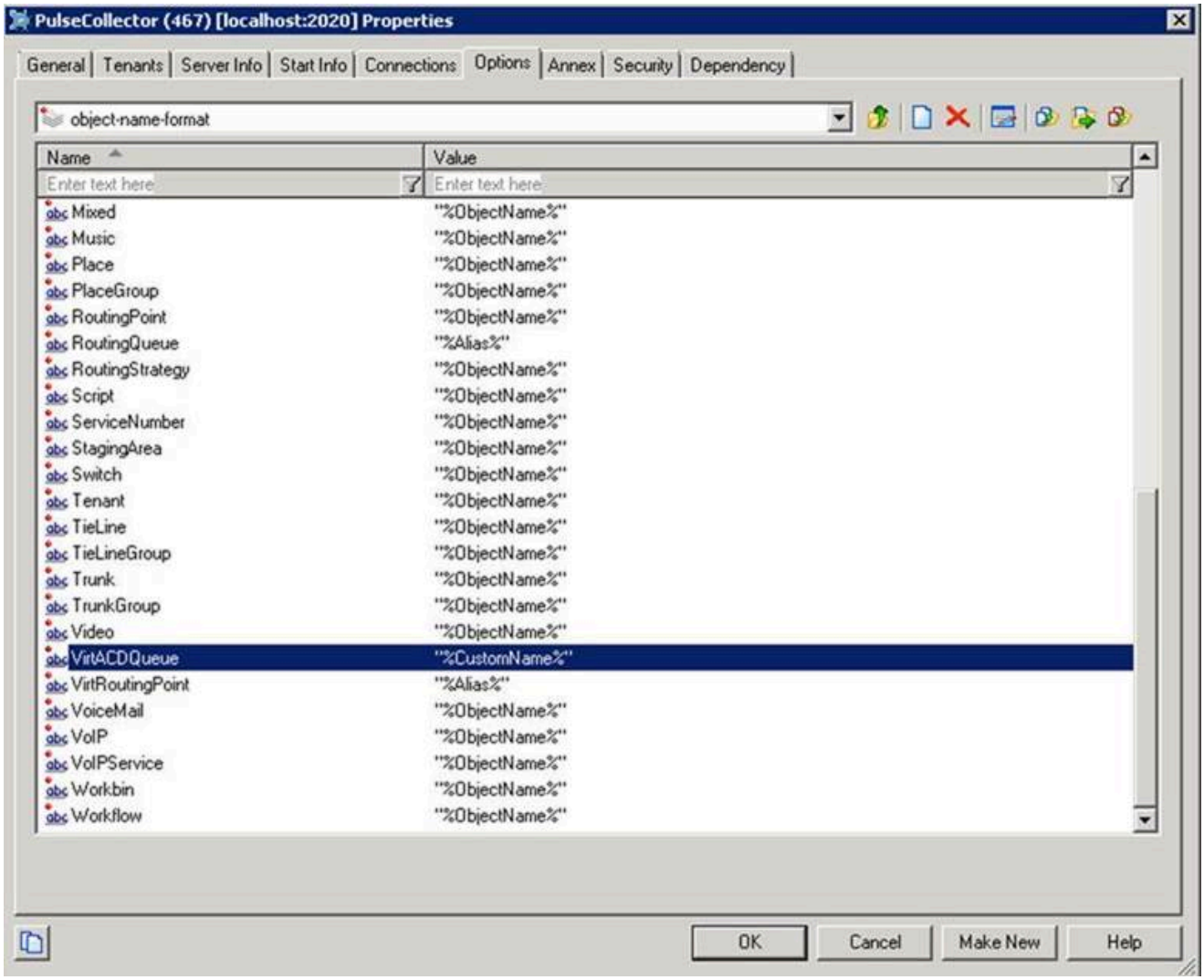
Stop Genesys Pulse Collector on UNIX using either of the following methods:

- 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 Pulse dashboard that works best for your business needs. You can assign a different display name or nickname to Pulse objects, such as queues, DN groups, agents, or VAGs.

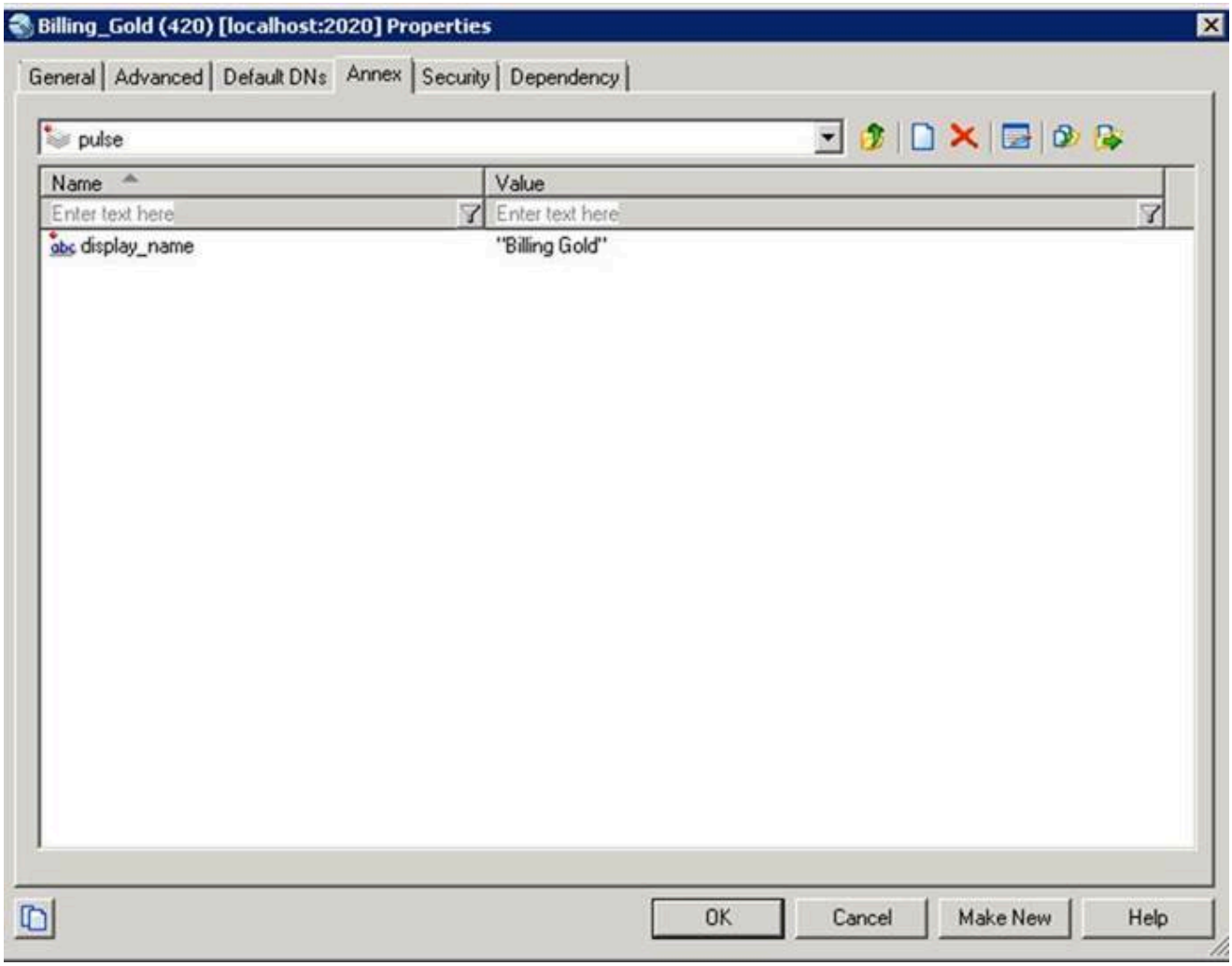
Change the default option



In order to use a custom name, you need to set the value of the specific object type option to `%CustomName%` in the 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), 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 Pulse Collector

You must restart Pulse Collector to apply the changes.

Cleanup Snapshot Data

Pulse Collector may generate large amounts of layout data snapshot files each day. These files are usually necessary for a relatively short time period, like 24 hours, and can be safely removed afterwards.

> **Important**
>
> For the troubleshooting purposes Genesys strongly recommends to keep Pulse Collector outputs for at least recent 2-3 days.

Genesys provides a filesystem cleanup utility (cleanup tool), to simplify this task in data intensive environments. The cleanup tool is included within the Pulse Collector installation for all supported operating systems. Unlike a simple file removal by created date, this cleanup tool analyzes layout snapshot files required for Pulse and removes only those that were not in use for the last 24 hours.

Deployment

After installing Pulse Collector, you can find the executable file called cleanuptool in Linux and cleanuptool.exe in Windows, located in the same folder that contains the Pulse Collector executable. Generally, this is a console application, which accepts some command-line options and requires the configuration file.

Genesys recommends you run this cleanup tool regularly using standard task scheduling software available within your operating system: Cron for Linux or Windows Task Scheduler for Windows. The cleanup tool must be run under user account that has permissions to read, write, and remove files and directories on the filesystem where Pulse Collector writes the layout data snapshots.

Define your scheduling period based on the data generation intensity and your target filesystem capacity. Genesys recommends that you run the cleanup tool every hour.

Command-line options

The cleanup tool supports number of command-line options:

Usage: cleanuptool [options] <path> ...

Options:

-c	--config-file <file-name>	Specify new configuration file to us (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

As stated above, the filesystem cleanup tool requires a configuration file, which defines the time periods for which files should be preserved. If the timestamp of file matches the time period, the file is preserved. Otherwise, the file is deleted.

You can also simulate full processing by running the cleanup tool in the “dry run” mode, when it just collects the information about the files to be deleted, but does not actually delete them.

The configuration file has the standard INI-file format: It contains a few sections and allows comment lines that start with semicolon. A sample configuration file is in the Pulse Collector installation directory, it is called cleanuptool.ini.sample.

General Section

The **general** section has two parameters:

- **active-intervals**—(required) lists names of active intervals. There is no default value.
- **measure**—(optional) defines time measurement unit for interval points. Valid values: m, min, minutes—minutes, s, sec, seconds—seconds. The default value is minutes.

Intervals Section

The **intervals** section defines set of intervals. Each interval is defined as separate INI file parameter under this section, in the format NAME=VALUE. You can define multiple intervals and still use only the ones you need. You can list the names of the ones you want to use in the active-intervals option in the general section.

VALUE is parameter definition string which format is defined as follows:

The form of interval definition is following:

[ALIGNMENT]<BEGIN>-<END>[:<GRANULARITY]>

Detailed description of elements:

- [ALIGNMENT]—Optional, alignment of granulation points. By default intervals are aligned to granularity number. Possible values: 'A' - aligned, U - unaligned.
  - <BEGIN>—Required, beginning of the interval, included to interval.
  - <END>—Required, end of the interval, not included to interval.
- [GRANULARITY]—Optional, granularity of the interval. Affects the points of granulation, which determine the files to be preserved. The default value is 0.
  Note: A zero value of granularity means that all files matching to the interval are preserved.

Examples

0-1	Preserves all files with modification time difference in range between 0 and 1 measure points.
10-20:1	In the interval from 10 to 20 measure points, preserves the files near granulation points standing each 1 measure point, aligned to granularity (according to default behavior).
U10-20:3	In the interval from 10 to 20 measure points preserves the files near granulation points standing each 3 measure points, unaligned to granularity. Points are 10, 13, 16, 19.
A20-60:7	In the interval from 20 to 60 measure points preserves the files near granulation points standing each 7 measure points, aligned to granularity. Points are 21, 28, 35, 42, 49, 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

Change History

This section lists content that is new or that has changed significantly since the first release of this content. The most recent changes appear first.

For *New in This Release* content, see the Pulse landing page for this release.

New in Document Version 8.5.1

The following topics have been added or changed since the previous release:

- Language Pack Deployment was added.
- Support for WebDAV was added.
- Pulse Restful Web Service API was updated.
- The following Configuration Options were updated:
  - The default value of max-objects-per-layout in the [limits] section was changed to 100.
  - New options were added to the new [parallel-processing] section.
  - max-stat-data-queue-size was added to the [statistic-request-handling] section.
  - worker-thread-count was added to the [transport-file] section.
  - exchange was added to the [transport-rabbitmq] section.
  - exchange-prefix was updated in the [transport-rabbitmq] section.

New in Document Version 8.5.010

The following topics have been added or changed since the previous release:

- Deploying Pulse Collector and Pulse was updated to include:
  - Software Requirements:
    - Windows Server 2012 Hyper-V
    - Microsoft SQL Server 2012 Cluster
    - Oracle 12c RAC
    - GAX release 8.5.000.68
  - Deploying RabbitMQ was added to provide an option for Quick Widget Updates.
  - The following privileges in Configuring User Access were changed:

    - Pulse Read Layout renamed Pulse View Dashboard.
    - Pulse Update Layout renamed Pulse Edit Widget Display .
    - Pulse Write Layout renamed Pulse Manage Widgets.
    - Pulse Write Template renamed Pulse Manage Templates and Default Dashboard.
    - Pulse Read All Layouts and Pulse Write Snapshot added.
  - `amqp-client-3.3.4.jar`, `spring-amqp-1.3.6.RELEASE.jar`, `spring-rabbit-1.3.5.RELEASE.jar`, and `spring-jetty-1.1.0.RELEASE.jar` added to the list of files to manually add if the installation procedure fails to find the GAX installation.
- Pulse Restful Web Service API was added.
- The following Configuration Options were changed:
  - The [transport-gpb-out] section was renamed [transport-file].
  - In the [collector] section, removed the output-transport option.
  - In the [statistic-request-handling] section, removed the data-flow-timeout and data-flow-check-interval.
  - In the [transport-gpb-out] section, removed the transport-type option.
  - The [transport-rabbitmq] section was added.
- The following Application Files were added:
  - Upgrade scripts: pulse_upgrade_08.5.001.02_mssql.sql, pulse_upgrade_08.5.001.02_oracle.sql, and pulse_upgrade_08.5.001.02_postgres.sql
  - JAR files: amqp-client-3.3.4.jar, spring-amqp-1.3.6.RELEASE.jar, spring-rabbit-1.3.5.RELEASE.jar, and spring-jetty-1.1.0.RELEASE.jar

New in Document Version 8.5.0

The following topics have been added or changed since the previous release:

- The Architecture diagram was updated.
- Software requirements and deployments were updated in Deploying Pulse Collector and Pulse.
- The step to execute `ALTER DATABASE <Pulse DB> SET READ_COMMITTED_SNAPSHOT ON;` was added to Deploying the Pulse database for the Microsoft SQL Server.
- Starting Pulse was updated.
- Prerequisites and Starting Pulse were updated.
- The following Configuration Options were changed:
  - Removed the [`layout-history`], [`parallel-processing`], [`thresholds`], and [`transport-gpb-history`] sections.
  - In the [`collector`] section:
    - Removed the `history-transport` option.

- In the [configuration-monitoring] section:
  - Added the excluded-objects-propagation-delay and remove-dynamic-object-delay options.
  - Removed the auto-detect-dynamic-metagroups, enable-differential-layout-recheck, recheck-active-layout-definition, and strict-tenant-security options.
- In the [layout-validation] section:
  - Renamed the enable-strict-tenant-security-validation-rule option to validate-strict-tenant-security.
- In the [limits] section:
  - Removed the max-custom-actions-per-layout, max-custom-thresholds-per-layout, max-threshold-associations-per-layout options.
- In the [output] section:
  - Added the collector-snapshot-log-level and snapshot-log-level option.
  - Removed the cleanup-output option.
- In the [pulse] section:
  - Added the editable_templates, install_templates, and snapshot_expire_timeout options.
  - Removed the pause_timeout option.
- In the [scripting] section:
  - Added the stop-compute-formula-threshold-for-snapshot option.
  - Removed the cumulative-formula-script-execution-timeout, cumulative-threshold-script-execution-timeout, enable-extended-statistic-value, post-processing-script-execution-timeout, and threshold-script-execution-timeout options.
- In the [statistics-request-handling] section:
  - Removed the optimize-statistic-requests options.
- In the [transport-gpb-out] section:
  - Added the output-file-mode option.
  - Removed the backend-file-ext, cleanup-file-modification-age-threshold, generate-layout-info-file, and output-format options.