



## Pulse Deployment Guide

Pulse 8.5.1

# Table of Contents

<b>Pulse Deployment Guide</b>	<b>3</b>
<b>Pulse Architecture and Components</b>	<b>5</b>
<b>Deploying Pulse Collector and Pulse</b>	<b>8</b>
<b>Pulse Configuration Options</b>	<b>30</b>
<b>Pulse Restful Web Service API</b>	<b>60</b>
<b>Starting and Stopping Pulse</b>	<b>80</b>
<b>Starting and Stopping Pulse Collector</b>	<b>81</b>
<b>How do I capture Pulse Collector memory dumps?</b>	<b>0</b>
<b>Change History</b>	<b>4</b>

# Pulse Deployment Guide

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 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 Pulse Help for an overview of how to use Pulse.

## Important



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

## About Pulse

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

---

Pulse Help  
Architecture  
Framework Components

## Deploying Pulse

Plan your environment and install deploy Pulse.

---

Deploying Pulse and Pulse Collector  
Configurations Options  
Configure Memory Dumps  
Display external data in Pulse

## Starting and Stopping

Start or stop the applications:

---

Pulse  
Pulse Collector

## Change History

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

---

Change History

# Pulse Architecture and Components

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. Pulse uses widgets to display user-defined List, Donut, Key Performance Indicator (KPI), or Grid charts of statistics for objects.

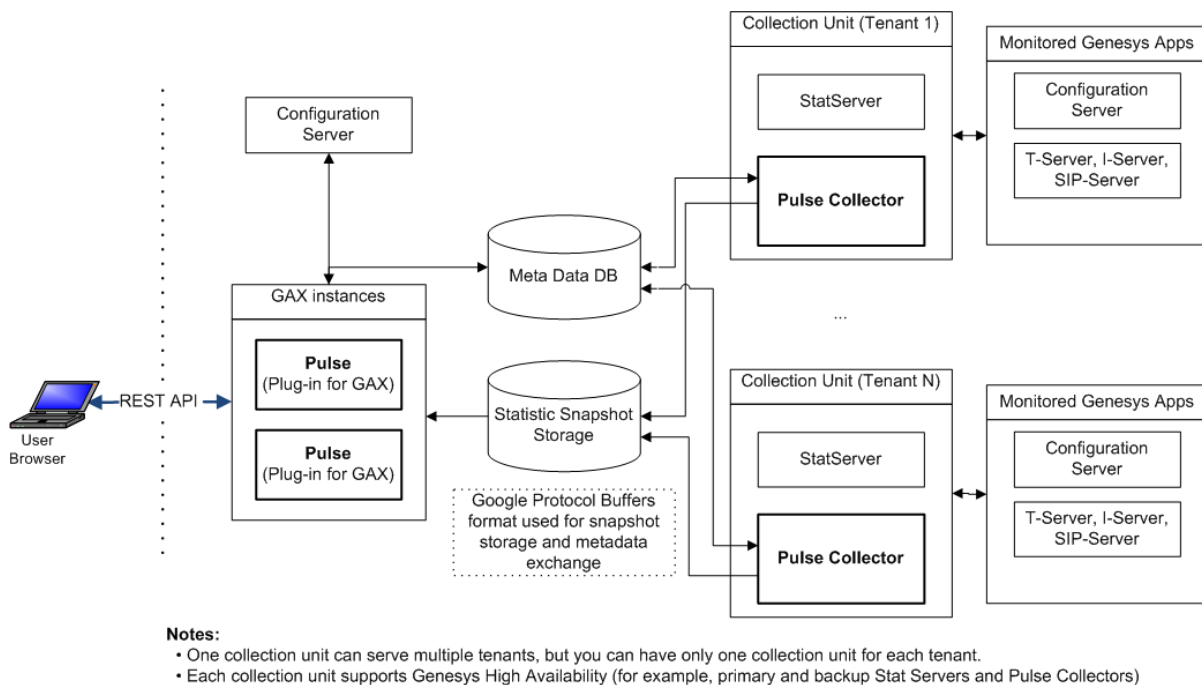
Using Pulse you can:

- 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.

## Pulse Architecture

For detailed architecture, see the Pulse Hardware Sizing and Performance Information.

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



## Pulse Collector

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

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

## Pulse

Pulse is a GAX plug-in that 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 Pulse Database, which it shares with Pulse Collector.

You can have only one Pulse application for each GAX instance. For configuration, Pulse requires a `[pulse]` section and is added to the options of the GAX Application object. Pulse must have its own database access point, but otherwise uses Management Framework and other connections from GAX.

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

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

## Genesys Framework Components

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 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).

Pulse Collector uses this data to provide content for Pulse.

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

## Stat Server

Stat Server tracks information about customer interaction networks consisting of one or more contact centers in conjunction with one or more computer networks. Stat Server receives information from one or more T-Server or SIP Server applications and converts the accumulated data for directory numbers, agents, agent groups, and so on, into statistical information.

As a client of Stat Server, Pulse Collector requests statistics for objects belonging to particular reports. Stat Server supplies both information about the interactions that the objects that handle as well as noninteraction-related data about the objects themselves (for example, agent status). Pulse Collector returns information for all `stattypes` that are configured in the options of all Stat Servers to which 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, Pulse Collector reads information about active widgets and updates the layout statuses, when layout status changes occur within the Pulse Collector.

Refer to the [Framework 8.1 DB Server User's Guide] for information about DB Server.

# Deploying Pulse Collector and Pulse

## Important

- This Deployment Guide provides instructions for a new installation of Pulse. To migrate from an earlier version of Pulse, start with the Deployment Procedure.
-  In these procedures, use the instructions provided in Genesys Administrator Extension (GAX) Help or the Framework 8.x Deployment Guide, and add the object-specific configuration requirements listed here.

## 1. Confirm Software Requirements.

### Confirm Software Requirements

You deploy Pulse as a GAX plug-in on a web application server to be accessed using a web browser through the GAX user interface. No additional Genesys software is needed on the client machine.

The following are prerequisites for Pulse deployment:

- GAX release 8.5.200.18 and higher. See the GAX Deployment Guide for details.
- Tomcat is no longer a prerequisite to use GAX. For those who choose to use Tomcat instead of Jetty, refer to the GAX Deployment Guide for additional information.
- 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 Pulse and Pulse Collector if you plan to install 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 Pulse supports the following relational database platforms:



- Microsoft SQL Server 2008, 2012, 2012 Cluster
- Oracle 11g, 12c, 12c RAC
- PostgreSQL 9.0
- The computer on which you install Pulse must be running one of the following:
  - Windows Server 2008 R2 (64-bit), 2012 (64-bit), 2012 Server Hyper-V
  - Red Hat Enterprise Linux AS 6.x (64-bit), with Updates from RHN enabled

### Important



Both Pulse Collector and GAX must be installed on the same host. Genesys does not support Pulse deployments that have components on separate hosts.

- If you use Red Hat Enterprise Linux AS, you must also install the following libraries to ensure backward compatibility for applications: compat-db
  - compat-expat1
  - compat-glibc
  - compat-libf2c-34
  - compat-libgcc-296
  - compat-libgfortran-41
  - compat-libstdc++-295
  - compat-libstdc++-296
  - compat-libstdc++-33
  - compat-libtermcap
  - compat-openldap
  - openssl098e

For more information, see [linuxtopia.org](http://linuxtopia.org).

- If you use Windows, you must also install Microsoft Visual C++ 2010 Redistributable Package (x86 for Collector x86 and x64 for Collector x64).
- Although not required for deployment, you must have Stat Server release 8.1.2 or higher installed for basic operation.
- You must install Java 1.7 or higher.

## 2. Prepare the Pulse Database.

## Prepare the Pulse Database

In High Availability (HA) configurations, configure both 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 the Pulse database (DB). Replace <Pulse DB> with the name of the Pulse DB and make sure that there are no other connections to the 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 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. Deploy Pulse Collector.

### Deploy Pulse Collector

To deploy Pulse Collector, which connects directly to Stat Server to collect statistics for contact center objects. Pulse Collector also connects to DB Server, through which it accesses the database where reporting layouts are stored.

1. Upload 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 Pulse Collector Installation Package (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 Pulse Collector Installation Package and Template:
  - a. Click on the Pulse Collector Installation Package to open the Properties tab.

#### Important



The Pulse Collector 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



The InstallPath should point to an empty folder where you plan to install Pulse Collector.

#### Tip



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

3. Optional: To change the default options for Pulse Collector in GAX, open the Pulse Collector `Application` object modify the values of configuration options described in Pulse Collector Application Object.
4. For a high-availability (HA) deployment, repeat step 2 to install the backup instance to separate folder. You need a Pulse Collector for each instance of the Pulse application.
5. In GAX, add the Pulse Collector `Application` object to the connections of your GAX `Application` object.
6. For a load-balanced environment configuration with two GAX applications and Pulse plug-ins, associate one GAX with the primary Pulse Collector and the second GAX with the backup Pulse Collector as follows:
  - a. Add the primary Pulse Collector `Application` object to the connections of the first (or primary) GAX `Application` object.
  - b. Add the backup Pulse Collector `Application` object to the connections of the second (or backup) GAX `Application` object.
  - c. Add the backup Pulse Collector `Application` object to the **General** tab of the primary Pulse Collector `Application` object as a backup server.  
**Note:** Several GAX instances can be connected to Collector. For example if you have 4 load balanced GAX applications you can connect 2 of them to Primary Collector and 2 of them to Backup Collector.
7. Create a new Database Access Point (DAP), which is necessary for connectivity to the Pulse database through the DB Server:
  - a. Select the Default connection type.
  - b. Specify Database Server `Application` object in **DB Server** field.
  - c. Specify the connectivity parameters for your RDBMS.
  - d. On the **Ports** tab, change the default port value to the value of your DB Server port.
8. In the connections of the Pulse Collector `Application`, add the DAP that is to be used by Pulse Collector.
9. In the connections of the Pulse Collector `Application`, add the primary Stat Server application that is to be used by Pulse Collector.
10. Add the Tenant objects to the Tenants tab for all Pulse Collectors that you plan to monitor in Pulse.

## 4. Deploy Pulse.

### Deploy Pulse

Configure the necessary objects required by Pulse using GAX.

---

1. Upload 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 Pulse Installation Package (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 Pulse Installation Package and Template:

- a. Click on the Pulse Installation Package to open the **Properties** tab.

Important



The 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 Pulse plugin will be installed (for example, C:\123).
- When installing 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 Pulse directly on the server by executing the Pulse installation procedure from the Pulse installation package.

If the installation procedure fails to find the GAX installation, you can manually copy the Pulse plugin files to enable Pulse in GAX.



After installation, find all jar files in root directory of installed 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.

3. 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 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.

4. Create a new Database Access Point, which is necessary for connectivity to the 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 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 Framework 8.0 DB Server User's Guide for additional information about configuring this application.

5. Add the Database Access Point to the connections of your GAX `Application` object.
6. Navigate to **Configuration > Environment > Application**, select the GAX Application, **Permissions** tab and configure the SYSTEM account to have Read, Update, and Execute permissions.
7. For a High Availability (HA) deployment,
  - a. Complete Steps 2-6, for each GAX `Application` object to be used with the 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 Pulse instances must use the same database.

8. From the user account created when you prepared the 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.

## 5. Optional: Deploy RabbitMQ.

### Optional: Deploy RabbitMQ for quick widget updates

#### Important

Pulse supports quick updates for CurrentStatus and ExtendedCurrentStatus statistics only to prevent a high performance load this causes on StatServer and 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 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)



2. Determine whether to use RabbitMQ with Pulse using Cluster or Single-Node configurations:

- **Cluster configuration**—Use at least the same number of RabbitMQ instances as the number of Pulse Collector applications. RabbitMQ can run on any host: either the one where Pulse runs or any other host accessible over reliable network.

The default configuration should be one RabbitMQ instance running on every host where 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 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 2008/2012:
  - 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 2008/2012:
  - 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 RabbitMQ 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 Pulse. If you create a vhost with a different name, you must specify it in Pulse Collector configuration options. On any of the hosts run the following command:

```
rabbitmqctl add_vhost /pulse
```

- j. Create a user for Pulse with name **'pulse'** and password **'pulse'**. If you create a user with a different name and password, you must specify them in 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 Pulse Collector to work with RabbitMQ cluster you need to have `transport-rabbitmq` section configured in options of Pulse Collector application object. The following options should be added to that section

option	value	default value	comment
<code>disabled</code>	<code>no</code>	<code>no</code>	You can completely remove this option from the section.
<code>nodes</code>	<code>host1:5672,host2:5672</code>	<code>localhost:5672</code>	<p>List a single host name with a port for single node configuration, and all cluster nodes host names and ports for cluster configuration.</p> <p>Note: If Pulse Backed and Pulse Collector it connects to run on one of the hosts in the list, put this host as the first entry in the list.</p>
<code>exchange-prefix</code>	<code>pulse.collector</code>	<code>pulse.collector</code>	The value of this field should be compatible with the permissions given to a user during configuration. The default

			value is compatible with the values used in this guide.
vhost	/pulse	/pulse	If needed, change the virtual host name to the one used during the RabbitMQ configuration.
username	pulse	pulse	If needed, change the user name to the one used during the RabbitMQ configuration.
password	pulse	pulse	If needed, change the password to the one used during the RabbitMQ configuration.
max-queue-length	1000	1000	Read the section on RabbitMQ memory usage for details.

6. Restart Pulse Collector and GAX to apply changes.

Important

### RabbitMQ memory and disc usage

RabbitMQ instances should not store any Pulse application data on disc, so the disc usage is insignificant unless the message queue for any of the 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 Pulse, which is controlled by option `max-queue-length` in section `transport-rabbitmq` of 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:

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

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.


## 6. 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 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 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 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).

4. Restart both Stat Server applications.

## 7. Configure user access.

### Configure User Access

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

#### Privilege Details:

- **Pulse View Dashboard**—View dashboard.
- **Pulse Edit Widget Display**—Edit widget display options.
- **Pulse Manage Widget**—Remove, create, clone, and edit widgets.
- **Pulse Manage Templates and Default Dashboard**—Remove, create, clone, and edit custom templates. Set a dashboard to be the default dashboard.

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

- **Pulse Read All Layouts**—View all 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.

#### Important

Users without permission to edit their own Person object cannot save their Custom dashboard configuration. Users without permission to edit both their own Person 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.

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** and **Pulse Edit Widget Display** in addition to **Pulse Manage Widgets**.

### Important



You must assign at least the **Pulse View Dashboard** 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; and **Update** is required to set a Dashboard as the Default dashboard.
    - c. User's own Person object
      - i. Select **Configuration**.
      - ii. From the **Accounts** pane, choose **Persons**.
      - iii. Click to open the required person to view its properties.
      - iv. Select **Permissions** tab.
      - v. Add the **Person** or **Access** group containing this user.
      - vi. Add required permissions: **Update** is required to save dashboard configuration.
    - d. 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 Environment tenant are required to log in to GAX; and **Update** on user's tenant is required to set a Dashboard as the Default dashboard.

## Pulse Configuration Options

The application templates for 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 Pulse. 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 Pulse stores the results of the object accessibility check in order to reduce the load on the Configuration server.

#### **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.

#### **health\_expire\_timeout**

Valid Values: zero or any positive number

Default Value: 30

Changes Take Effect: After restart

Specifies how long, in seconds, 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 Pulse installs or updates the Genesys-provided templates for the current release when 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 Pulse restores all original Genesys-provided templates after restart for the current release. 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.

### **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 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.

## **Pulse Collector Application Object**

### **[collector] Section**

#### **application-id**

Valid Values: 0-65535

Default value: 0

**Note:** A zero (0) value instructs Pulse Collector to attempt to determine the application ID automatically.

The Pulse Collector application ID, as registered in the Pulse database.

**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 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 Pulse Collector is running.

## [configuration-monitoring] Section

**check-layout-presence-timeout**

Valid Values: 0-3600

Default Value: 900

Changes Take Effect: After restart

Specifies how often, in seconds, 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, Pulse Collector obtains updates from the 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 Pulse Collector attempts to propagate excluded objects in the affected layouts after an object is deleted. A zero value eliminates the timeout and 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 Pulse Collector verifies metagroup contents (such as Agent Group, Place Group, and DN Group) after notification from Configuration Server notifies 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 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 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 Pulse Collector waits before re-checking the 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 Pulse Collector waits before excluding and closing

statistics for the Agent from affected layout that was excluded from the StatServer-based VAG. A zero value turns off the delay and Pulse Collector processes changes immediately.

## **[heartbeat] Section**

### **external-heartbeat-folder**

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

Default Value: \\server\share\path\to\heartbeat\folder

Changes Take Effect: After GAX restart

Enables a remote GAX instance to access the Collector heartbeat. This option must be used together with the heartbeat-folder.

### **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 Pulse Collector writes the heartbeat file.

### **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.
- collector-db—The main DB connection should be available.

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 Pulse Collector.

**Note:** 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

Enables Pulse Collector to fail the layout validation when the option strict-tenant-security in the configuration-monitoring section is set to yes and 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.

## **[localization] Section**

### **default-snapshot-message-language-code**

Valid Values: Language code (non-empty string, typically 2 letters)

Default Value: en

Changes Take Effect: After restart

Specifies the default language code to use for layout snapshot messages, if the layout snapshot does not override this value.

### **language-pack-folder**

Valid Values: Folder path

Default Value: ./messages

Changes Take Effect: After restart

Specifies the folder from which Pulse Collector reads localized message files to show layout errors in the localized form.

## **[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

**Note:** To ease the troubleshooting process, consider using unique names for log files that different applications generate.

### **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.

### **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.

### **CommDN**

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 Comm DN. 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.

### **DestinationLabel**

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 Destination Label. 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.

### **GVPDID**

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 GVPDID. 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.

### **TieLine**

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 Tie Line. For details, see “Object Properties” at the bottom of this section.

### **TieLineGroup**

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 Tie Line Group. For details, see “Object Properties” at the bottom of this section.

### **Trunk**

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 Trunk. For details, see “Object Properties” at the bottom of this section.

### **TrunkGroup**

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 TrunkGroup. 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.

### **VoIPService**

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 Service. For details, see “Object Properties” at the bottom of this section.

### **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, 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
<ul style="list-style-type: none"> <li>• %EmployeeID:10%</li> <li>• %EmployeeID:L10%</li> </ul>	Both specify 10 characters of property EmployeeID from the left side.
<ul style="list-style-type: none"> <li>• %EmployeeID:.10%</li> <li>• %EmployeeID:L.10%</li> </ul>	Specifies 10 characters of property EmployeeID from left, but if length was less than 4 symbols, pad it with spaces.
<ul style="list-style-type: none"> <li>• %DBID:R4%</li> </ul>	Specifies 4 characters of property DBID from the right side.
<ul style="list-style-type: none"> <li>• %DBID:R04%</li> </ul>	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	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	<p>Specifies the ID number of the object in Pulse Collector.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• This is the typically the configuration layer DBID, but for some types of objects (for</li> </ul>

		<p>example, Campaign Group, Campaign Calling List) this is a composite 64-bit ID.</p> <ul style="list-style-type: none"> <li>The composite 64-bit ID is an unsigned 64-bit number with the following composition: <ul style="list-style-type: none"> <li>higher 32 bits: DBID of Campaign</li> <li>lower 32 bits: DBID of Calling List or Agent/Place group</li> </ul> </li> </ul>
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).
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 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 Pulse user interface. If the set frequency, however, is greater than the value of this option, 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 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 Pulse user interface. If the set frequency, however, is less than the value of this option, 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, 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 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.

### **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, Pulse Collector assigns the particular formula-based statistic the `ERROR` 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 Pulse Collector to request all statistics through the StatServer New API that uses the proper parameter set.

### **data-source-choice-strategy**

Valid Values: PrimaryRunMode, LastGood, FirstAvailable, MostUpToDate, PrimaryInCME

**Note:** All values are case-insensitive.

Default Value: PrimaryRunMode

Changes Take Effect: After restart

Specifies which StatServer configured in the configuration layer that Pulse Collector uses as a data source for a snapshot:

- PrimaryRunMode—Pulse Collector uses the StatServer running in the primary mode. If both Stat Servers appear to be backup, Pulse Collector attempts the next strategy.
- LastGood—Pulse Collector uses the last good StatServer if available. Otherwise, Pulse Collector attempts the next strategy.
- FirstAvailable—Pulse Collector uses the Primary StatServer if available. If the Primary StatServer is unavailable, Pulse Collector uses the Backup StatServer. Otherwise, Pulse Collector attempts the next strategy.
- MostUpToDate—Pulse Collector uses StatServer that sent statistic data with most recent Server Time. Otherwise, Pulse Collector attempts the next strategy.
- PrimaryInCME—Pulse Collector always uses the Primary StatServer.

### **max-stat-data-queue-size**

Valid Values: 1-2147483647

Default Value: 2147483647

Changes Take Effect: After restart

Specifies the limit for the internal Pulse Collector statistical data queue, which stores unprocessed data from StatServer. Pulse drops incoming data if it exceeds this limit. Genesys recommends you set the value of this option to 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 Pulse Collector servers. If Stat Server disconnects 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, Pulse Collector closes the current connection to StatServer and attempts to reconnect.

### **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 Pulse Collector to log verbose messages about the objects for which it requests statistics.

## **[transport-file] Section**

### **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-output-folder**

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

Default Value: \\server\share\path\to\output\folder

Changes Take Effect: After GAX restart

Enables a remote GAX instance to access the Collector output files. This option must be used together with the output-folder.

### **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.

### **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 Pulse Collector writes output files. If you specify a folder that does not exist, Pulse Collector creates it for you.

**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 Collector restart

Specifies the full name of a RabbitMQ exchange where Collector writes delta messages. This should be different for primary and backup Pulse Collectors. The valid value for this option overrides the value specified by the `exchange-prefix` configuration option. In a cluster all primary collectors are given one exchange name, and all backup collectors the different one.

**Important:** The value of exchange option should be different for the primary and backup collectors.

**exchange-prefix**

Valid Values: A non-empty sequence of these characters: letters, digits, hyphen, underscore, dot, or colon.

Default Value: pulse.collector

Changes Take Effect: After Pulse Collector restart

Specifies the prefix for a name of a RabbitMQ exchange. The exchange prefix appends a dot and a DBID of Pulse Collector to form the name of an exchange.

**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

Specifies the nodes and ports for messaging.

**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.

# Pulse Restful Web Service API

Create external data for display in Pulse by using the Pulse Restful Web Service API.

## Important



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

## Methods

- **/api/wbrt/layouts**

### [+] GET

Returns an array of layouts for a tenant. If Tenant ID is omitted, returns the layouts for user's tenant and templates.

Example request URIs:

- /api/wbrt/layouts
- /api/wbrt/layouts?tenantId=1
- /api/wbrt/layouts?tenantId=0

Request query parameters

Name	Required	Type	Description
tenantId	not	integer	Specifies tenant to which layouts belong
uscn	not	integer	Specifies USCN, for filtering layouts changed after this USCN (non-inclusive)

		string, available values:	
type	not	ItGENERIC ItPCREGULAR ItPCPERFORMANCE ItDATADEPOT ItOTHER	Specifies layout type

Required permission:

- **Pulse Read All Layouts** - for any request parameters
- **Pulse View Dashboard** - for all templates request (tenantId=0)

Available response representations:

- 200 - application/json with array of layouts
- 400 - Request is not valid, application/json with details:
  - { "message": "INVALID\_URL" } - Specified wrong request parameters.
- 403 - User does not have **Pulse Read All Layouts** permission or user has **Pulse View Dashboard** permission, but the request is different from /api/wbrt/layouts?tenantId=0 (request for all templates)

## [+] POST

Creates a new layout.

Available request representations:

- application/json with layout configuration

Required permission:

- **Pulse Manage Widgets** for creating layouts
- **Pulse Manage Templates and Default Dashboard** for creating templates

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": "LAYOUT\_PARSE\_ERROR" }, when widget has an invalid format
- 403 - User does not have **Pulse Manage Widgets** permission and tries to create layout or user does not have **Pulse Manage Templates and Default Dashboard** permission and tries to create template

- **/api/wbrt/layouts/<id>**

### **[+] GET**

Returns layout with specified ID.

Required permission:

- **Pulse Read All Layouts** or **Pulse View Dashboard**

Available response representations:

- 200 - application/json with requested layout
- 403 - User does not have **Pulse View Dashboard** or **Pulse Read All Layouts** permissions
- 404 - Layout does not exist or the layout is not accessible by the user

### **[+] PUT**

Updates layout with specified ID. If there is more than one widget for this layout then a new layout is created.

Available request representations:

- application/json with layout configuration

Required permission:

- **Pulse Manage Widgets**, for updating layout
- **Pulse Manage Templates and Default Dashboard**, for updating template

Available response representations:

- 200 - Layout was updated, redirects to new newly created layout
- 400 - Provided layout configuration is not valid, application/json with details:

- { "message": "LAYOUT\_PARSE\_ERROR" } - layout has invalid format
- 403 - User does not have **Pulse Manage Widgets** permission and tries to update layout or user does not have **Pulse Manage Templates and Default Dashboard** permission and tries to update template
- 404 - Layout not exists, application/json with details:
  - { "message": "LAYOUT\_NOT\_FOUND" } - layout does not exist

## [+] DELETE

Delete layout with specified ID.

Required permission:

- **Pulse Manage Widgets**, for deleting layout
- **Pulse Manage Templates and Default Dashboard**, for deleting template

Available response representations:

- 200 - Layout was deleted
- 204 - Layout was already deleted
- 403 - User does not have **Pulse Manage Widgets** permission and tries to delete layout or user does not have **Pulse Manage Templates and Default Dashboard** permission and tries delete template

- /api/wbrt/layouts/<id>/snapshot

## [+] GET

Returns recent snapshot for specified layout.

**Note:** If user doesn't have **Pulse Read All Layouts** privilege then this method performs additional row filtering based on user access restrictions for snapshots with `layout_type = ltPCREGULAR` and `layout_type = ltPCPERFORMANCE`.

Rows with objects not accessible for user are filtered out. It must work as follows:

if column `_Object$CfgType` exists, then use pair `(_Object$CfgType, _Object$ID)` to check permissions; otherwise attempt the usual combination `(_Object$Type, _Object$ID)`.

Required permission:

- **Pulse View Dashboard**

Available response representations:

- 200 - application/json snapshot
- 204 - no content, snapshot does not exist
- 403 - User does not have **Pulse View Dashboard**
- 404 - Related resource is not found, application/json with details:
  - { "message": "LAYOUT\_NOT\_FOUND" } - layout does not exist

## [+] POST

Saves a snapshot for layout with specified ID. Uses timestamp property from Layout Snapshot to distinguish different snapshots. Any saved snapshot with the same timestamp is overwritten.

Available request representations:

- application/json with snapshot

Required permission:

- **Pulse Write Snapshot**

Available response representations:

- 200 - snapshot was successfully saved
- 400 - snapshot is not valid, application/json with details:
  - { "message": "SNAPSHOT\_PARSE\_ERROR" } - snapshot has invalid format
  - { "message": "INVALID\_URL" } - URL layout ID does not match layout ID in the snapshot
- 403 - user does not have **Pulse Write Snapshot** permission
- 404 - related resource not found, application/json with details:
  - { "message": "LAYOUT\_NOT\_FOUND" } - layout does not exist

- **/api/wbrt/layouts/<id>/snapshots**

## [+] GET



Returns array of snapshots for specified layout and matched filter period. If the 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=1400000000&frequency=10

Request query parameters

Name	Required	Type	Description
start	not	long	Unix epoch timestamp indicating the start of the period for which snapshots should be returned. If not specified then all snapshots generated before end time are returned.
end	not	long	Unix epoch timestamp indicating the end of the period for which snapshots should be returned. If not specified then all snapshots generated after start time are returned.
frequency	not	integer	Minimum interval, in seconds, between two snapshots in history. Needed for reducing the number of snapshots returned by this request. Default value is 0, so all existing snapshots inside the specified period are returned.
columns	not	Array of Strings	Array of Column.id that is to be included to snapshot. If not specified then all columns are included.

**Note:** This method performs additional row filtering based on user access restrictions for snapshots with layout\_type = **ItPCREGULAR** and layout\_type =

**ItPCPERFORMANCE.** Rows with objects not accessible for user are filtered out. It must work as follows: if the column `_Object$CfgType` exists then use pair (`_Object$CfgType`, `_Object$ID`) to check permissions; otherwise attempt the usual combination (`_Object$Type`, `_Object$ID`).

Required permission:

- Pulse Write Snapshot

Available response representations:

- 200 - application/json with array of snapshots
- 403 - In cases when user does not have **Pulse View Dashboard**

- **/api/wbrt/widgets**

**[+] POST**

Creates a new widget. If the parameter `default` is not specified then a user creates a widget that cannot be seen by other users. If the parameter `default` is specified and not equal to `false` then the default widget is created and can be seen by anyone.

Example request URI(s):

- `/api/wbrt/widgets?default=true`
- `/api/wbrt/widgets?default=false`
- `/api/wbrt/widgets`

Request query parameters

Name	Required	Type	Description
default	not	string	If specified and different from <code>false</code> then a default widget is created.

**Note:** `owner_user_name` field is ignored; widget is created for current user or as default widget.

Available request representations:

- application/json with widget configuration

Required permission:

- **Pulse Manage Widgets** - for creating user widget
- **Pulse Manage Templates and Default Dashboard** - for creating default widget

Available response representations:

- 200 - New widget was successfully created, redirect to newly created widget.
- 400 - Provided widget configuration is not valid, application/json with details:
  - { "message": "WIDGET\_PARSE\_ERROR" } - widget has invalid format
- 403 - In cases when user does not have **Pulse Manage Widgets** and tries to create user widget or **Pulse Manage Templates and Default Dashboard** and tries to create default widget
- 404 - Widget configuration contains nonexistent data, application/json with details:
  - { "message": "LAYOUT\_NOT\_FOUND" } - widget layout does not exist

- **/api/wbrt/widgets/<id>**

### **[+] GET**

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

Required permission:

- **Pulse View Dashboard**

Available response representations:

- 200 - application/json with widget configuration
- 403 - In cases when user does not have **Pulse View Dashboard**
- 404 - Widget with requested ID is not found, application/json with details:
  - { "message": "WIDGET\_NOT\_FOUND" }

### **[+] PUT**

Update widget configuration for specified widget.

Example request URI(s):

- /api/wbrt/widgets/11?default=true
- /api/wbrt/widgets/12?default=false
- /api/wbrt/widgets/10

Request query parameters

Name	Required	Type	Description
default	not	string	If specified and not "false" then default widget are updated.

**Note:** owner\_user\_name field is ignored; widget is updated for current user or default widget.

Available request representations:

- application/json with widget configuration

Required permission:

- **Pulse Manage Widgets** - for updating user widget
- **Pulse Manage Templates and Default Dashboard** - for updating default widget

Available response representations:

- 200 - Widget was successfully updated
- 400 - Provided widget configuration is not valid, application/json with details:
  - { "message": "WIDGET\_PARSE\_ERROR" } - widget has invalid format
  - { "message": "INVALID\_URL" } - widget ID or default parameter in URL does not correspond to ID in the body or widget's owner.
  - { "message": "INVALID\_LAYOUT\_HASH" } - specified layout has different body\_hash than provided in snapshot

- 403 - In cases when user does not have **Pulse Manage Widgets** and tries to update user widget or **Pulse Manage Templates and Default Dashboard** and tries to update default widget
- 404 - Widget configuration contains nonexistent data or widget itself does not exist, application/json with details:
  - { "message": "WIDGET\_NOT\_FOUND" } - specified widget does not exist
  - { "message": "LAYOUT\_NOT\_FOUND" } - widget layout does not exist

## [+] DELETE

Delete widget with specified ID.

Request query parameters

Name	Required	Type	Description
default	not	string	If specified and not "false" then default widget are deleted.

Required permission:

- **Pulse Manage Widgets** - for deleting user widget
- **Pulse Manage Templates and Default Dashboard** - for deleting default widget

Available response representations:

- 200 - Widget was deleted
- 204 - Widget was already deleted
- 403 - In cases when user does not have **Pulse Manage Widgets** and tries to delete user widget or **Pulse Manage Templates and Default Dashboard** and tries to delete default widget

- **/api/wbrt/dashboards**

## [+] GET

Returns list of user's dashboard.

Required permission:

- **Pulse View Dashboard**

Available response representations:

- 200 - application/json with list of user's dashboards

**Response body example:**

```
[
  {
    "name": "tab1413974752841",
    "idx": 0,
    "title": "My Dashboard"
  },
  {
    "name": "tab1414522307471",
    "idx": 1,
    "title": "New Dashboard"
  }
]
```

- 403 - In cases when user does not have **Pulse View Dashboard**
- 503 - application/json with malfunction details:
  - { "message": "CFG\_SERVER\_ERROR" } - no connection to conf server

- **/api/wbrt/dashboards/<id>/widgets**

**[+] GET**

Returns list of widgets placed on the dashboard with specified ID.

**ID** is a dashboard **Name** or **default** if you want to get widgets from the default dashboard.

Example request URI(s):

- /api/wbrt/dashboards/tab123456789/widgets
- /api/wbrt/dashboards/default/widgets

Required permission:

- **Pulse View Dashboard**

Available response representations:

- 200 - application/json with list of widgets
- 403 - In cases when user does not have **Pulse View Dashboard**
- 404 - application/json with details:
  - { "message": "DASHBOARD\_NOT\_FOUND" } - specified dashboard does not exist
- 503 - application/json with malfunction details:
  - { "message": "CFG\_SERVER\_ERROR" } - no connection to conf server

## [+] POST

Put widget on the dashboard with specified ID.

**ID** is a dashboard **Name** or **default** if you want to put widget on the default dashboard.

Example request URI(s):

- /api/wbrt/dashboards/tab123456789/widgets
- /api/wbrt/dashboards/default/widgets

Required permission:

- **Pulse Manage Widgets** - for putting widget to user dashboard
- **Pulse Manage Templates and Default Dashboard** - for putting widget to the default dashboard

Available request representations:

- application/json with dashboard widget configuration

## Request Body Example

```
[{
  "widgetId": 683, //widget id to be placed
  "col": 1, //optional, desired column
  "row": 2 //optional, desired row
}]
```

Available response representations:

- 201 - Widget successfully placed to the dashboard
  - 403 - In cases when user does not have **Pulse Manage Widgets** and tries to put widget on the user dashboard or **Pulse Manage Templates and Default Dashboard** and tries to put widget on the default dashboard
  - 404 - application/json with details:
    - { "message": "WIDGET\_NOT\_FOUND" } - specified widget does not exist
    - { "message": "DASHBOARD\_NOT\_FOUND" } - specified dashboard does not exist
  - 503 - application/json with malfunction details:
    - { "message": "CFG\_SERVER\_ERROR" } - no connection to configuration server
- **/api/plugins/wbrt/health**

### [+] 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 are available



- { "message": "COLLECTOR\_DOES\_NOT\_RESPOND" } - collector stopped updating heartbeat file
- { "message": "DATABASE\_ERROR" } - no connection to database
- { "message": "CFG\_SERVER\_ERROR" } - no connection to conf server

## Examples: Generic Layout Definition

### Generic Layout Definition Example

```
[{
  "definition": {
    "tenant_dbid": 1,
    "refresh_interval": 15,
    "layout_type": "ltGENERIC",
    "column": [
      {
        "category": "ccDIMENSION",
        "is_delta_key": true,
        "id": "_Object$ID"
      },
      {
        "category": "ccDIMENSION",
        "is_delta_key": true,
        "id": "_Object$Type"
      },
      {
        "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"
        }
    ],
    "id": -1,
```

```
        "size": "1x4",
        "label": "Test  Widget"
    },
    "name": "Third party source",
    "description": "Example of third party source"
},
"id": 3,
"state": {
    "requested_status": "stACTIVE",
    "body_hash_1": 583854940,
    "body_hash_2": 583854940
}
}]
```

## Layout Snapshot

### Layout Snapshot Example

```
[{
  "layout_id": 3,
  "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": [
    "Agent",
    "Agent",
    "Agent",
    "Agent",
    "Agent",
    "Agent",
    "Agent",
    "Agent",
    "Agent",
    "Agent"
  ],
  "col": {
    "category": "ccDIMENSION",
    "vt": "vSTR",
    "id": "_Object$Type"
  }
},
{
  "v": [
    "Agent1",
    "Agent2",
    "Agent3",
    "Agent4",
    "Agent5",
    "Agent6",
    "Agent7",
    "Agent8",
    "Agent9",
    "Agent10"
  ],
  "col": {
    "category": "ccDIMENSION",
    "vt": "vSTR",
    "id": "_Object$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": 1408098466
}]
```

## Widget

### Widget Example

```
[{
    "id": 2,
    "layout_id": 3,
    "label": "MyWidget",
    "size": "1x4",
    "owner_user_name": "default",
    "view": [{
```

```
        "type": "BarView",
        "column_selector": ["Login_Duration"],
        "sorting": [{
            "is_asc": false
        }]
    }
}
```

# Starting and Stopping Pulse

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 Pulse Collector must be running.
- d. The RDBMS for the 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.



# Starting and Stopping Pulse Collector

You can start Pulse Collector on a Windows or UNIX platform. Invoking 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 Pulse Collector.

## Prerequisites

The following must be running prior to starting Pulse Collector:

- **Backup Configuration Server**

To restart the Pulse Collector application when the backup Configuration Server switches to the primary mode, you must specify the backup Configuration Server parameters when starting 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, Pulse Collector determines which multibyte encoding is used by Configuration Server. You can allow 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 one of the following command-line parameters:

- `-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](https://msdn.microsoft.com/en-us/library/windows/desktop/dd373814(v=vs.85).aspx) If the parameter is not

specified, Pulse Collector assumes Configuration Server uses the code page that corresponds to the locale of the Windows operating system where Pulse Collector is running.

- `-cs_encoding`. It accepts Microsoft-compatible symbolic name of locale and results choice of default code page for that locale. For more information about Microsoft-compatible locale names, see <http://msdn.microsoft.com/en-us/library/dd373814.aspx>

### 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](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 Pulse Collector

### Starting Pulse Collector

You can start Pulse Collector on any of the supported platforms.

## Solution Control Interface (SCI)

You can start Pulse Collector:

### [+] From SCI.

To start Pulse Collector from the Solution Control Interface (SCI):

1. From the `Applications` view, select your 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 Pulse Collector application starts.

For information about how to use SCI, refer to *Framework 8.0 Solution Control Interface Help*.

## Windows

On a Windows platform, you can start Pulse Collector:

### [+] Manually from the Programs menu as an application.

To start 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 Pulse Collector. The Pulse Collector application name, version, and connectivity parameters appear in the title bar.

### [+] Manually from a console window as an application.

1. To start Pulse Collector as an application from a console window: At the command-line prompt, go to the directory where Pulse Collector has been installed. Type the name of the Pulse Collector executable followed by the appropriate command-line parameters using the following syntax:

```
pulse_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 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 Pulse Collector with parameters specifying the host as `cs-host`, port as 2020, and name as `Pulse Coll`, type:

```
pulse_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`.

</ol>

## [+] 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 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 Pulse Collector.

## UNIX Platforms

You can start Pulse Collector:

### [+] Manually from UNIX Platforms.

1. Go to the directory where Pulse Collector has been installed.

#### Important



You can invoke Pulse Collector only from the directory in which it was installed.

2. Type the name of the Pulse Collector executable followed by the appropriate command-line parameters using the following syntax:  
`./pulse_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 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 Pulse Collector with parameters specifying the host as `cs-host`, port as `2020`, and name as `Pulse Coll`, type:

```
./pulse_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 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 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 Pulse Collector output (on most UNIX shells), use the following syntax:

```
./pulse_collector -host hostname -port portno -app appl >  
log_file.log
```

To have both log file and console, within Genesys Administrator add the following to 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`.

Pulse Collector writes messages to `<log_file_name.log>` in the same directory where Pulse Collector is installed. To have Pulse Collector write to a different location, specify the full path for this parameter.

## End of procedure

## Stop Pulse Collector

### Stopping Pulse Collector

You can stop Pulse Collector on any of the supported platforms.

#### Important



Be sure that the autorestart property is cleared for the Pulse Collector Application object in Genesys Administrator.

### Solution Control Interface (SCI)

You can stop Pulse Collector:

#### [+] From SCI.

If you use LCA and Solution Control Server (SCS), you can stop Pulse Collector from SCI. To do so:

1. From the `Applications` view, select your 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 Pulse Collector:

- **[+] Manually from its console window as an application.**

If 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 Pulse Collector as a Windows service, you should stop it only from the Microsoft Management Console.

To stop 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 Pulse Collector service from the list and select `Stop`.
- If LCA and SCS are used, you can stop Pulse Collector from SCI.

## On a Unix platform

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

- On the command line, type `kill -9 processid`, where `processid` is Pulse Collector's UNIX process ID.
- Press `^C` from the active Pulse Collector console.
- If LCA and SCS are used, you can stop Pulse Collector from SCI.



## How do I capture Pulse Collector memory dumps?

This information will be useful when you need to:

- Take memory dump from a running Collector process
- Configure operating system for automatically generating crash dumps so that when Collector crashes you get a crash dump

### Taking a Memory Dump of the Running Pulse Collector Process on Linux

1. Open Linux terminal.
2. Type **gcore** in the terminal window to confirm you have the **GCore** utility installed. If it is missing, you must install it. **GCore** is part of the **gdb** package.
  - a. Ubuntu: **sudo apt-get install gdb**
  - b. RHEL, CentOS: **sudo yum install gdb**
3. Determine the process ID of Pulse Collector: **ps -ef | grep collector**
4. Change to the directory where you want to store the dump: **cd ~/memory\_dumps**
5. Run commands **gcore <PID>** where <PID> is process ID
6. You should get file **core.<PID>** .
7. If you need to submit this core dump file to Genesys:
  - a. Compress it using BZip2 or any method that has a better compression ratio:
    - i. with bzip2: **bzip2 -9 core.<PID>**
    - ii. or with gzip: **gzip -9 core.<PID>**
  - b. Submit your compressed file **core.<PID>.bz2** (or **core.<PID>.gz** ) to the location specified by Genesys Customer Care.

### Taking a Memory Dump of the Running Pulse Collector Process on Windows

1. Make sure you have ProcDump utility. If not, complete following steps
  - a. Download freeware **SysInternals ProcDump** utility from this page <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 column PID. If not:
  - a. Choose menu item **View->Select Columns...**
  - b. Check item called **PID (process identifier)** in the list

4. In the Windows Task Manager, press button "**Show processes from all users**"
5. In the Windows Task manager, sort processes by process name and find appropriate **collector.exe** and notice its PID
6. Open command prompt
7. Change current directory to the location where you plan to store the dump. For example: **cd /D D:\MemoryDumps**
8. Type following command: **procdump -ma -o <PID> collector.<PID>.dmp** where <PID> is process ID of Collector
9. You will get memory dump file named as **collector.<PID>.dmp** .
10. If you need to submit this memory dump file to Genesys:
  - a. Install freeware **7-Zip** archiver if you don't already have it. You may download from this web site <http://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.

## Tuning Linux Operating System to Generate a Core Dump for the Pulse Collector in the Case it has Crashed

1. Create new startup script **/etc/init.d/core\_settings** with following contents (replace path to real path where you want to store core files).

```
#!/bin/bash
echo /path/to/core/files/storage>/core.%e.%p.%s.%t > /proc/
sys/kernel/core_pattern
```

You can use command like this to do that:

**sudo vim /etc/init.d/core\_settings**

or

**sudo nano /etc/init.d/core\_settings**

2. Adjust file privileges:  
**sudo chmod 0755 /etc/init.d/core\_settings**
3. Create link to it in the startup folders:  
**sudo ln -s /etc/init.d/core\_settings /etc/rc3.d/S01core\_settings**
4. Apply settings immediately:  
**sudo /etc/init.d/core\_settings**
5. Enable core dumps globally: edit file **/etc/security/limits.conf**, add or uncomment (if already present) the following line  
**\* soft core unlimited**
6. Edit **/etc/profile** and **/etc/init.d/functions** - add or uncomment following lines:

In **/etc/profile** (Redhat)

```
# No core files by default
ulimit -S -c 0 > /dev/null 2>&1
```

In **/etc/init.d/functions** (Redhat)

```
# make sure it doesn't core dump anywhere unless requested
ulimit -S -c ${DAEMON_COREFILE_LIMIT:-0} >/dev/null 2>&1
```

7. Restart the system
8. Now core files with names formatted as **core.<executable name>.<pid>.<signal #>.<timestamp>** will appear in the specified folder.
9. If you need to submit this core dump file to Genesys:
  - a. Compress it (BZip2 is recommended, as long as gives better compression ratio):
    - i. with bzip2: **bzip2 -9 <strong>core.<executable name>.<pid>.<signal #>.<timestamp> </strong>**
    - ii. or with gzip: **gzip -9 <strong>core.<executable name>.<pid>.<signal #>.<timestamp> </strong>**
  - b. Submit resulting file **core.<executable name>.<pid>.<signal #>.<timestamp> .bz2** (or **core.<executable name>.<pid>.<signal #>.<timestamp> .gz** ) to the location specified by the Genesys Technical Support.

## Tuning Windows 2008/2012 Operating System to Generate a Crash Dump for the Pulse Collector in the Case it has Crashed

1. Open Notepad text editor
-

2. Enter there following text, replacing value of the **DumpFolder** parameter with real path where you want to store Collector crash dumps:

```
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 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 Collector crashes, full memory dump of Collector will appear in the specified folder.
6. 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 <http://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 7-Zip Archive file to the location specified by the Genesys Technical Support.

## Suggested Additional Reading

1. WER Settings (Windows) [https://msdn.microsoft.com/en-us/library/windows/desktop/bb513638\(v=vs.85\).aspx](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:

- 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.**