



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

# Deployment Guide

Genesys Web Engagement 8.5.0

# Table of Contents

<b>Genesys Web Engagement Deployment Guide</b>	<b>4</b>
<b>What is Genesys Web Engagement?</b>	<b>6</b>
<b>Multi-Tenancy</b>	<b>12</b>
<b>Deployment Scenarios</b>	<b>13</b>
<b>Prerequisites</b>	<b>16</b>
Sizing Information	19
<b>Installing the Genesys Web Engagement Server</b>	<b>24</b>
Configuring Genesys Rules System	43
Configuring External Cassandra	51
Automatic Provisioning	67
Tuning Your JVM	73
<b>Installing the Plug-in for Workspace Desktop Edition</b>	<b>75</b>
<b>Installing the Plug-in for Genesys Administrator Extension</b>	<b>80</b>
Tuning Role-Based Access in Genesys Administrator	83
<b>Installing the Reporting Dashboard Plug-in (optional)</b>	<b>91</b>
<b>Reporting</b>	<b>94</b>
Installing Genesys Data Processing Server	95
Using Web Engagement with Data Processing Server	119
Installing the Web Engagement Reporting Server	123
<b>Security</b>	<b>149</b>
Security Tips for Third-Party Components	150
Secure Connections to HTTP Clients	152
Transport Layer Security (TLS) with Genesys Servers	155
Authentication	162
Cassandra Security	165
Protecting Personally Identifiable Information	174
<b>Configuring Specific Features</b>	<b>177</b>
Pacing Algorithm	178
Chat Channel	188
Web Callback Channel	193
GeolP Information	197
<b>Monitoring Web Engagement Server</b>	<b>198</b>
View Metrics with JMX	206
<b>Load Balancing</b>	<b>211</b>
<b>Configuration Options</b>	<b>217</b>

cassandraEmbedded Section	218
cassandraKeyspace Section	225
elasticsearch Section	228
privacy Section	229
cep Section	230
chat Section	231
cometd Section	233
engagement Section	235
kibana Section	237
metrics Section	238
pacing Section	241
queues Section	243
userData Section	245
webcallback Section	247
log Section	248
security Section	253
web Section	255

# Genesys Web Engagement Deployment Guide

Welcome to the Genesys Web Engagement 8.5 Deployment Guide. This document introduces you to the concepts, terminology, and procedures relevant to Genesys Web Engagement. See the summary of chapters below.

## Getting Started

Find information to help plan your Genesys Web Engagement Deployment.

---

- [What is Genesys Web Engagement?](#)
- [Deployment Scenarios](#)
- [Prerequisites](#)
- [Sizing](#)

## Installing GWE in a Lab

Find procedures to install and configure Genesys Web Engagement in a lab environment.

---

- [Lab Deployment Scenario](#)
- [Installing Genesys Web Engagement](#)
- [Automatic Provisioning](#)

## Installing the GWE Plug-ins

Find procedures to install and configure the Genesys Web Engagement plug-ins.

---

- [Installing the Plug-in for Workspace Desktop Edition](#)
- [Installing the Plug-in for Genesys Administrator Extension](#)

## Deploying GWE in Production

Find procedures to deploy and configure Genesys Web Engagement in a production environment.

---

- [Production Deployment Scenario](#)
- [Load Balancer Sample Configurations](#)
- [External Cassandra Settings](#)

## Configuring Specific Features

You can configure these features to

enable their functionality:

---

A Pacing Algorithm

A Chat Channel

A Web Callback Channel

Access to GeolP Information

# What is Genesys Web Engagement?

If you're just getting started with Web Engagement, check out [A First Look at Genesys Web Engagement](#).

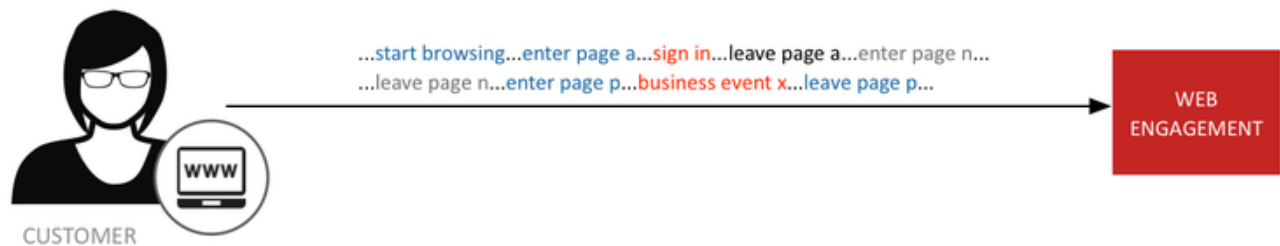
## Overview

Genesys Web Engagement enhances customer experience by bringing together three related technologies:

- **Monitoring** records the customer's web browsing activity.
- **Notification** lets a customer know about an offer to engage via chat or web callback so they can accept it or decline it.
- **Engagement** provides the actual chat or web callback window to the customer's browser.

Here's how they work together:

## Monitoring

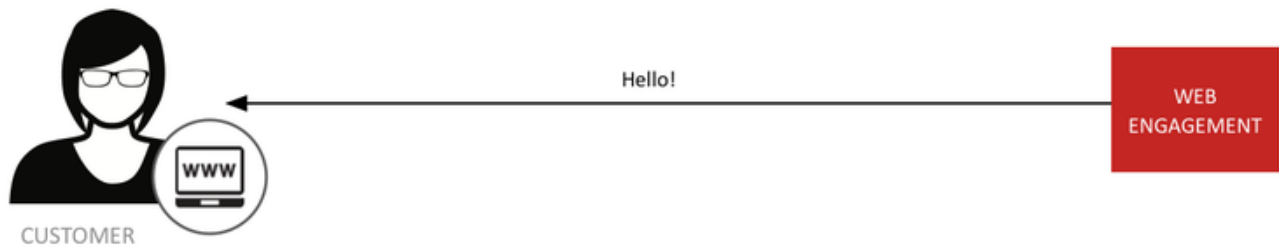


**Monitoring** allows you to gather data on customer browsing behavior so you can analyze it.

For example, if you've noticed that a lot of customers reach a certain point on your site and then abandon their transactions, you may want to offer a chat as soon as other people reach that point.

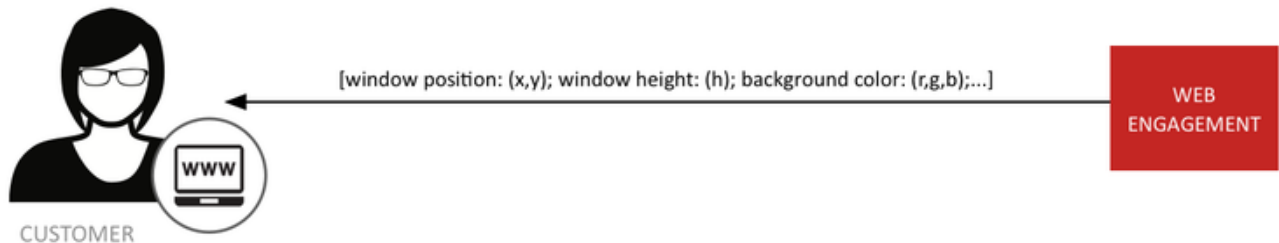
Another case might be where someone has been looking at a certain type of product, and you want your agents to know what it was, so they can respond to the customer's queries more intelligently.

## Notification



While the concepts behind **notification** are simple, the power it gives you can lead to a lower abandon rate and higher levels of customer satisfaction—all because Web Engagement provides easy-to-use JavaScripts that allow you to reach out to those customers who need help.

## Engagement



**Engagement** uses simple JavaScript technologies to send chat or web callback windows to the appropriate customers.

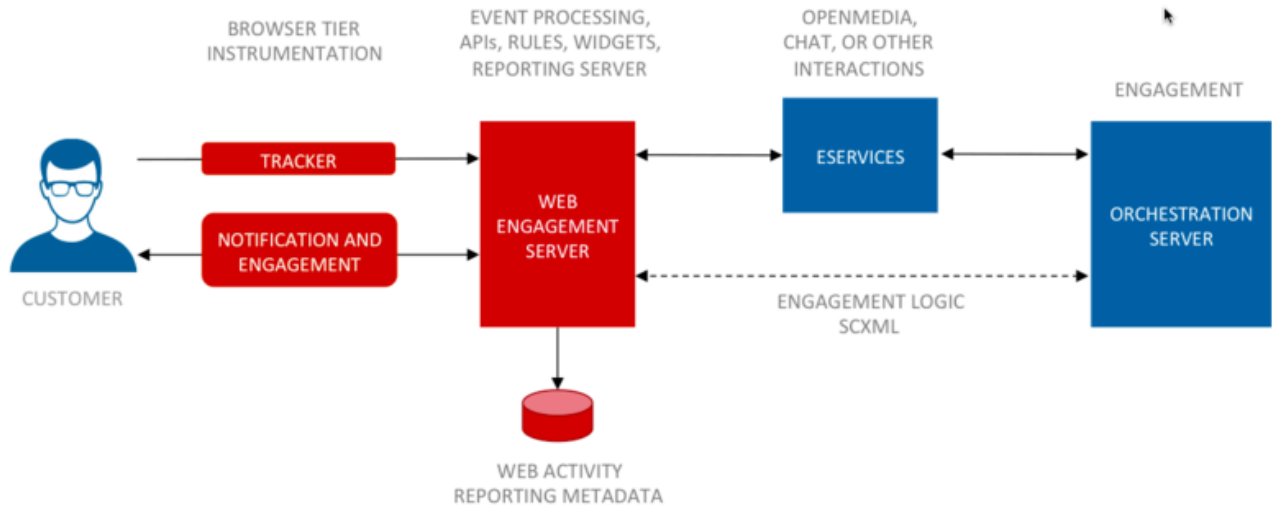
## Basic Architecture

### Browser Tier

Each of these three functions has an agent in the browser tier, as shown in the diagram below.

The browser tier connects to a load balancer that in turn communicates with a cluster of Web Engagement Servers. This cluster is connected to the rest of the Genesys solution, as well as to the Web Engagement database, which is frequently located in the Enterprise Tier.

# What is Genesys Web Engagement?

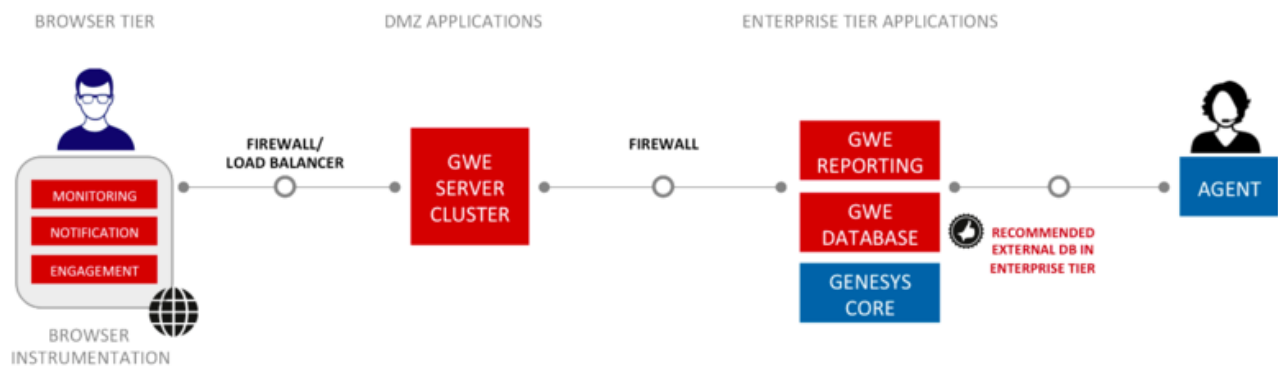


## Enterprise Tier

Let's take a closer look at this architecture, with a somewhat sharper focus on the enterprise tier.

Web Engagement uses Cassandra database storage and Elasticsearch as an indexing engine. These two components work closely together to provide high-performance access to large amounts of data, both for operational needs and as input to your reporting. As mentioned, the database is often housed in the enterprise tier, but these components can live anywhere within reach of the Web Engagement Server cluster.

The rest of the enterprise tier consists of the various Genesys components that work together with Web Engagement, backing it up with the full power of the Genesys CX solution.



## The Web Engagement Tier

The Web Engagement Cluster is built on top of an N + 1 architecture to support an easily extendable



set of Web Engagement Servers. These servers facilitate a complicated set of services that bring together the power of Web Engagement. Let's take a look at some of the most important ones.

## Events, Categories, and Rules

### System Events

The basic goal of Web Engagement is to help you decide when to invite a customer to engage with your agents. You can use Web Engagement's six **System Events** to help with this. These events are:

- **VisitStarted**—Includes the URL the customer first visited on your site and the time they started the visit
- **PageEntered**—Includes the URL and time when a customer entered a page
- **PageExited**—Includes the URL and time when a customer exited a page
- **UserInfo**—Reflects that a visitor is recognized, but not signed in
- **SignIn**—Reflects that the user has signed in
- **SignOut**—Reflects that the user has signed out

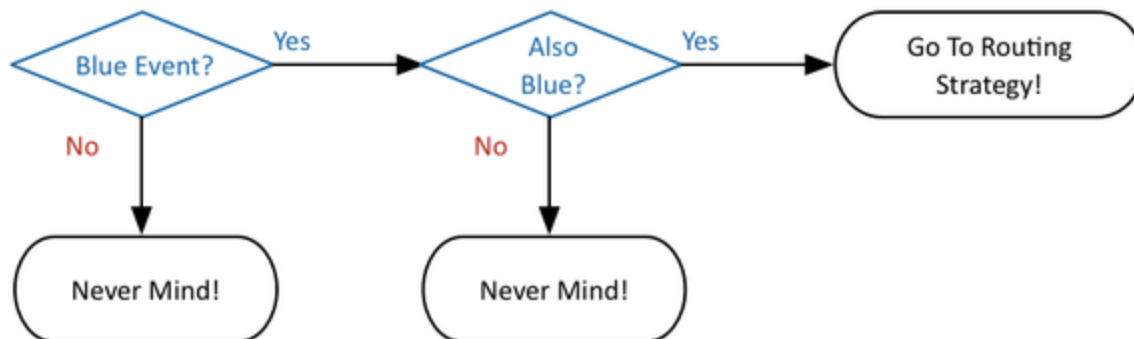
Sometimes you just need to know how long someone has been on your site. But in many situations, you are looking for more complicated patterns. These six events give you a lot to work with.

### Categories

<u>Electronics</u>	<u>ExpensivePurchase</u>	<u>ContactUs</u>	<u>None</u>
PageEntered	AddToCart	PageEntered	
Timeout-30	RemoveFromCart	Timeout-30	
PageExited		LocationMapClick	
		PageExited	

And then there are the times when you need to pay attention to certain *types* of activity on your site. For example, you may be having a special on 40-inch HDTVs, so you want to keep track of customers who are visiting the pages for 40-inch models. You can set up a **Category** in Genesys Administrator that allows you to tag the **PageEntered** events for these pages so you can track them.

## Rules



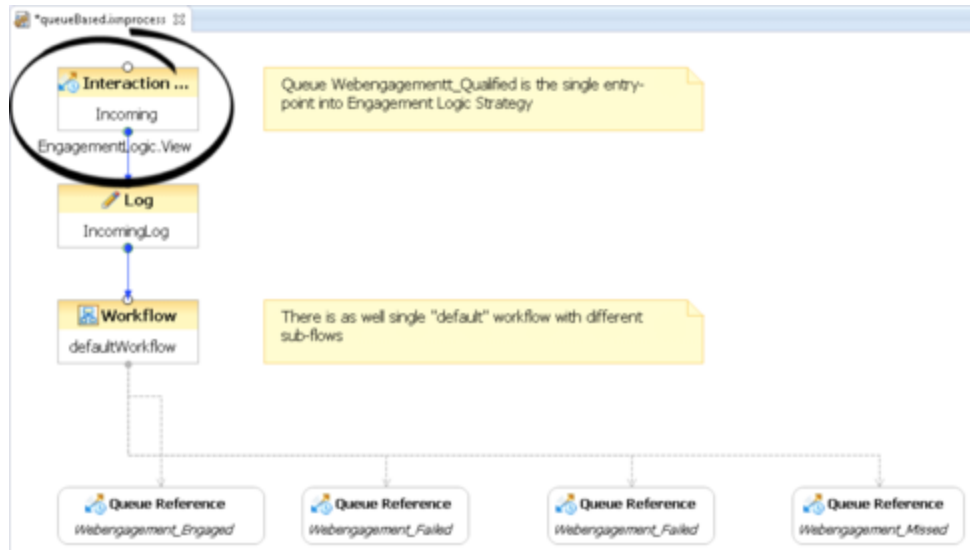
Once you know which system events and categories you're interested in, you need to write some logic that decides what to do with the information you have gathered about them. This logic is known as a **Rule**.

## Business Events

In some cases, the system events and categories don't tell you everything you need to know. In those situations, you may want to define your own **Business Events**. As with system events, you can often get a lot more out of your business events by using them in conjunction with categories.

## Routing and Pacing

### Routing Strategies



Having identified your events and categories, and after writing your rules, you can finally decide who to engage. Interactions for these customers will be processed by the appropriate **Routing Strategy**.

### Pacing

When you start sending out chat or callback invitations, you need to make sure you don't send more than your agents can handle. Web Engagement includes intelligent **pacing** technology to help you optimize the number of invitations you send at any given time.

## Web Engagement Applications

To make all of these features work together, you must develop a Web Engagement **application**, so that your website can start using the power of Genesys Web Engagement.

**Important:** Before using Genesys Web Engagement, you must either **create** and **deploy** your own Web Engagement application or deploy the provided sample application. Otherwise, the product will not work.

# Multi-Tenancy

You can create one Genesys Web Engagement cluster per tenant.

# Deployment Scenarios

Genesys Web Engagement has two flavors of deployment: the simplest is appropriate for a lab environment, while a production environment requires solution that is a bit more complex. Select the appropriate tab below for details about each deployment scenario and the tasks to install and configure GWE.

## Important

Genesys strongly recommends that you first install Genesys Web Engagement in a lab configuration, which will make it easier to start working with your GWE application.

## Lab

### Overview

This deployment is appropriate for a lab environment and consists of a single Web Engagement Cluster and a single Web Engagement Server node—they can be installed together on the same host.



### Deployment Tasks

Complete the following tasks to deploy Genesys Web Engagement:

1. **Review the prerequisites.** Make sure your planned environment meets the requirements for Genesys Web Engagement and contains the right versions of the required Genesys components.
2. **Install the Web Engagement servers.** Complete these procedures to configure and install the cluster and node.
3. **Configure Genesys Rules System.** You need to configure Genesys Rules Authoring Tool and Genesys Rules Development Tool to work with GWE.
4. **Install the Plug-in for Workspace Desktop Edition.** You will need this plug-in to enable chat and web callback engagement features in Workspace Desktop Edition.

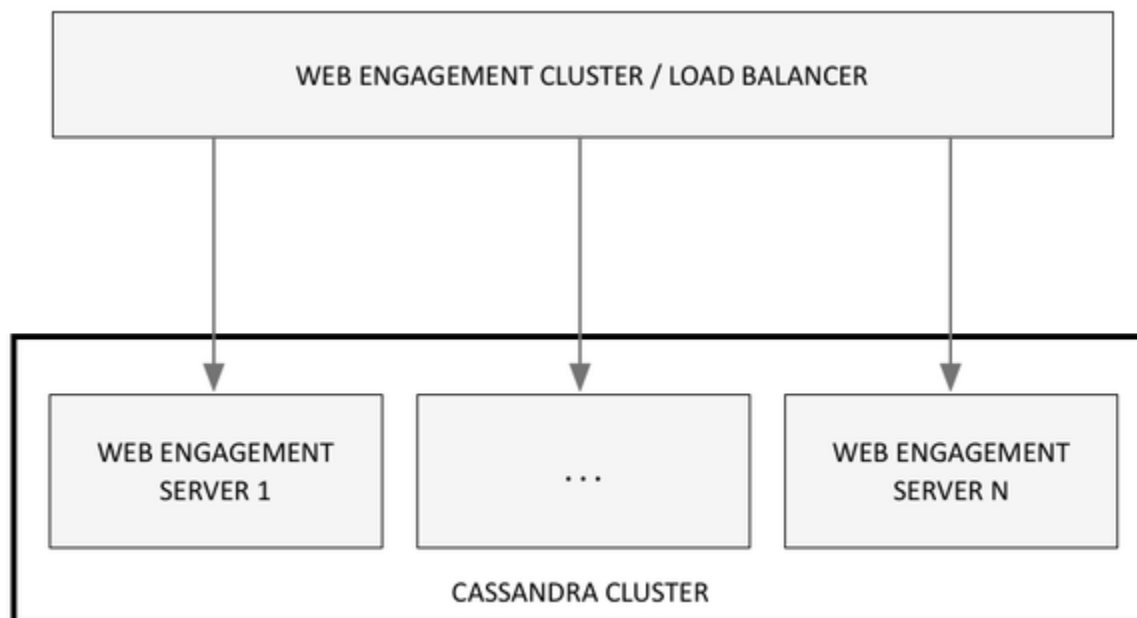
5. **Install the Plug-in for Genesys Administrator Extension.** You will need this plug-in to create categories for your application and to generate the instrumentation script snippet that needs to be placed on your website.
6. **Configure your required security.** You can enable SSL, configure TLS for Genesys and GWE servers, and enable authentication for the Web Engagement Server, Workspace Desktop Edition, and the Engagement Strategy.
7. **Configure your required features.** You can set up the pacing algorithm, enable authentication and SSL, and more.
8. **Develop an application.** You're now ready to move on to the Developer's Guide, where you will learn how to develop and customize a Web Engagement application, not to mention being able to create categories, business events, and rules.

## Production

### Overview

This deployment scenario is appropriate for a production environment and consists of a single Web Engagement Cluster and multiple Web Engagement Server nodes. Ideally, these servers should be installed on separate hosts to provide the best performance and the best high availability.

The figure below shows the **minimum solution size** Genesys recommends for a production deployment.



### Important

If you use embedded Cassandra, you should plan to host your Web Engagement Servers in a secure zone, along with your Chat Server(s), in order to protect your data. The Web Engagement Servers do require internet access for chat traffic, but this can be solved by using a reverse proxy, which is a standard function of most load balancers.

## Deployment Tasks

Once you have thoroughly explored Web Engagement's features in your lab environment, you can complete the following tasks to deploy Genesys Web Engagement in production:

1. **Review the prerequisites.** Make sure your planned environment meets the requirements for Genesys Web Engagement and contains the right versions of the required Genesys components.
2. **Install the Web Engagement servers.** Complete these procedures to configure and install the cluster and nodes.
3. **Configure Load Balancing.** This guide includes sample configurations for Apache and Nginx, although your settings may be different depending on the architecture of your cluster.
4. **Configure External Cassandra.** Genesys recommends that you use an external Cassandra ring in your production environment. This article discusses how to do that.
5. **Enable SSL.** You need to use a certificate issued by a third-party Certificate Authority for a production environment.
6. **Configure your other required security.** You can also configure TLS for Genesys and GWE servers, and enable authentication for the Web Engagement Server, Workspace Desktop Edition, and the Engagement Strategy.
7. **Configure Genesys Rules System.** You need to configure Genesys Rules Authoring Tool and Genesys Rules Development Tool to work with GWE.
8. **Configure your required features.** You can set up the pacing algorithm, enable authentication and SSL, and more.
9. **Install the Plug-in for Workspace Desktop Edition.** You will need this plug-in to enable chat and web callback engagement features in Workspace Desktop Edition.
10. **Install the Plug-in for Genesys Administrator Extension.** You will need this plug-in to create categories for your application and enable GWE on your website.
11. **Develop an application.** You're now ready to develop and customize your production Web Engagement application.

# Prerequisites

## Hardware Requirements

See [Sizing](#) for details.

## OS Requirements

See the Operating Systems section of the [Genesys Web Engagement](#) page in the *Genesys Supported Operating Environment Reference Guide* for more information on supported operating systems.

## Browser Support

### Web Browsers

See the Browsers section of the [Genesys Web Engagement](#) page in the *Genesys Supported Operating Environment Reference Guide* for supported browsers.

The following previously supported browsers, although tested, have been **discontinued** and are no longer supported by Genesys:

- Safari 3 and 5
- Firefox 3, 4, and all non-ESR versions
- Internet Explorer 7, 8, 9, and 10

### Mobile Browsers

- iOS Safari
- Android Chrome

### Known Limitations

The Playground application included in the Genesys Web Engagement solution does not support Internet Explorer 8.



## Java Requirements

For Java prerequisites, see the Prerequisites section of the [Genesys Web Engagement](#) page in the [Genesys Supported Operating Environment Reference Guide](#) for more detailed information and a list of all prerequisites.

## Genesys Environment

In addition to having a [Genesys Management Framework](#) environment installed and running, the following table lists the **mandatory** Genesys components that are used with a Genesys Web Engagement installation.

### Mandatory Components

Component Name	Minimum Compliant Version
Orchestration Server	8.1.300.55
Stat Server	8.5.000.17
Interaction Server	8.5.100.18
Chat Server	8.1.001.41
Genesys Rules Authoring Server	8.5.200.00
Configuration Server (for UTF-8 support)	8.1.200.17

**Note:** Web Engagement supports a connection to the [Interaction Server Proxy](#) as an alternative to a direct connection to Interaction Server.

### Components for Application Development

The following components are mandatory to create customized Genesys Web Engagement applications:

Component Name	Minimum Compliant Version	Details
Genesys Rules Authoring Tool	8.5.303.08	<p>You must create a user with Roles Privileges that enable the creation of Rules package in Genesys Rules Authoring. See <a href="#">Role-Based Access Control</a> in the Genesys Rules System Deployment Guide.</p> <p><b>Note:</b> Genesys Web Engagement only supports GRAT versions <b>8.5.303.08</b> through <b>8.5.303.14</b>.</p>
Genesys Rules Development Tool	8.1.400.05	This component must be deployed as Composer plug-in or independently in Eclipse; make

## Prerequisites

---

Component Name	Minimum Compliant Version	Details
		sure that settings are correct for Configuration Server and Repository Server, as detailed in <a href="#">Installing the GRDT Component</a> in the Genesys Rules System Deployment Guide.
Composer	8.1.430.03	Mandatory to publish the <a href="#">Web Engagement rules template</a> . This component can also be used to update or deploy routing and engagement strategies.
Genesys Administrator Extension	8.5.210.05	Required to install the Web Engagement Plug-in for Genesys Administrator Extension. Can also be used to provision Web Engagement.
Genesys Administrator	8.1.300.02	Optional. Can be used as an alternative to Genesys Administrator Extension when provisioning Web Engagement.

## Components for Interaction Management

Component Name	Workspace Desktop Edition Plug-in Version	Minimum Compliant Version
Workspace Desktop Edition	8.5.000.42	8.5.111.12
	8.5.000.11 - 8.5.000.36	8.5.000.55

## Additional Components (Optional)

Component Name	Minimum Compliant Version
CCPulse+	8.0.000.36
Pulse	For information on Pulse support for Web Engagement and Genesys Data Processing Server, see <a href="#">Using Web Engagement with Data Processing Server</a>

---

# Sizing Information

Before deploying the Genesys Web Engagement (GWE) solution to your production site, you must estimate the size of solution that will be able to handle the expected user load. Genesys recommends that you download the **GWE Sizing Calculator**, an Excel workbook that you can use to help calculate the number of Web Engagement Server nodes required for your production deployment.

Click [HERE](#) to download the **GWE Sizing Calculator**. See download tips if the download was not started automatically.

The process of estimation starts from input values, usually given in the terms of business operations; for example, daily visit rates, or page view rates. Using some math, and having in mind the workflow that is applied to the input traffic, you can then produce the expected load values in terms of requests per second. Applying these values to the experimentally produced measurements, you can estimate the size of the solution required to be deployed.

## Important

The exact deployment architecture and solution size will vary depending on your hardware equipment, and that the deployed system can be fine-tuned to get the best performance on given equipment and with given user load. However, the estimation can give some basic ideas for the deployment.

## Input Data for Load Estimation

To estimate the load, you need to know the following data:

- Average visits rate and page views rate (per hour)
- Maximum visits rate and page views rate (per hour)

To estimate your disk space consumption, it is helpful to have the following information about the configuration of the solution:

- Number of business events per page or visit
- Portion of categorized events

You can use these numbers as inputs into the **GWE Sizing Calculator**.

---

## Basic Load Estimation

### Visits Rate Estimation

Having the maximum or average visits count and page view count per hour (or day), you can get visits rate and visit depth:

- Visits per second = Visits per hour / 3600 = Visits per Day / 24\*3600
- Visit depth = Page views per hour / Visits per hour = Page views per day / Visits per day

### Events Rate Estimation

Having the following input data about solution configuration:

- Average number of business events (timeout, search, and so on) per page – page custom events
- Average number of user events (SignIn/SignOut/UserInfo) per visit – custom visit events
- The fact that every visit produces one system event (visit started) and every page view produces two system events (PageEntered and PageExited)

The calculator can now estimate the average event rate produced by user visits:

- Events per second = Visits per second \* (custom visit events + 1) + (Visits per Seconds \* Visit Depth) \* (custom page events + 2)

### Requests Per Second

Assuming that each visit also contains two requests for loading client scripts and DSL, and every page view also contains one request for categorization info:

- Requests per second = Visits per second \* 1 + (Visits per second \* Visit depth) \* 2 + Events per second

## Peak Load Estimation

Having only average values for visit and page views, you can only predict some average load, and, consequently, only an *average* solution size that will be able to handle such load. However, the user load is not evenly distributed during the day, so you must estimate possible peak load during busy hours.

To cover that scenario, you can take your Visits as the Poisson distributed flow. To estimate the possible peak load, use the following calculation:

- Maximum Visits per Second = Average Visits per Second + 4 \* SQRT (Average Visits per Second)

Now, having the Maximum Visits per Second value, it is possible to estimate the Maximum Events per Second and the Maximum Requests per Second, using the same approach as for Basic Load, but replacing average visits rate with maximum visits rate.

---

## Load Estimation Example

### Visits Rate Estimation

Having the maximum visits and page view rates per hour, you're calculating visits rate and visit's depth. For example, having the following values:

- maximum number of visits is 3000 per hour
- maximum number of page views is 32000 per hour

The estimated visit depth is  $32000/3000 = 10.7$  pages.

- Average load: 3000 visits per hour = 0.83 visits per second
- Maximum load:  $0.83 + 4*\sqrt{0.83} = 4.5$  visits per second

### Events Rate Estimation

Given two business events per page, and about 10 percent of visits producing additional events:

- Average events per second =  $0.83*(1 + 0.1) + (0.83*10.7)*(2 + 2) = 37$  events per second

### Requests per Second

Assuming that every visit generates one request for script and another for data, and every page view generates one request for categories:

- Average requests per second =  $37 \text{ events} + 0.83*2 + (0.83*10.7)*1 = 47$  requests per second

### Peak Load Estimation

Using Maximum Visits per second instead of Average Visits per Second, the following values can be calculated:

- Maximum events per second =  $4.5*(1 + 0.1) + (4.5*10.7)*(2 + 2) = 197$  events per second
- Maximum requests per second =  $197 + 4.5*2 + (4.5*10.7)*1 = 254$  requests per second

## Minimum Solution Size

The solution deployed should handle all user input, and have some N+1 redundancy. The next two sections give the minimum solution size for external and embedded versions of Cassandra, while the third section contains information on disk space requirements.

### External Cassandra

- Web Engagement Server—two nodes (to support high availability and load balancing).

- Cassandra—three nodes (to provide data consistency and to allow a fault tolerance of one node). The consistency level must be LOCAL\_QUORUM.

For more help calculating the number of Cassandra nodes you need to support data consistency in the cluster, see <http://www.ecyrd.com/cassandrascalculator/>.

## Embedded Cassandra

The Web Engagement Server cluster should contain at least three nodes (to support high availability and load balancing, and to provide data consistency and allow a fault tolerance of one node). The cluster must have a consistency level of LOCAL\_QUORUM.

### Important

Starting in 8.5.0, Embedded Cassandra mode is deprecated in Web Engagement; support for this mode will be discontinued in 9.0.

## Disk Space Usage

Disk space usage directly depends on the following factors:

- The event rate in the solution
- The average count of the categories applied to each event
- The replication factor specified for Cassandra (and Elasticsearch)

## Glossary

*Visit*—series of page views from the same uniquely identified browser. Note that pages opened in different tabs or windows of the same browser will be treated as part of the same visit.

*Average visits per second*—service load in terms of customer site.

*Request*—a single request to the service. A single page view produces a series of requests to the Web Engagement Server:

- Loading of monitoring script
- Loading of categorization information
- Sending information about **PageEntered**, **PageExited**, and business events back to the Web Engagement Server

*Requests per second (RPS)* – actual load in terms of service performance.

## Useful Links

Refer to the following links for more information about planning an external Cassandra cluster:

- For help understanding the Cassandra architecture, see <http://docs.datastax.com/en/cassandra/2.2/cassandra/architecture/archTOC.html>
- For information about hardware considerations for Cassandra nodes, see <http://docs.datastax.com/en/cassandra/2.2/cassandra/planning/planPlanningHardware.html>
- For details about Cassandra cluster configuration, refer to <http://docs.datastax.com/en/cassandra/2.2/cassandra/initialize/initSingleDS.html>
- For more about Cassandra clusters and memory, see <http://docs.datastax.com/en/cassandra/2.2/cassandra/operations/opsTuneJVM.html>
- For more help calculating the number of Cassandra nodes you need to support data consistency in the cluster, see <http://www.ecyrd.com/cassandrascalculator/>.

---

# Installing the Genesys Web Engagement Server

The 8.5 release of Genesys Web Engagement incorporates **two major changes** in the server architecture:

- The Web Engagement Frontend Server and Backend Server have been consolidated into a single Web Engagement Server.
- In order to simplify the configuration process, **you must configure a Web Engagement Cluster application object**, even if you are only using a single node.

Each of these changes enhances the performance and the simplicity of Web Engagement.

## Important

Before using Genesys Web Engagement, you must either **create** and **deploy** your own Web Engagement application or deploy the provided sample application. Otherwise, the product will not work.

After Genesys Web Engagement is ready to use, but before you use it, Genesys recommends that you verify the status of your **Cassandra cluster** and your **Elasticsearch cluster**.

## About the Web Engagement Cluster

Genesys Web Engagement is built on the principles of N+1 architecture. This means that:

- The cluster combines 1 or more nodes. In other words, **you must create and configure at least one node in order to use Web Engagement**. And every time you **add another node**, you need to create and configure it using the same steps you used to **create** and **configure** the first one.
- All nodes are treated as equivalent to each other, so that almost all of the configuration is defined in the cluster application, while the nodes only contain the options required to connect to a particular host and their cluster.
- All server connections, such as Interaction Server and Chat Server, are defined for the cluster.  
**Note:** Web Engagement supports a connection to the **Interaction Server Proxy** as an alternative to a direct connection to Interaction Server.
- Connections to external clusters, such as a cluster of Chat Servers, must be configured in the cluster application, rather than adding connections to particular nodes. In other words, if you add nodes to the cluster, you don't have to reconfigure the existing nodes and cluster, whether you are migrating from one node to two nodes, or adding a thousand nodes to the system.

These features make it easy to maintain your cluster configuration and help you avoid faulty



configuration scenarios.

## Deploying Web Engagement

To deploy Web Engagement, follow these steps:

1. [Importing the Web Engagement Cluster Template](#)
2. [Creating the Cluster Application](#)
3. [Configuring the Cluster Application](#)
4. [Configuring a Connection to a Cluster of Chat Servers \(Optional\)](#)
5. [Importing the Web Engagement Server Template](#)
6. [Creating a Node Application](#)
7. [Configuring a Node Application](#)
8. [Adding Nodes to a Cluster](#)
9. [Installing the Web Engagement Server](#)
10. [Configuring alarms](#)

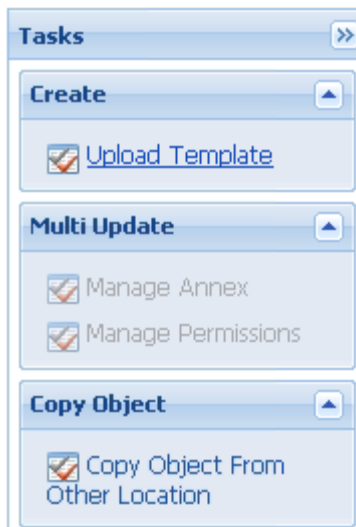
**Note:** For more information on how to work with templates and application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

## Importing the Web Engagement Cluster Template

**Note:** For more information on how to work with templates in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Application Templates**.
2. In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.
4. Browse to the **Web\_Engagement\_Cluster.apd** file. The **New Application Template** panel opens.
5. Click **Save & Close**.

**End**

## Creating the Cluster Application

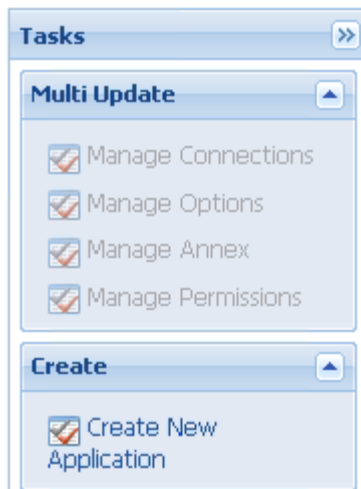
**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Importing the Web Engagement Cluster Template](#).

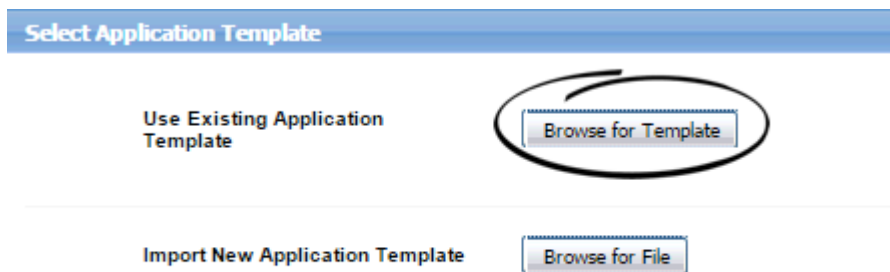
### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.



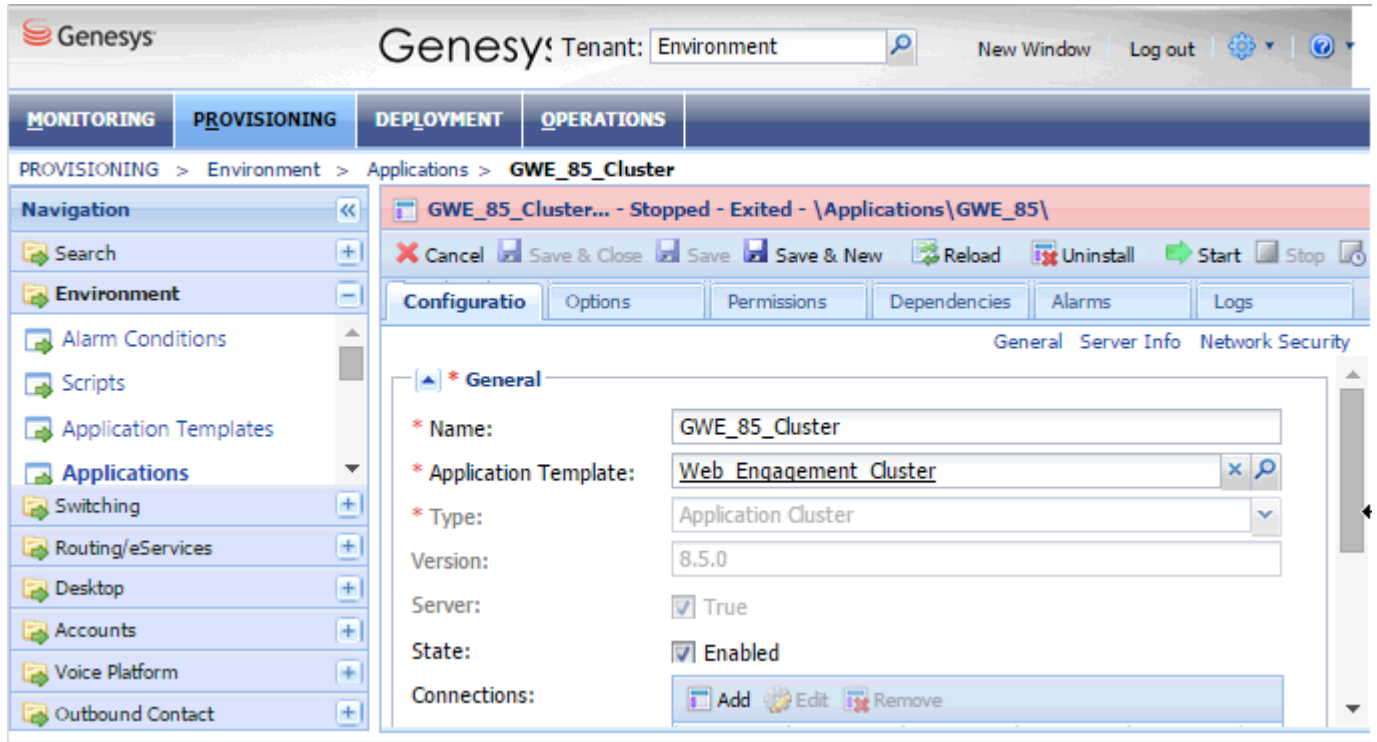
Create New Application link.

3. In the **Select Application Template** panel, click **Browse for Template**.



Browse for Template.

4. Select the Web Engagement Cluster template that you imported in [Importing the Web Engagement Cluster Template](#). Click **OK**.
5. The template is added to the **Select Application Template** panel. Click **Next**.
6. In the **Select Metadata file** panel, click **Browse**, then click **Add**, and then select the **Web\_Engagement\_Cluster.xml** file. Click **Open**.
7. The metadata file is added to the **Select Metadata file** panel. Click **Next**.
8. In the **Specify Application parameters** tab:
  - Enter a name for your application. For instance, GWE\_85\_Cluster.
  - Make sure that **State** is enabled.
  - Select the **Host** on which the Web Engagement Cluster will reside.
  - Click **Create**.
9. The **Results** panel opens.
10. Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The Web Engagement Cluster application form opens and you can start configuring the Web Engagement Cluster application.



Web Engagement Cluster app opened in Genesys Administrator.

**End**

## Configuring the Cluster Application

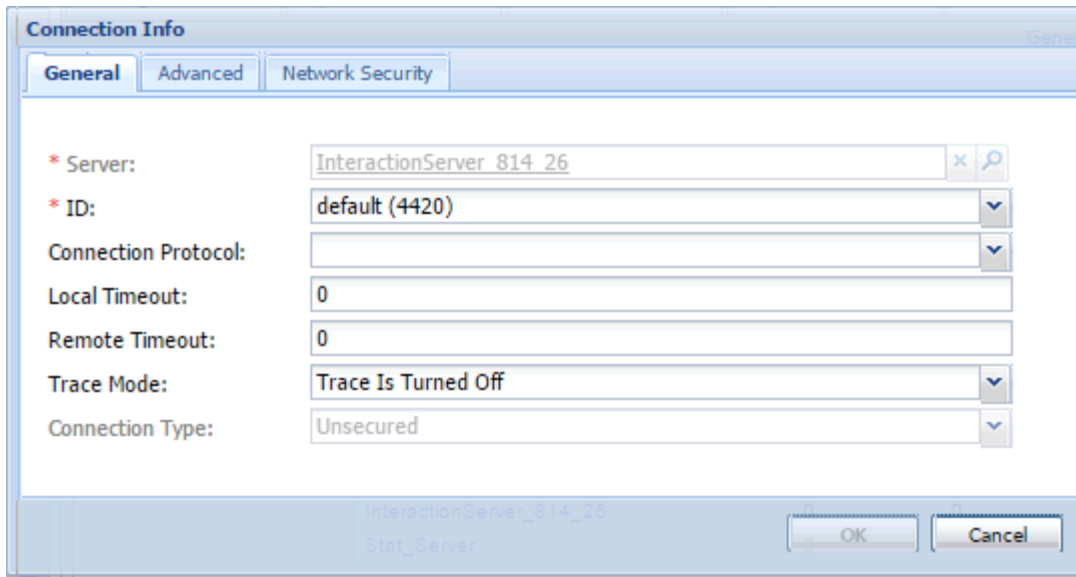
**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Creating the Cluster Application](#).

### Start

- If your Cluster application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the Web Engagement Cluster and click **Edit...**
- In the **Connections** section of the **Configuration** tab, click **Add**. The **Browse for applications** panel opens. Select the Genesys application defined for **Interaction Server**, then click **OK**.  
**Note:** Web Engagement supports a connection to the [Interaction Server Proxy](#) as an alternative to a direct connection to Interaction Server.



The port **ID** is set to the default port.

3. Repeat the previous step for **Stat Server**. Optionally, you can also add a connection to Message Server (to apply the **network logging option**).
4. Configure a connection to the Chat Server or to a cluster of Chat Servers. If you are using a single Chat Server, you must set your port ID to `webapi` when specifying your connection to Chat Server. For information on how to connect a cluster of Chat Servers, see [Configuring a Connection to a Cluster of Chat Servers \(Optional\)](#).
5. Expand the **Server Info** pane.
6. In the Tenant section, click **Add** and select your tenant. For instance, Environment. Click **OK**.
7. If your **Host** is not defined, click the lookup icon to browse to the hostname of your application.
8. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
 

**Note:** You must specify the default port.

  - Enter the **Port**. For instance, 9081.
  - Choose `http` for the **Connection Protocol**.
  - Click **OK**. The HTTP port with the default identifier appears in the list of **Listening ports**.

The screenshot shows the 'Port Info' dialog box with the 'General' tab selected. The fields are filled with the following values:

- \* ID: default
- \* Port: 9081
- Connection Protocol: http
- HA sync:  True
- Select Listening Mode: Unsecured
- Description: Default http port for the Web Engagement Cluster

At the bottom, there are 'OK' and 'Cancel' buttons. Below the dialog box, the text 'Default HTTP port' is visible.

Default HTTP port

9. Create the port with the server/data/rules/deploy identifier by clicking **Add**.
  - Enter server/data/rules/deploy for the ID.
  - Enter the same port value as you did for the default port. For instance, 9081.
  - Select http for the **Connection Protocol**.
  - Click **OK**. The HTTP port with server/data/rules/deploy ID appears in the list of Listening ports.
10. Optionally, you can add a secure listening port for authenticated users, secured connections, and secure chat. Click **Add**. The **Port Info** dialog opens.
  - Enter https for the ID field. This specific ID is required in GWE 8.5.
  - Enter the **Port**. For instance, 9443.
  - Enter https for the **Connection Protocol**.
  - Choose Secured for the **Listening Mode**.
  - Click **OK**.

**Port Info**

**General** | Advanced | Network Security

\* ID: https

\* Port: 9443

Connection Protocol: https

HA sync:  True

Select Listening Mode: Secured

Description:

OK Cancel

Secure listening port

11. Ensure the **Working Directory** and **Command Line** fields contain "." (period).

**Configuration** | Options | Permissions | Dependencies | Alarms | Logs

General Serv

\* Working Directory: .

\* Command Line: .

Command Line Arguments:

\* Startup Timeout: 90

\* Shutdown Timeout: 90

Backup Server: [Unknown Backup Server]

\* Redundancy Type: Not Specified

\* Timeout: 10

\* Attempts: 1

Auto Restart:  True

Log On As SYSTEM :  True

\* Log On Account: [Unknown Log On Account]

Commands

12. Click **Save**.
13. The **Confirmation** dialog for changing the application's port opens. Click **Yes**.
14. **Important:** Genesys recommends that you use external Cassandra, which you can configure by

---

following [these instructions](#). However, embedded Cassandra can be a better choice for a lab installation.

**Important:** Starting in 8.5.0, Embedded Cassandra mode is deprecated in Web Engagement; support for this mode will be discontinued in 9.0.

To configure embedded Cassandra, select the **Options** tab and complete these steps:

1. Make sure the `enabled` option in the `[cassandraEmbedded]` section is set to `true`.
  2. Set the `clusterName` option in the `[cassandraEmbedded]` section to the name of your Cassandra cluster, which must not contain spaces. For example, `My_Cassandra_Cluster`.
  3. In the `[cassandraEmbedded]` section, you can take the default values for all of the options except `seedNodes`, which requires a comma-separated list of seed nodes, where each seed node is represented by an IP address or fully qualified domain name (FQDN). You must select your seed nodes from one or more of the nodes where Web Engagement Servers are deployed. For more information, see the description of the `seedNodes` option. Here are two examples of seed node lists:
    - `192.168.0.1,192.168.0.3`
    - `host1.mydomain.com,host2.mydomain.com`.
  4. Make sure that the `replicationStrategy` option in the `[cassandraKeyspace]` section is set to `NetworkTopologyStrategy`.
  5. Make sure that the `replicationStrategyParams` option in the `[cassandraKeyspace]` section is specified in accordance with the configuration of your Cassandra cluster.
  6. Edit ***Web Engagement installation directory/server/resources/cassandra-rackdc.properties***.
  7. Edit ***Web Engagement installation directory/server/resources/es-index.properties***.
  8. Verify that the required communication ports are opened.
15. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.

**End**

## Configuring a Connection to a Cluster of Chat Servers (Optional)

**Note:** If you are running Web Engagement in a production environment, Genesys recommends connecting the Web Engagement Cluster to a cluster of Chat Servers rather than to a single instance of Chat Server.

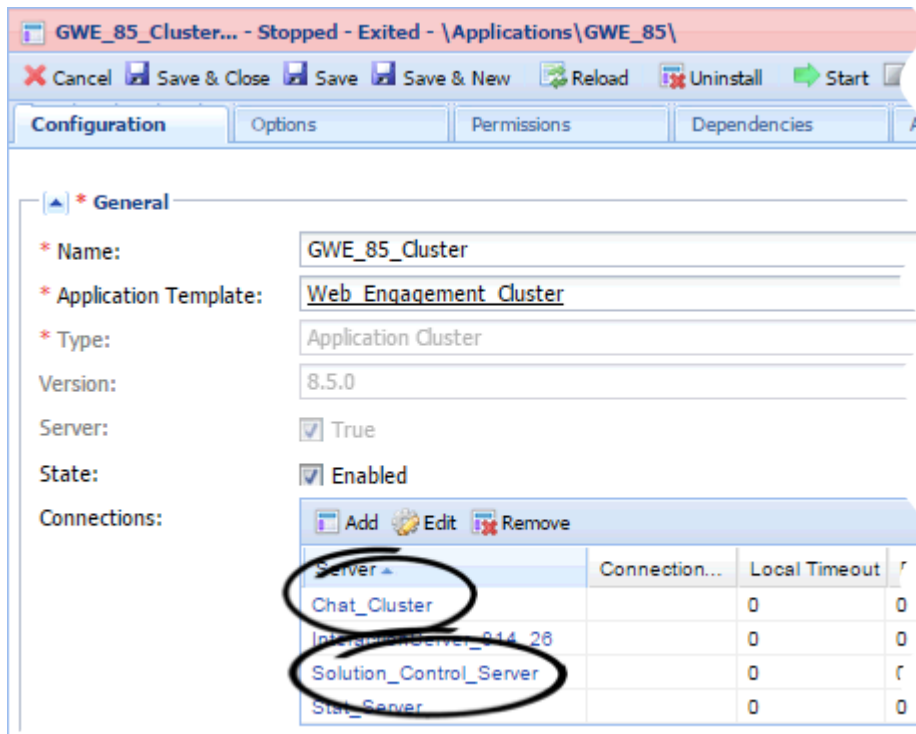
Complete the steps below to configure a cluster of Chat servers for the Cluster.

**Start**

1. In Genesys Administrator, create an application with a type of **Application Cluster**. This example uses an Application Cluster app called **Chat Cluster**.
2. Navigate to **Provisioning > Environment > Applications**, select your Cluster application, and click **Edit**.
3. In the **Connections** section, add a connection to Solution Control Server and an Application Cluster

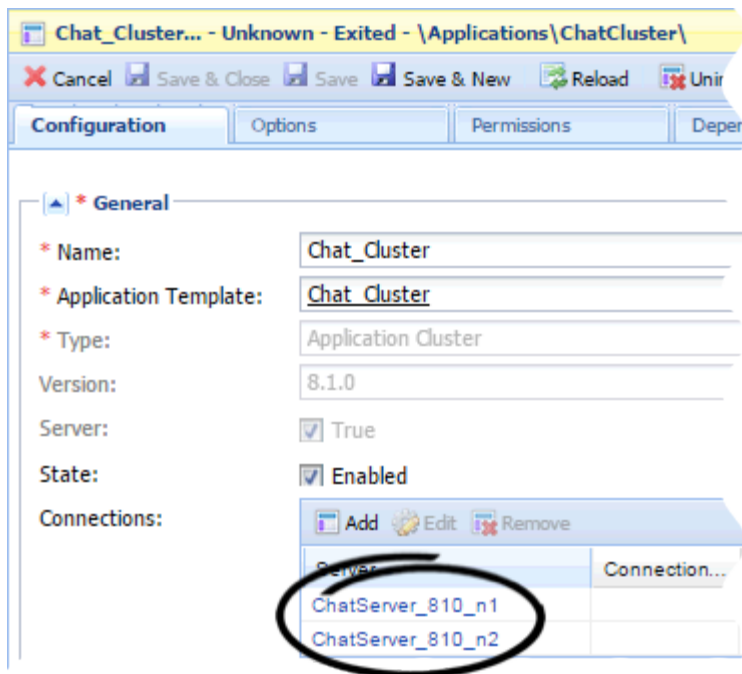


application that has connections to one or more Chat servers.



The Cluster has a connection to the **ChatCluster** Application Cluster application.

4. Click **Save & Close**
5. Open your Application Cluster application.
6. In the **Connections** section, add connections to one or more Chat Servers, using a port ID of **webapi** for each connection.



The **ChatCluster** application has connections to two Chat servers.

7. Click **Save & Close**.

**End**

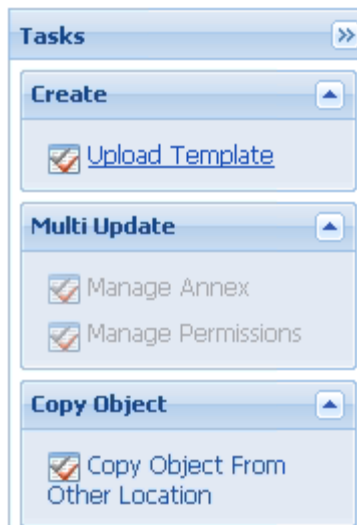
## Importing the Web Engagement Server Template

### Prerequisites

- You completed [Configuring the Cluster Application](#).

### Start

- Open Genesys Administrator and navigate to **Provisioning > Environment > Application Templates**.
- In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.
4. Browse to the **Web\_Engagement\_Server.apd** file or, if your Configuration Server does not support Web Engagement specific types, select **Web\_Engagement\_Server\_Generic.apd**, available in the **templates** directory of your installation CD. The **New Application Template** panel opens.
5. Click **Save & Close**.

**End**

## Creating a Node Application

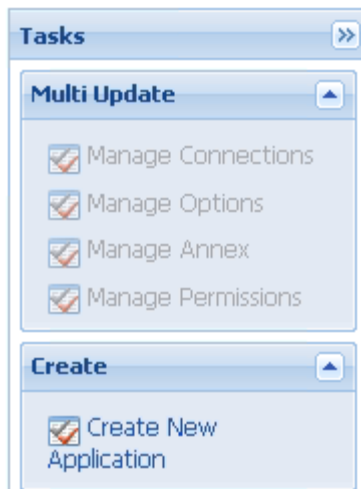
You must create and configure every node that you add to the cluster, using the instructions in this section and the [next one](#).

### Prerequisites

- You completed [Importing the Web Engagement Server Template](#).

### Start

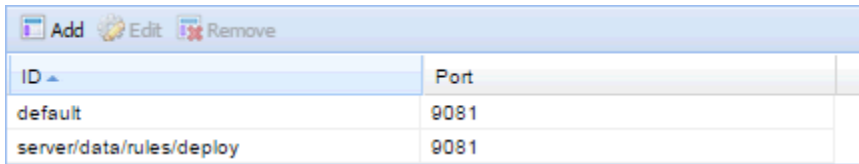
1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.



Create New Application link.

3. In the **Select Application Template** panel, click **Browse for Template** and select the Web Engagement Server template that you imported in [Importing the Web Engagement Server Template](#). Click **OK**.
4. The template is added to the **Select Application Template** panel. Click **Next**.
5. In the **Select Metadata file** panel, click **Browse**, then click **Add**, and select the **Web\_Engagement\_Server.xml** file or select the **Web\_Engagement\_Server\_Generic.xml** file if you chose **Web\_Engagement\_Server\_Generic.apd** in Step 4 of [Importing the Web Engagement Server Template](#). Click **Open**.
6. The metadata file is added to the **Select Metadata file** panel. Click **Next**.
7. In **Specify Application parameters**:
  - Enter a name for your application. For instance, GWE\_85\_Node\_1.
  - Make sure that **State** is enabled.
  - Select the **Host** on which the node will reside.
  - Click **Create**.
8. The **Results** panel opens. Click the **Finish** button. The Web Engagement Node application form opens and you can start configuring the Web Engagement Node application.
9. Expand the **Server Info** pane.
10. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
  - Enter the **Port**. For instance, 9081.
  - Choose http for the **Connection Protocol**.
  - Click **OK**. The HTTP port with the default identifier appears in the list of **Listening ports**.
11. Create the port with the server/data/rules/deploy identifier by clicking **Add**.
  - Enter server/data/rules/deploy for the ID.
  - Enter the same port value as you did for the default port. For instance, 9081.
  - Select http for the **Connection Protocol**.

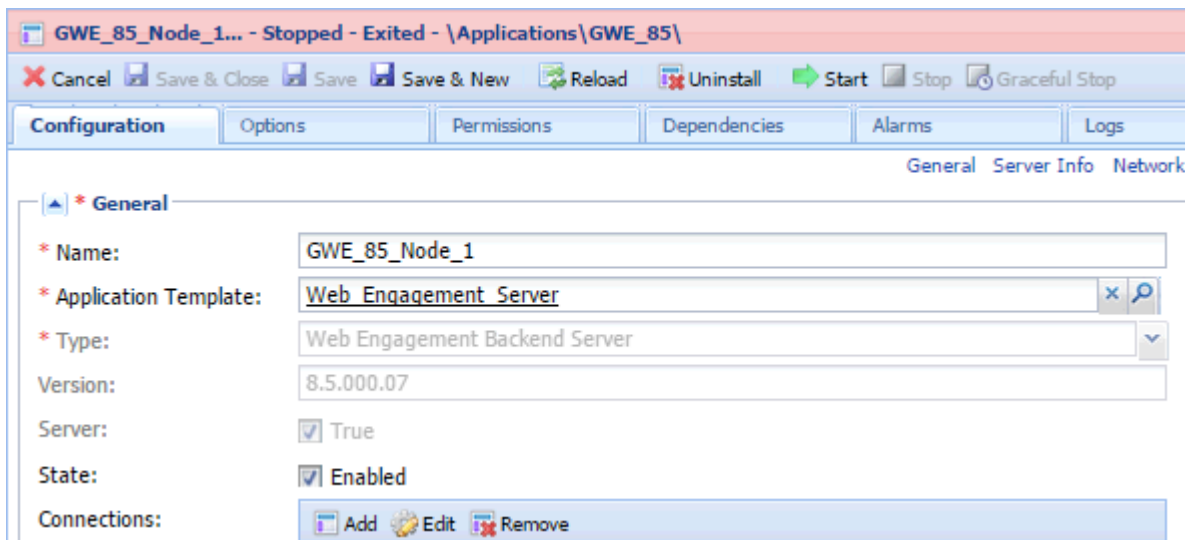
- Click **OK**. The HTTP port with server/data/rules/deploy ID appears in the list of Listening ports.



ID	Port
default	9081
server/data/rules/deploy	9081

Node listening ports

- Click **Save**.
- A **Confirmation** dialog opens. Click **Yes**.
- (Optional) If you are using a Web\_Engagement\_Server\_Generic template for your application object, you need to create a **[webengagement]** section in the Annex. Click **New** in the **Options** tab. The **New Options** dialog opens.
  - Select Annex for **Location**.
  - Enter webengagement for **Section**.
  - Enter type for **Name**.
  - Enter gweserver for **Value**.
  - Click **OK** to create the new annex. You can see it by selecting Advanced View (Annex) in the **View** selector.
- Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.
- Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The GWE\_85\_Node\_1 application form opens and you can start configuring the node application.



GWE\_85\_Node\_1... - Stopped - Exited - \Applications\GWE\_85\

Cancel Save & Close Save Save & New Reload Uninstall Start Stop Graceful Stop

Configuration Options Permissions Dependencies Alarms Logs

General Server Info Network

\* General

\* Name: GWE\_85\_Node\_1

\* Application Template: Web Engagement Server

\* Type: Web Engagement Backend Server

Version: 8.5.000.07

Server:  True

State:  Enabled

Connections: Add Edit Remove

Node app opened in Genesys Administrator.

**End**

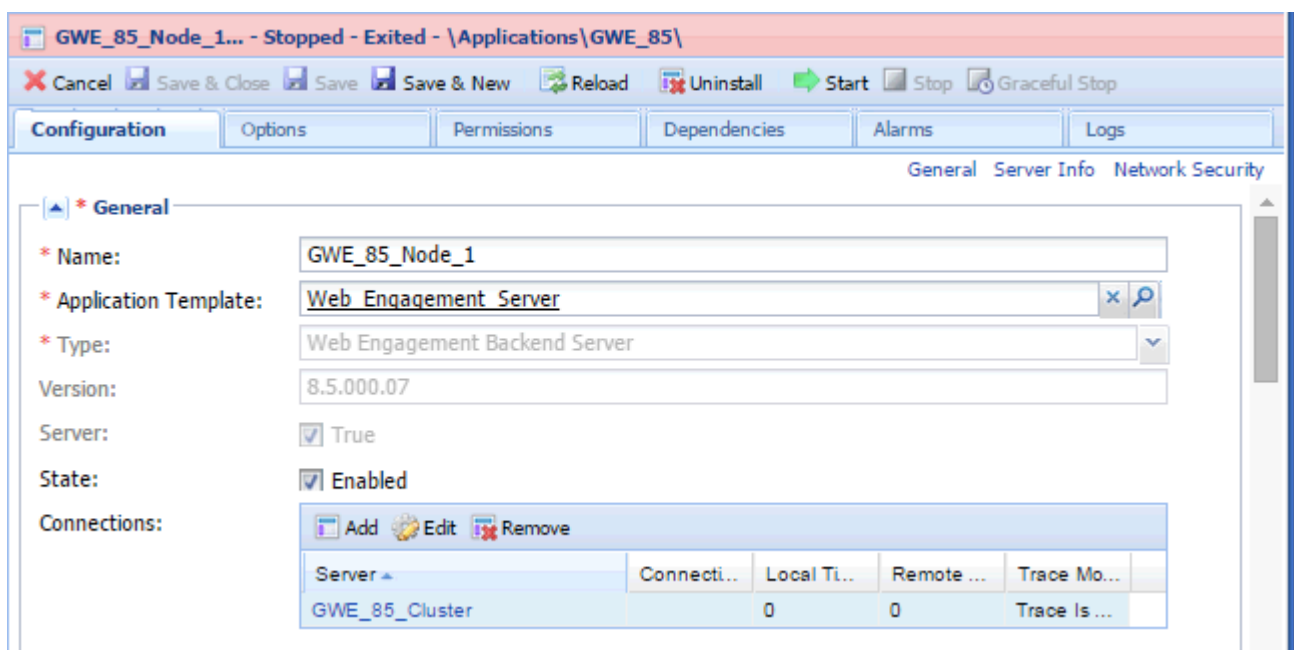
## Configuring a Node Application

### Prerequisites

- You completed [Creating a Node Application](#).

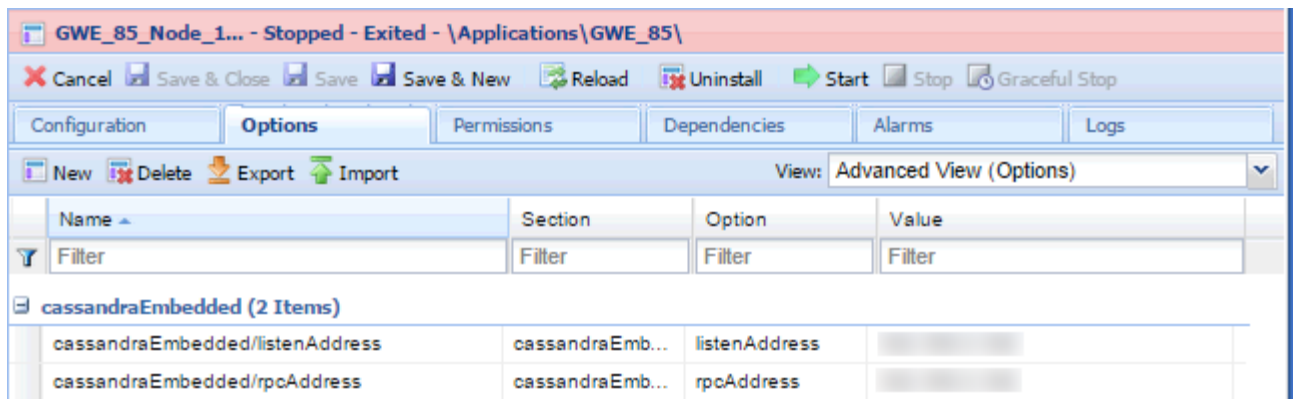
### Start

- If your node application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the node and click **Edit...**
- In the **Connections** section of the **Configuration** tab, click **Add**. The **Browse for applications** panel opens. Select the Web Engagement Cluster application you defined above, then click **OK**.



Node connection to Cluster

- Expand the **Server Info** pane.
- In the Tenant section, click **Add** and select your tenant. For instance, Environment. Click **OK**.
- If your **Host** is not defined, click the lookup icon to browse to the hostname of your application.
- The **Confirmation** dialog for changing the application's port opens. Click **Yes**.
- Important:** Genesys recommends that you use external Cassandra, which you can configure by following [these instructions](#). However, embedded Cassandra can be a better choice for a lab installation.  
**Important:** Starting in 8.5.0, Embedded Cassandra mode is deprecated in Web Engagement; support for this mode will be discontinued in 9.0.  
To configure embedded Cassandra, select the **Options** tab. In the `[cassandraEmbedded]` section, set the values for `listenAddress` and `rpcAddress`, using a fully qualified domain name or the appropriate IP address.



Cassandra Listen and RPC Addresses

8. (Optional) In the `[log]` section, the `all` option is set to `stdout` by default. Enter a filename if you wish to enable logging to a file. For example, you can enter `stdout, C:\Logs\WebEngagement\GWE_Node1` to force the system to write logs in the console and in a file called `GWE_Node1.log`.

## End

## Adding Nodes to a Cluster

### Prerequisites

You completed:

- [Creating the Cluster Application](#).
- [Configuring the Cluster Application](#)
- [Creating a Node Application](#)
- [Configuring a Node Application](#)

**Note:** A single-node configuration works well in a lab environment, but in production, you need more than one node, in order to provide high availability. And every time you establish a new node, you must complete the following steps in order to create and configure it.

### Start

1. Follow the instructions above for [Creating a Node Application](#), but use a different name for the new node.
2. [Configure the new node application](#), as shown above, but point to a different server address.

### End

**Important:** If you use more than one node, you need to set up [Load Balancing](#) in your environment.

---

## Installing the Web Engagement Server

Install the Web Engagement Server on Windows or Linux.

**Note:** For more information on how to install apps that you have configured in Genesys Administrator, consult [Generic Installation Procedures](#).

### Windows

#### Prerequisites

- [Configuring a Node Application](#)

#### Start

1. In your installation package, locate and double-click the **setup.exe** file. The Install Shield opens the welcome screen.
  2. Click **Next**. The **Connection Parameters to the Configuration Server** screen appears.
  3. Under **Host**, specify the host name and port number where Configuration Server is running. (This is the main "listening" port entered in the **Server Info** tab for Configuration Server.)
  4. Under **User**, enter the user name and password for logging on to Configuration Server.
  5. Click **Next**. The **Select Application** screen appears.
  6. Select the Web Engagement Server Application—that is, the Node app you created above—that you are installing. The **Application Properties** area shows the **Type**, **Host**, **Working Directory**, **Command Line executable**, and **Command Line Arguments** information previously entered in the **Server Info** and **Start Info** tabs of the selected Application object.
  7. Click **Next**. The **Choose Destination Location** screen appears.
  8. Under **Destination Folder**, keep the default value or browse for the desired installation location.
  9. Click **Next**. The **Backup Configuration Server Parameters** screen appears.
  10. If you have a backup Configuration Server, enter the **Host name** and **Port**.
  11. Click **Next**. The **Ready to Install** screen appears.
  12. Click **Install**. The Genesys Installation Wizard indicates it is performing the requested operation for Web Engagement Server. When through, the **Installation Complete** screen appears.
  13. Click **Finish** to complete your installation of the Web Engagement Server.
- **Before using Genesys Web Engagement, you must either create and deploy your own Web Engagement application or deploy the provided sample application.**

#### End

---



---

## Linux

### Prerequisites

- [Configuring a Node Application](#)

### Start

1. Open a terminal in the Genesys Web Engagement CD/DVD or the Genesys Web Engagement IP, and run the **install.sh** file. The Genesys Installation starts.
2. Enter the hostname of the host on which you are going to install.
3. Enter the connection information to log in to Configuration Server:
  - The hostname. For instance, `demorv.genesyslab.com`.
  - The listening port. For instance, `2020`.
  - The user name. For instance, `demo`.
  - The password.
  - If the connection settings are successful, a list of keys and Web Engagement applications is displayed.
4. Enter the key for the Web Engagement Server application—that is, the Node app you created above—that you created previously in Configuration Server.
5. Enter the location where Genesys Web Engagement is to be installed on your web server.  
**Note:** This location must match the previous settings that you entered in Configuration Server.
6. If you have a backup Configuration Server, enter the Host name and Port.
7. If the installation is successful, the console displays the following message:  
Installation of Genesys Web Engagement Server, version 8.5.x has completed successfully.
  - **Before using Genesys Web Engagement, you must either [create](#) and [deploy](#) your own Web Engagement application or deploy the provided sample application.**

### End

## Configuring alarms

Genesys recommends that you tune the following [Web Engagement alarms](#):

- Incorrect load balancer routing
- Event Duration
- VisitProfile cache size
- Events cache size
- DroolsSession cache size

- Heap Memory Usage
- GC Latency

**Next Steps**

- Before using Genesys Web Engagement, you must either **create** and **deploy** your own Web Engagement application or deploy the provided sample application.
- At that point, you can **Configure Genesys Rules System** to work with GWE.

# Configuring Genesys Rules System

Complete the procedures in the tabs below to tune Genesys Rules System to work with Web Engagement.

## Genesys Rules Authoring Tool

### Configuring Genesys Rules Authoring Tool

#### Prerequisites

- Your environment includes Genesys Rules Authoring Tool (GRAT). See [Genesys environment prerequisites](#) for compliant versions. For more information about installing GRAT, refer to the [Genesys Rules System Deployment Guide](#).

#### Start

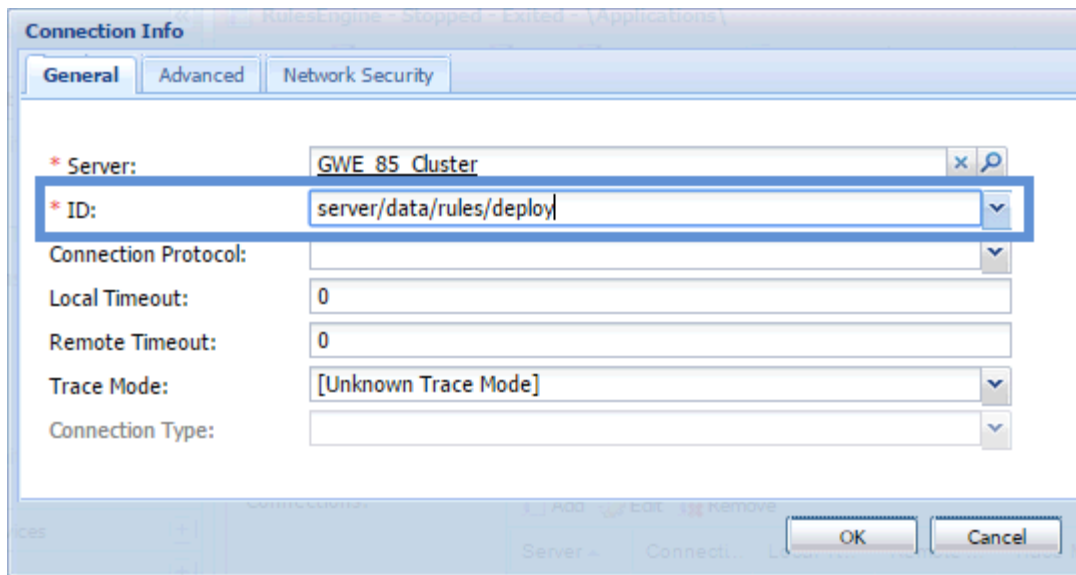
1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the GRAT Server application (the application with type Business Rules Application Server), and click **Edit**.
2. In the Connections section, click **Add....**
3. Select the Genesys Web Engagement Cluster and click **OK**.

**Important:** Starting with versions 8.5.100.23 and 8.5.200.06, the Genesys Rules Authoring Tool (GRAT) no longer recognizes connections with Genesys Generic Server objects. Because of this, you can no longer deploy rules using Web Engagement Servers that are represented by Genesys Generic Server application objects.

**Important:** If your version of GRAT does not recognize connections with your Application Cluster object, you can connect to any of your Web Engagement Server node applications instead.

You can determine whether your Application Cluster is recognized by seeing whether it shows up as a valid location in the [rules package deployment scenario](#).

4. Select the Cluster in the list of connections and click **Edit**.
5. In the **Connection Info** window, select the correct port for the ID field, that is, /server/data/rules/deploy. Click **OK**.



ID is set to the correct port.

6. Select the **Options** tab.
7. In the **[settings]** section, set **verify-deploy-address** to false. You must set this option because in Genesys Web Engagement, rule packages are deployed to the Cluster, not to Genesys Rules Engine. When set to false, this option prevents GRS from trying to verify that the server deploying the rules is Genesys Rules Engine.

Configuration		Options	Permissions	Dependencies	Alarms	Logs
New Delete Export Import View: Advanced View (Options)						
Name	Section	Option	Value			
Filter	Filter	Filter	Filter			
<b>settings (6 Items)</b>						
settings/group-by-level	settings	group-by-level	false			
settings/max-connections	settings	max-connections	99			
settings/session-timeout	settings	session-timeout	30			
settings/session-timeout-alert-interval	settings	session-timeout-aler...	1			
settings/strict-mode	settings	strict-mode	true			
settings/verify-deploy-address	settings	verify-deploy-address	false			

**verify-deploy-address** is set to false.

8. In the **Permissions** tab, set a user who has Read, Create, Change rights for the Scripts folder in **Log On As**. This user should also have: Read access to all tenants which are supposed to be used; Role with sufficient permissions (as detailed in [Genesys Rules System Deployment Guide](#)); Read access to Business Structure folder and associated nodes that are supposed to be used; Read access to Scripts folder and Scripts objects (which are representations of the rule templates).
9. Click **Save & Close**.

**End**

## (Optional) Configuring Roles Settings for Rules Management

You should complete this procedure if you need to import permissions to enable a user to create rules for Genesys Web Engagement. Once roles are imported, you can assign them to the user who publishes the rule templates and creates rules for GWE.

You should not complete this procedure if you have already created a user who has permissions to create rules packages in Genesys Rules Authoring (as described in the "Role-based Access Control" chapter of the [Genesys Rules System Deployment Guide](#)).

### Start

1. In Genesys Administrator, navigate to **Provisioning > Accounts > Roles** and click **New...**
2. Enter a name for the new role. For example, GWE\_Rules\_Administrator.
3. In the Members section, you can specify who should have this role. Click **Add** to add as many access groups or users as you need.

The screenshot shows the 'New Role' configuration window in Genesys Administrator. The window title is 'GWE\_Rules\_Administrator - \Roles\'. The 'Configuration' tab is selected, with sub-tabs for 'Role Privileges' and 'Permissions'. The 'General' section contains the following fields:

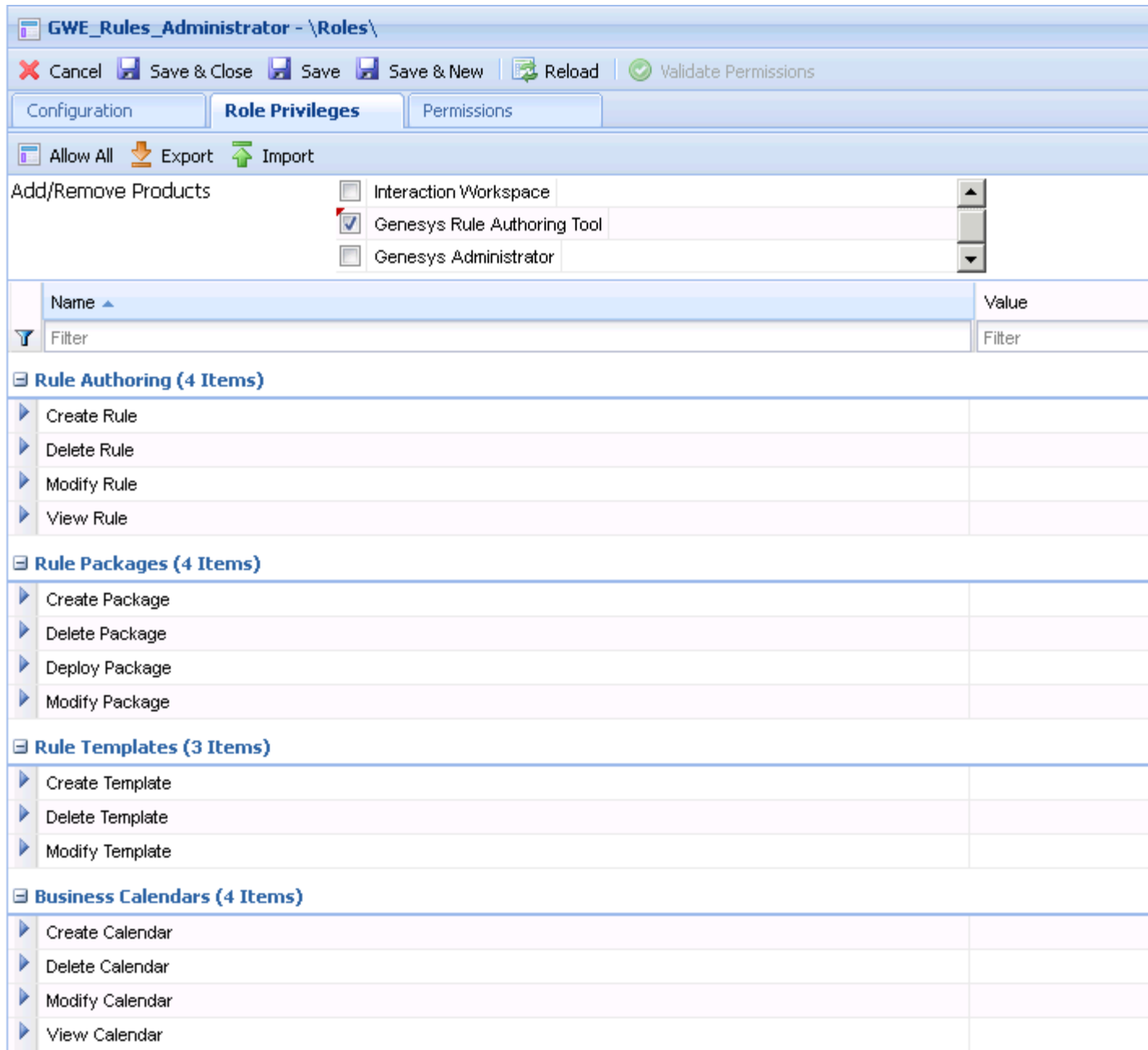
- Name:** GWE\_Rules\_Administrator
- Description:** in GRDT - import or create, modify and publish templates, in GRAT - create, modify, delete and dep
- Tenant:** Environment
- State:**  Enabled

The 'Members' section contains two empty tables:

- Users:** A table with columns 'User Name', 'Agent', 'Last Name', 'First Name', and 'Emp'. It contains the text 'No objects to display'.
- Access Groups:** A table with columns 'Name' and 'Type'. It contains the text 'No objects to display'.

At the bottom left of the window, the text 'New Role' is visible.

4. Select the **Role Privileges** tab and select Genesys Rules Authoring Tool. The privileges for GRAT are added to the GWE\_Rules\_Administrator role.



Imported Role Privileges

5. Click **Save & Close**.

## End

Members of the GWE\_Rules\_Administrator role can now do the following:

- Create, modify, and delete rules
- Create, modify, delete, and deploy rule packages

- Create, modify, and publish CEP rule templates

## Genesys Rules Development Tool

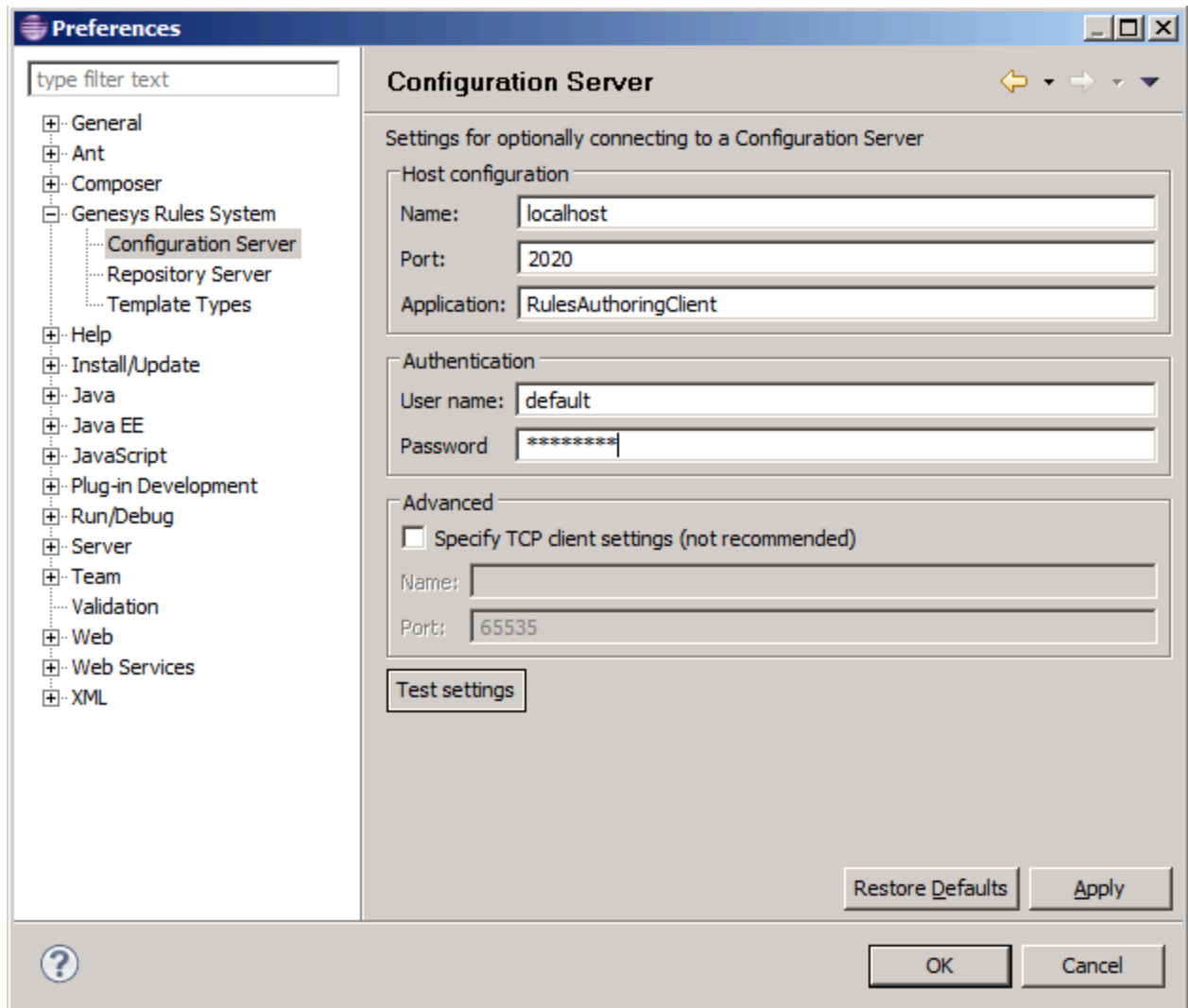
### Configuring Genesys Rules Development Tool

#### Prerequisites

- Your environment includes Genesys Rules Development Tool (GRDT), deployed as a Composer plug-in. See [Genesys environment prerequisites](#) for compliant versions. For more information about installing GRDT, refer to [Installing the GRDT Component](#) in the the Genesys Rules System Deployment Guide.
- You enabled the Galileo update site in GRDT, as described in [Installing the GRDT Component](#).

#### Start

1. Open Genesys Rules Development Tool by starting Composer.
2. Navigate to **Window > Preferences**. The **Preferences** window opens.
3. Navigate to **Genesys Rules System > Configuration Server**. Edit the settings.
  - Enter the Configuration Server host name. For instance, localhost.
  - Enter the Configuration Server port. For instance, 2020.
  - Enter the application name for the Rules Authoring Client application. For instance, RulesAuthoringClient.
  - In the Authentication section, enter the name and password for a user who can connect to Configuration Server and click **Apply**.



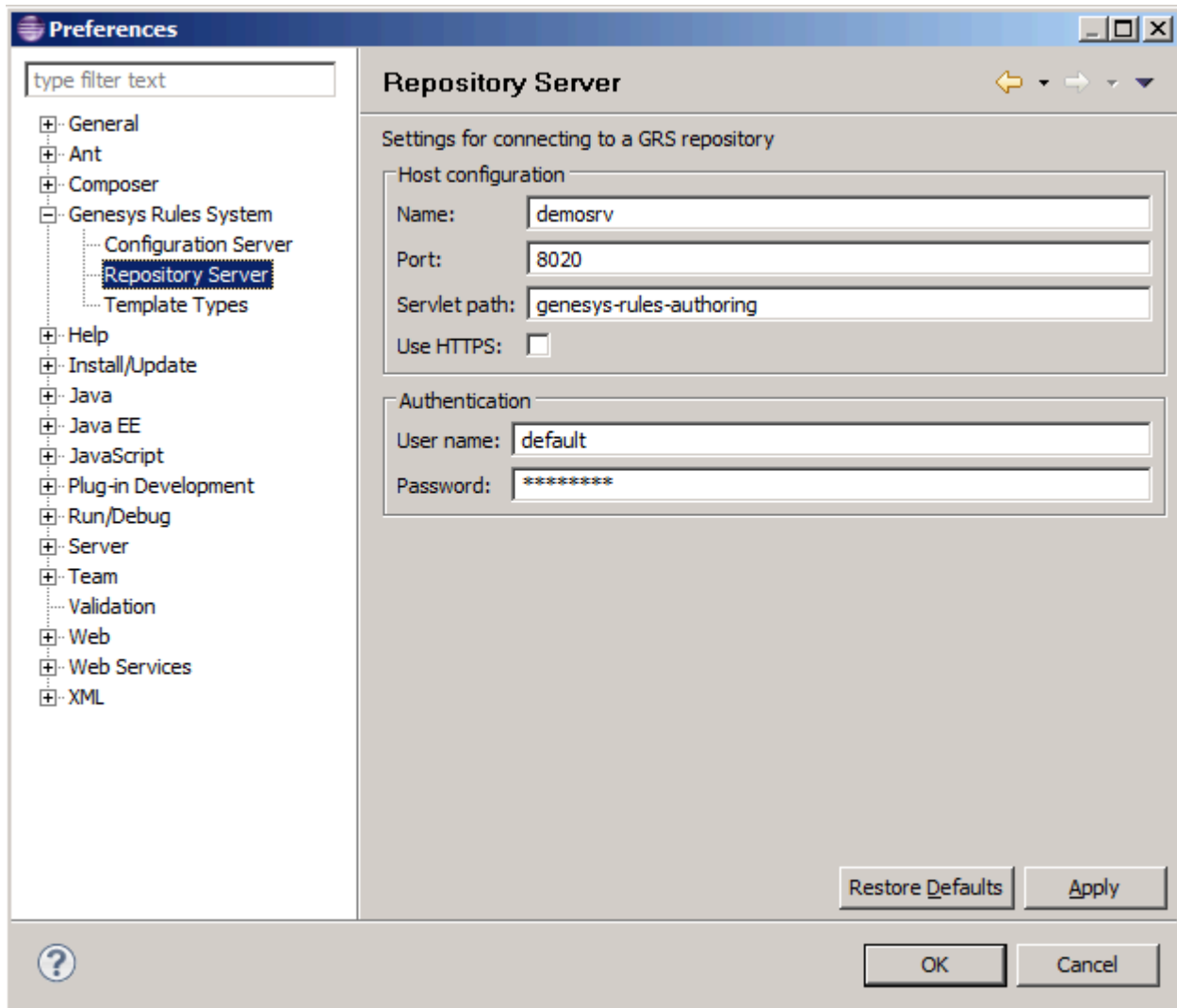
GRDT settings for Configuration Server.

4. Navigate to **Genesys Rules System > Repository Server**. Edit the settings.

- Enter the Repository Server host name. This is the name of the host specified for the GRS application (application with type Business Rules Execution Server) in the Genesys Configuration Layer. For instance, localhost.
- Enter the Repository Server port. This is the port with the ID genesys - rules - engine that is specified for the GRS application (with type Business Rules Execution Server) in the Genesys Configuration Layer. For instance, 8020.
- Enter the Servlet path as genesys - rules - authoring.
- In the Authentication section, enter a name and password for a user who:
  - Has Read and Execute permissions for the Genesys Rules Authoring client application (set up in Configuration Server); this user must have explicit Read and Execute permissions or must belong to an access group with those permissions.
  - Belongs to a Role with the following privileges: Template - Create, Template - Modify, Template - Delete.

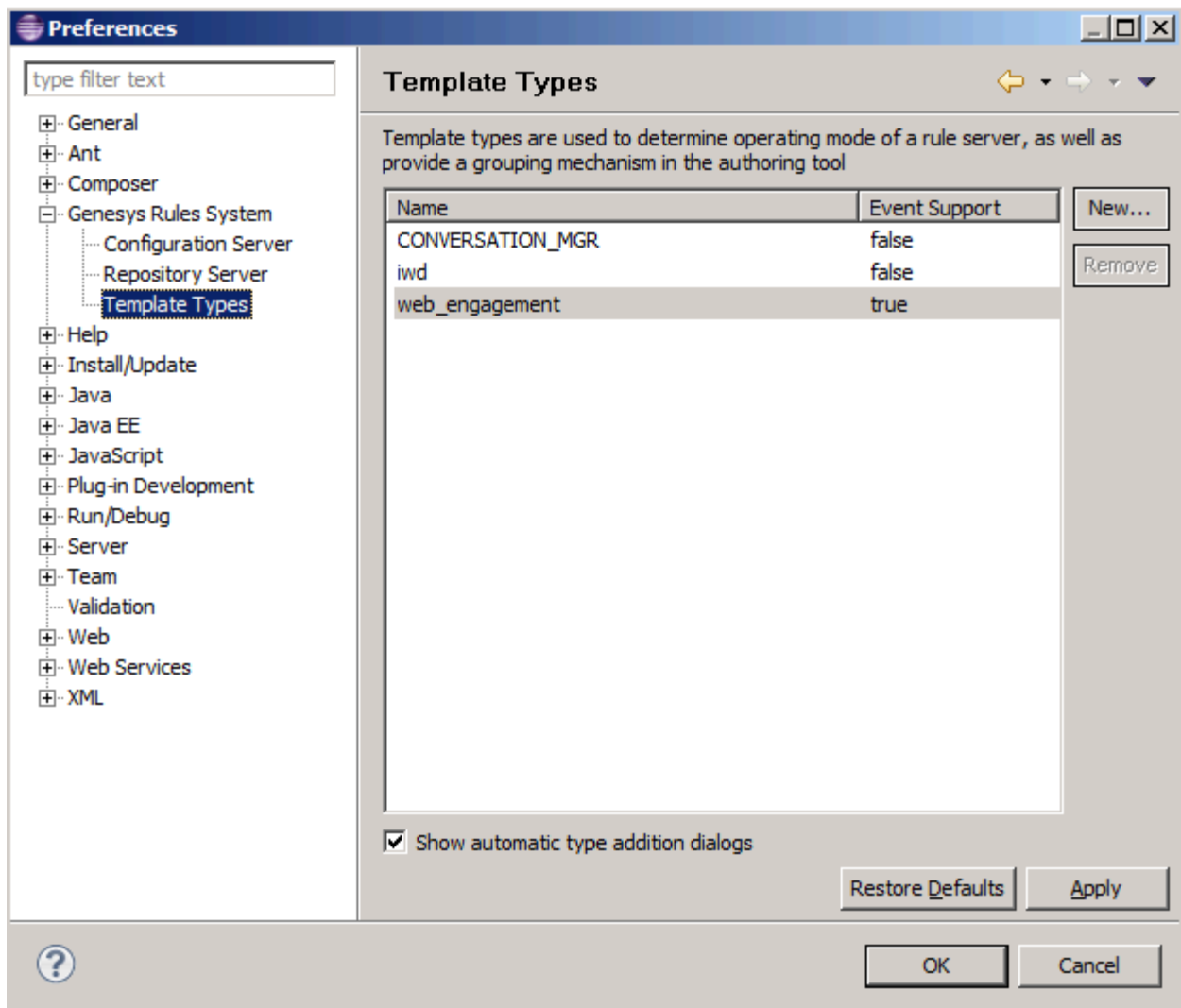


5. Click **Apply**.



GRDT settings for the Repository Server.

6. Navigate to **Genesys Rules System > Template Types**. If it is not present, add the **web\_engagement** template type and set **Event Support** to true. Click **Apply**.



Template types in Composer

7. Click **OK**.

**End**

### Next Steps

- Configure the [Generic Cassandra Settings](#) for your Web Engagement Server.

# Configuring External Cassandra

Genesys recommends using an external Cassandra cluster in your deployment architecture, because it gives you dedicated management of memory and other hardware resources. This means that both your Web Engagement Server processes and your Cassandra processes can better handle demand as your traffic increases.

To set up your own external Cassandra cluster, follow these steps:

- Create and configure [Cassandra access points](#) in Genesys Administrator for each node of the external Cassandra cluster.
- [Deploy Cassandra](#).

**Note:** Genesys recommends that you use Linux when deploying an external Cassandra cluster.

**Note:** You must synchronize the time on all hosts that contain Cassandra nodes. Failure to do this may lead to problems with removing data by TTL.

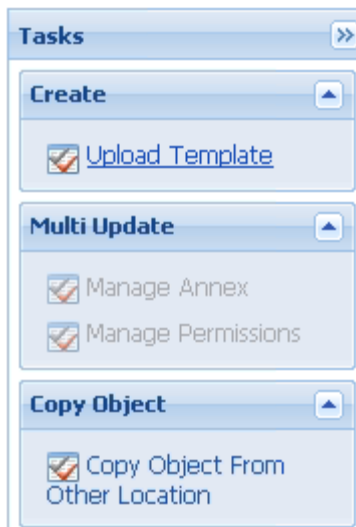
**Note:** If you plan to establish secure communications with your Cassandra cluster, Genesys recommends that you carefully evaluate the related [security considerations](#).

## Importing the Cassandra Resource Access Point Template

**Note:** For more information on how to work with templates in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Application Templates**.
2. In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.
4. Browse to the **Cassandra\_Resource\_Access\_Point\_850.apd** file. The **New Application Template** panel opens.
5. Click **Save & Close**.

**End**

## Creating the Cassandra Resource Access Point Application

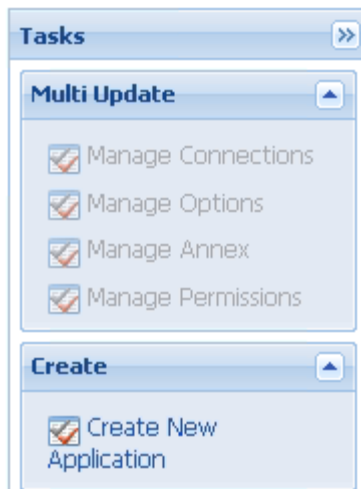
**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Importing the Cassandra Resource Access Point Template](#).

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.



Create New Application link.

3. In the **Select Application Template** panel, click **Browse for Template** and select the Cassandra Resource Access Point template that you imported in [Importing the Cassandra Resource Access Point Template](#). Click **OK**.
4. The template is added to the **Select Application Template** panel. Click **Next**.
5. In the **Select Metadata file** panel, click **Browse** and select the **Cassandra\_Resource\_Access\_Point\_850.xml** file. Click **Open**.
6. The metadata file is added to the **Select Metadata file** panel. Click **Next**.
7. In **Specify Application parameters**:
  - Enter a name for your application. For instance, `GWE_Cassandra_Access_Point`.
  - Make sure that **State** is enabled.
  - Select the **Host** on which the Access Point will reside.
  - Click **Create**.
8. The **Results** panel opens.
9. Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The Cassandra Resource Access Point application form opens and you can start configuring the Web Engagement Cluster application.

**End**

## Configuring the Cassandra Resource Access Point Application

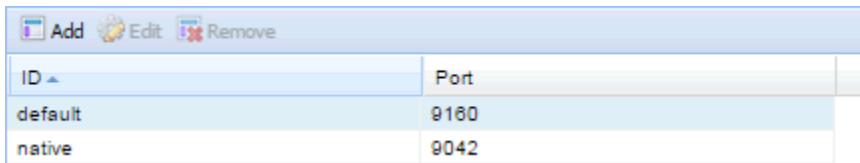
**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Creating the Cassandra Resource Access Point Application](#).

## Start

1. If your Cassandra Resource Access Point application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the Cassandra Resource Access Point and click **Edit...**
2. Expand the **Server Info** pane.
3. In the Tenant section, click **Add** and select your tenant. For instance, Environment. Click **OK**.
4. If your **Host** is not defined, click the lookup icon to browse to the hostname of your application, which should point to the host where you plan to locate your Cassandra node.
5. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
  - Enter the **Port**. For instance, 9160.
  - Click **OK**. The **default** port appears in the list of **Listening ports**.
6. Click **Add** again. The **Port Info** dialog opens.
  - In the **ID** field, enter native.
  - Enter the **Port**. For instance, 9042.
  - Click **OK**. The **native** port appears in the list of **Listening ports**.



ID	Port
default	9160
native	9042

Cassandra port settings

7. Ensure the **Working Directory** and **Command Line** fields contain "." (period).

Configuration	Options	Permissions	Dependencies	Alarms	Logs
* Working Directory:	.				
* Command Line:	.				
Command Line Arguments:					
* Startup Timeout:	90				
* Shutdown Timeout:	90				
Backup Server:	[Unknown Backup Server]				
* Redundancy Type:	Not Specified				
* Timeout:	10				
* Attempts:	1				
Auto Restart:	<input type="checkbox"/> True				
Log On As SYSTEM :	<input checked="" type="checkbox"/> True				
* Log On Account:	[Unknown Log On Account]				

Commands

8. Click **Save**.
9. The **Confirmation** dialog for changing the application's port opens. Click **Yes**.
10. To configure External Cassandra, select the **Options** tab.
  - In the **[resource]** section, make sure **type** is set to **cassandra**.

Name	Section	Option	Value
Filter	Filter	Filter	Filter
<b>cassandraClient (1 Item)</b>			
cassandraClient.transportCompression	cassandraClient	transportCompression	LZ4
<b>resource (1 Item)</b>			
resource.type	resource	type	cassandra

Cassandra resource setting

11. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.

**End**

**Note that you must execute this procedure and the previous one for each Cassandra node in your cluster.**

## Configuring the Web Engagement Cluster for Use with External Cassandra

### Prerequisites

- You completed [Configuring the Cassandra Resource Access Point Application](#).

### Start

- Navigate to **Provisioning > Environment > Applications**. Select the application defined for the Web Engagement Cluster and click **Edit...**
- In the Connections section of the **Configuration** tab, click **Add**. The **Browse for applications** panel opens. Select a Genesys application defined as a Cassandra Resource Access Point, then select the **native** connection port, then click **OK**.
- Repeat the previous step for all Cassandra Resource Access Point applications configured for external Cassandra nodes belonging to the same data center.  
**Note:** You must not connect Cassandra Resource Access Point applications belonging to different data centers to the same Web Engagement Cluster.
- Select the **Options** tab.
  - In the `[cassandraEmbedded]` section, set `enabled` to false.

The screenshot shows a configuration window for 'GWE\_85\_Cluster... - Stopped - Exited - \Applications\GWE\_85\'. The 'Options' tab is selected, and the 'View' is set to 'Advanced View (Options)'. A table lists configuration options for the 'cassandraEmbedded' section. The 'enabled' option is highlighted with a blue border and set to 'false'.

Name	Section	Option	Value
Filter	Filter	Filter	Filter
<b>cassandraEmbedded (10 Items)</b>			
cassandraEmbedded/clusterName	cassandraEmbe...	clusterName	Cluster
cassandraEmbedded/commitLogDirectory	cassandraEmbe...	commitLogDirect...	./storage/commitLog
cassandraEmbedded/dataDirectory	cassandraEmbe...	dataDirectory	./storage/data
cassandraEmbedded/enabled	cassandraEmbe...	enabled	false

Cassandra resource setting

- If you plan to use Web Engagement Reporting Server:
  - Make sure that the `replicationStrategy` option in the `[cassandraKeyspace]` section is set to `NetworkTopologyStrategy`.
  - Make sure that the `replicationStrategyParams` option in the `[cassandraKeyspace]` section is



specified in accordance with the configuration of your Cassandra cluster.

- In the `[kibana]` section, add the `elasticsearch_url` option. This option should point to the load balancer, which will balance requests to the Cassandra nodes using port 9200 (or, alternatively, the port specified in the `-Des.http.port` option, as explained later in this paragraph). For example, `http://external_cassandra_host:9200`.

**Note:** The Elasticsearch port is not specified in the default settings for external Cassandra. Because of this, you should use the default value of 9200. If you need to use a different port for Elasticsearch, use the `-Des.http.port` option.

5. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.

## End

# Deploy a Cassandra Cluster Node

## Linux

### Installation

1. Download version 2.2.9 or higher from the [Cassandra 2.2](#) stream.
2. Unpack the archive into the installation directory, for example:

```
cd /genesys
tar xzf apache-cassandra-2.2.x-bin.tar.gz
```

### Important

Do not use paths with spaces when installing Cassandra 2.2

### Configuration

1. Go to the directory where you installed your Cassandra node.
2. Edit `conf/cassandra.yaml`, using the following custom values:
  - **cluster\_name:** *cluster name without spaces, for example WebMe\_Cassandra\_Cluster*
  - **seeds:** *<comma-separated list of fully qualified domain names (FQDN) or IP addresses of one or more Cassandra nodes>* See the description of the `seedNodes` option for details. **Note:** This value must be the same for all nodes. Here are two examples:
    - 192.168.0.1,192.168.3
    - host1.mydomain.com, host2.mydomain.com

- **storage\_port**: 7000 (default value)
- **ssl\_storage\_port**: 7001 (default value)
- **listen\_address**: *<current node host name>* **Note:** This address is used for inter-node communication, so it must be available for use by other Cassandra nodes in your cluster.
- **native\_transport\_port**: 9042 (default value)
- **rpc\_address**: *<current node host name>* **Note:** This address is used by Web Engagement to connect to Cassandra, so it must be available to all Web Engagement hosts.
- **rpc\_port**: 9160 (default value)
- **endpoint\_snitch**: GossipingPropertyFileSnitch

**Note:** Make sure that each Cassandra node has access to the ports specified for the other nodes.

3. Edit **conf/cassandra-rackdc.properties**.
4. In order to set up indexing for Elasticsearch:
  1. Copy all jar files from **Web Engagement installation directory/tools/pulse/lib/es-index** to **Cassandra installation directory/lib**.
  2. Copy the **es-index.properties** file from **Web Engagement installation directory/server/resources** to **Cassandra installation directory/conf**.
  3. Edit the **es-index.properties** file.
5. Verify that the required communication ports are opened.

## Setting Up a Cassandra Service

The sample script described in the following procedure should give you an idea of how to set up Cassandra as a service process.

1. Create the **/etc/init.d/cassandra** startup script.
2. Edit the contents of the file:

```
#!/bin/sh
#
# chkconfig: - 80 45
# description: Starts and stops Cassandra
# update daemon path to point to the cassandra executable
DAEMON=<Cassandra_installation_dir>/bin/cassandra
start() {
    echo -n "Starting Cassandra... "
    $DAEMON -p /var/run/cassandra.pid
    echo "OK"
    return 0
}
stop() {
    echo -n "Stopping Cassandra... "
    kill $(cat /var/run/cassandra.pid)
    echo "OK"
    return 0
}
```

```

case "$1" in
  start)
    start
    ;;
  stop)
    stop
    ;;
  restart)
    stop
    start
    ;;
  *)
    echo $"Usage: $0 {start|stop|restart}"
    exit 1
esac
exit $?

```

3. Make the file executable: `sudo chmod +x /etc/init.d/cassandra`

4. Add the new service to the list: `sudo chkconfig --add cassandra`

5. Now you can manage the service from the command line:

- `sudo /etc/init.d/cassandra start`
- `sudo /etc/init.d/cassandra stop`

6. Configure the service to be started automatically together with the VM: `sudo chkconfig --level 2345 cassandra on`

## Windows

### Installation

1. Download version 2.2.9 or higher from the [Cassandra 2.2](#) stream.
2. Unpack the archive into a path without spaces.

### Configuration

1. Go to the directory where you installed your Cassandra node.
2. Edit **cassandra.yaml**, using the following custom values:
  - **cluster\_name**: *cluster name without spaces, for example WebMe\_Cassandra\_Cluster*
  - **seeds**: *<comma-separated list of fully qualified domain names (FQDN) or IP addresses of one or more Cassandra nodes>* See the description of the [seedNodes](#) option for details. **Note:** This value must be the same for all nodes. Here are two examples:
    - 192.168.0.1,192.168.3
    - host1.mydomain.com, host2.mydomain.com
  - **storage\_port**: 7000 (default value)

- **ssl\_storage\_port**: 7001 (default value)
- **listen\_address**: *<current node host name>* **Note:** This address is used for inter-node communication, so it must be available for use by other Cassandra nodes in your cluster.
- **native\_transport\_port**: 9042 (default value)
- **rpc\_address**: *<current node host name>* **Note:** This address is used by Web Engagement to connect to Cassandra, so it must be available to all Web Engagement hosts.
- **rpc\_port**: 9160 (default value)
- **endpoint\_snitch**: GossipingPropertyFileSnitch

**Note:** Make sure that each Cassandra node has access to the ports specified for the other nodes.

3. Edit **conf/cassandra-rackdc.properties**.
4. In order to set up indexing for Elasticsearch:
  1. Copy all jar files from **Web Engagement installation directory/tools/pulse/lib/es-index** to **Cassandra installation directory/lib**.
  2. Copy the **es-index.properties** file from **Web Engagement installation directory/server/resources** to **Cassandra installation directory/conf**.
  3. Edit the **es-index.properties** file.
5. Verify that the required communication ports are opened.
6. Start Cassandra.

## Configuring cassandra-rackdc.properties

For a single data center, use the following as a guide:

```
dc=<Data Center name>  
rack=<RACK ID>
```

For example,

```
dc=OperationalDC  
rack=RAC1
```

**Note:** Genesys recommends that you use the same **rack** ID if you do not have a clear understanding of your servers' rack usage. For more information about `cassandra-rackdc.properties`, refer to <http://docs.datastax.com/en/cassandra/2.2/cassandra/architecture/archsnitchGossipPF.html>.

## Configuring es-index.properties

- For every node in a given data center, set the **discovery.zen.ping.unicast.hosts** property in the **es-index.properties** file for that node to a comma-separated list of the servers that host the Cassandra nodes contained in that data center.

If you have *n* data centers in your Cassandra cluster, then you should have *n* different versions of the **es-index.properties** file (one for each data center). The primary difference between these lists will be the values of the **discovery.zen.ping.unicast.hosts** property.

For example:

- The Cassandra nodes located on **host\_DC1\_A**, **host\_DC1\_B**, and **host\_DC1\_C** belong to data center **DC1**. Because of this, the **discovery.zen.ping.unicast.hosts** property in the **es-index.properties** files for nodes **host\_DC1\_A**, **host\_DC1\_B**, and **host\_DC1\_C** will be defined as `host_DC1_A,host_DC1_B,host_DC1_C`.
- The Cassandra nodes located on **host\_DC2\_X**, **host\_DC2\_Y**, and **host\_DC2\_Z** belong to data center **DC2**. Because of this, the **discovery.zen.ping.unicast.hosts** property in the **es-index.properties** files for nodes **host\_DC2\_X**, **host\_DC2\_Y**, and **host\_DC2\_Z** will be defined as `host_DC2_X,host_DC2_Y,host_DC2_Z`.
- Set the **index.number\_of\_shards** property to three times the number of Cassandra nodes in the current data center. For example, if you have 3 Cassandra nodes in the current data center, then the number of shards for each index should be 9.
- If you prefer not to store your Elasticsearch data in the default directory, you can use the **path.data** property to specify another location. This property is commented out by default. **Important!** You must make sure that the user on whose behalf Cassandra was started has write permissions to the path directory, whether this directory was explicitly specified or selected by default.

## Communication Ports

Cassandra and Elasticsearch use the following ports for external and internode communication.

**Note:** Either or both of them may not work as expected unless you ensure that these ports are opened for communication between all servers that host Cassandra nodes.

Port	Default	Where to Change the Value
Cassandra Storage port	7000	<code>storagePort</code> in the <b>[cassandraEmbedded]</b> section of the Web Engagement Cluster Application (for embedded Cassandra) or <b>storage_port</b> in <b>cassandra.yaml</b> (for external Cassandra)
Cassandra SSL Storage port	7001	<code>sslStoragePort</code> in the <b>[cassandraEmbedded]</b> section of the Web Engagement Cluster Application (for embedded Cassandra) or <b>ssl_storage_port</b> in <b>cassandra.yaml</b> (for external Cassandra)
Cassandra Thrift port	9160	<code>rpcPort</code> in the

Port	Default	Where to Change the Value
		<b>[cassandraEmbedded]</b> section of the Web Engagement Cluster Application (for embedded Cassandra) or <b>rpc_port</b> in <b>cassandra.yaml</b> (for external Cassandra)
Cassandra CQL port	9042	<b>nativeTransportPort</b> in the <b>[cassandraEmbedded]</b> section of the Web Engagement Cluster Application (for embedded Cassandra) or <b>native_transport_port</b> in <b>cassandra.yaml</b> (for external Cassandra)
Elasticsearch REST request service port	9200	<b>http.port</b> property in the <b>es-index.properties</b> file
Elasticsearch transport port	9300	<b>transport.tcp.port</b> property in the <b>es-index.properties</b> file

## Starting the Cassandra Cluster Nodes

Your Cassandra nodes must be started in a certain order:

1. Start the seed nodes.
2. Start the other non-seed nodes.

The seed node is one of the nodes specified in the **seeds** option.

## Verifying Your Cassandra Cluster

After you have deployed your Cassandra Cluster, you may want to verify that all of the nodes can communicate with each other. To do this, execute the following command on any Database VM:

### Linux

```
cd <Cassandra_installation_dir>/bin
./nodetool -h <hostname> status
```

### Windows

```
cd <Cassandra_installation_dir >/bin
nodetool -h <hostname> status
```

### Sample Output

This command should produce output that looks something like this:

```
Datcenter: DC1
=====
```

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--      Address                               Load                Tokens      Owns    Host
ID
UN      10.51.XX.XXX                106,36 KB          256          ?
da6c-4f6a-820e-14538bd24a39          RAC1
UN      10.51.XX.XXX                108,22 KB          256          ?
aa1d-417b-911f-22340ae62c38          RAC1
UN      10.51.XX.XXX                107,61 KB          256          ?
fa4d-410e-431b-51297af13e96          RAC1
```

Datacenter: DC2

=====

```
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--      Address                               Load                Tokens      Owns    Host
ID
UN      10.51.XX.XXX                104,06 KB          256          ?
48ad4d08-555b-4526-8fab-d7ad021b14af  RAC1
UN      10.51.XX.XXX                109,56 KB          256          ?
8ca0fb45-aeef7-4f0a-
ac4e-a324ceea90c9                    RAC1
UN      10.51.XX.XXX                105,18 KB          256          ?
1c45e1fa-9f82-4bc4-a896-5575bad53808  RAC1
```

## Verifying Your Elasticsearch Cluster

Each of your Cassandra data centers must have a dedicated Elasticsearch cluster. To verify the status of your Elasticsearch clusters, execute the following request. (You can use your browser for this.)

### Prerequisites

- Your Cassandra cluster was initialized
- Your Cassandra cluster was started
- At least one Web Engagement Server instance is running in each of your Web Engagement Clusters

### Start

`http://<cassandra_node_host>:9200/_cluster/state?pretty`

### End

**Note:** This example uses port 9200, which is the default Elasticsearch HTTP port. If you are not using the default Elasticsearch HTTP port, substitute the port number that you are using.

### Output

The output from this request will describe your nodes, as shown here:

```
"nodes": {
  "xxx-xxxxxx-xxxxxxxxxxx": {
    "name": <node name>,
    "transport_address": <address for inter-node Elasticsearch communication>,
    "attributes": {}
```

```
}  
}
```

If your Elasticsearch clusters are correctly configured:

- Any requests executed on Cassandra nodes belonging to the *same* data center should result in *identical* lists of Elasticsearch nodes.
- Any requests executed on Cassandra nodes belonging to *different* data centers should result in *different* lists of Elasticsearch nodes.

## Upgrading Cassandra Nodes

You can upgrade your Cassandra version without interrupting service if:

- The version you are upgrading to is in the same stream (for example, from one 2.2.x version to another)
- You are not changing your database schema

Use the following steps for this task:

1. Stop the first Cassandra seed node.
2. Preserve your database storage.
3. Upgrade your Cassandra version, following the instructions in the Release Notes for the new version.
4. Be sure that your database storage is in the preserved state (the same set of files).
5. Start the first Cassandra seed node.
6. Execute steps 1 through 5 for the other seed nodes.
7. Execute steps 1 through 5 for the other non-seed nodes.
8. Verify that the Cassandra cluster is working, as shown above in [Verifying Your Cassandra Cluster](#).

If your upgrade plans include changing your database schema or changing Cassandra versions between streams (for example, from 2.0 to 2.2), then you will have to interrupt service. Use the following steps for this task:

1. Stop all of your Cassandra nodes.
2. If your database schema has been changed since you installed the previous version, update the Cassandra database, following the instructions in the Release Notes for the new version.
3. Configure each node, following the instructions in the Release Notes for the new version.
4. Start the Cassandra seed nodes.
5. Start the other nodes.
6. Verify that the Cassandra cluster is working, as shown above in [Verifying Your Cassandra Cluster](#).



## Maintenance

Because Cassandra is a critical component of Web Engagement, it is essential to keep track of its health. The Datastax documentation provides some really good information about how to do this at [http://docs.datastax.com/en/cassandra/2.0/cassandra/tools/toolsNodetool\\_r.html](http://docs.datastax.com/en/cassandra/2.0/cassandra/tools/toolsNodetool_r.html).

Genesys recommends that you use the **nodetool** utility that is bundled with your Cassandra installation package and that you make a habit of using the following nodetool commands to monitor the state of your Cassandra cluster.

### ring

Displays node status and information about the cluster, as determined by the node being queried. This can give you an idea of the load balance and whether any nodes are down. If your cluster is not properly configured, different nodes may show a different cluster; this is a good way to check that every node views the cluster the same way.

```
nodetool -h <HOST_NAME> -p <JMX_PORT> ring
```

### status

Displays cluster information.

```
nodetool -h <HOST_NAME> -p <JMX_PORT> status
```

### compactionstats

Displays compaction statistics.

```
nodetool -h <HOST_NAME> -p <JMX_PORT> compactionstats
```

### getcompactionthroughput \ setcompactionthroughput

Displays the compaction throughput on the selected Cassandra instance. By default it is 32 MB/s.

You can increase this parameter if you observe permanent growth of database size after the TTL and grace periods are passed. Note that increasing compaction throughput will affect memory and CPU consumption. Because of this, you need make sure to have sufficient hardware to support the rate that you have selected.

```
nodetool -h <HOST_NAME> -p <JMX_PORT> getcompactionthroughput
```

To increase compaction throughput to 64 MB/s, for example, use the following command:

```
nodetool -h <HOST_NAME> -p <JMX_PORT> setcompactionthroughput 64
```

## Recovery

Depending on the replication factor and consistency levels of a Cassandra cluster configuration, the Web Engagement Cluster can handle the failure of one or more Cassandra nodes in the data center without any special recovery procedures and without interrupting service or losing functionality. When the failed node is back up, the Web Engagement Cluster automatically reconnects to it.

- Therefore, if an eligible number of nodes have failed, you should just restart them.

However, if too many of the Cassandra nodes in your cluster have failed or stopped, you will lose functionality. To ensure a successful recover from failure of multiple nodes, Genesys recommends that you:

- Stop every node, one at a time, with at least two minutes between operations.
- Then restart the nodes one at a time, with at least two minutes between operations.

---

# Automatic Provisioning

## Overview

You can create all the configuration information related to Genesys Web Engagement in Configuration Server by running the Provisioning Tool, located in the *Web Engagement installation directory\tools\provisioning* directory. The tool is run automatically as part of the installation process, but you can also **run the tool** to modify your configuration information after Genesys Web Engagement is installed. You don't usually need to do this, but it can be necessary when you change things like the application name or your Configuration Server parameters.

Note that you can run the tool under Windows or Linux.

## Created (or Corrected) Objects

The Provisioning Tool connects to Configuration Server and reads the configuration for the Web Engagement applications. It creates Genesys objects used by the Web Engagement Servers and edits the configuration files required to launch the Web Engagement Servers.

The following objects are created or corrected when you run the Provisioning Tool:

### Agent Groups

Provisioning creates two "default" Agent Groups: Web Engagement Chat and Web Engagement Voice. These groups are used in the default engagement and chat routing strategies provided by Genesys Web Engagement. They are also used to provide input for the pacing algorithm.

Location in Genesys Administrator: **Provisioning > Accounts > Agent Groups**

#### Important

Provisioning does not create Agent objects, so make sure you add agents to your new Agent Groups after running the tool.

### Categories

Genesys Web Engagement includes two out-of-the-box sample applications: **docs.genesys.com** and **playground**. Provisioning creates category objects to support these applications:

- **docs.genesys.com** sample application — Categories start with the **docs.genesys-** prefix.
- **playground** sample application — categories start with the **PlayGround-** prefix.

---

Location in Genesys Administrator: **Provisioning > Routing/eServices > Business Attributes > Web Engagement Categories > Attributes Values**

## Interaction Queues

Provisioning creates Interaction Queue objects that serve as the entry points for the Engagement Logic SCXML strategy and the Chat Routing SCXML strategy. Provisioning also creates Interaction Queues to support real-time reporting using CCPulse and Pulse templates. Provisioning creates the following queues:

- Engagement Logic SCXML strategy (and also for real-time reporting): Webengagement\_Qualified
- Chat Routing SCXML strategy: Webengagement\_Chat
- Real-time reporting:
  - Webengagement\_Qualified (also used as the entry point for the Engagement Logic SCXML strategy)
  - Webengagement\_Engaged
  - Webengagement\_Accepted
  - Webengagement\_Missed
  - Webengagement\_Rejected
  - Webengagement\_Timeout
  - Webengagement\_Failed

Location in Genesys Administrator: **Provisioning > Routing/eServices > Interaction Queues (or Provisioning > Environment > Scripts)**

## Interaction Queue Views

For each Interaction Queue object the Provisioning Tool creates, it also creates an Interaction Queue View object. Provisioning creates the following Interaction Queue View objects:

- Webengagement\_Qualified.EngagementLogic.View
- Webengagement\_Chat.ChatRouting.View
- Webengagement\_Engaged.Clean
- Webengagement\_Accepted.Clean
- Webengagement\_Missed.Clean
- Webengagement\_Rejected.Clean
- Webengagement\_Timeout.Clean
- Webengagement\_Failed.Clean

Location in Genesys Administrator: **Provisioning > Environment > Scripts**

---

## Enhanced Routing objects

Provisioning creates a set of Enhanced Routing objects to work with the previously created Interaction Queues.

- The `Webengagement_Qualified.Routing` object is used to provide the Engagement Logic SCXML strategy for Orchestration.
- The `Webengagement_Chat.Routing` object is used to provide the Chat Routing SCXML strategy for Orchestration.

The tool also creates the following Enhanced Routing objects, which are used for cleaning purposes (for example, to clean interactions that for some reason were stuck in one of the statistical-related Interaction Queues):

- `Webengagement_Engaged.Routing`
- `Webengagement_Accepted.Routing`
- `Webengagement_Missed.Routing`
- `Webengagement_Rejected.Routing`
- `Webengagement_Timeout.Routing`
- `Webengagement_Failed.Routing`

Location in Genesys Administrator: **Provisioning > Routing/eServices > Orchestration (or Provisioning > Environment > Scripts)**

## Case Data

Provisioning creates a pair of attributes under Case Data Business Attribute to support functionality in the Genesys Web Engagement Plug-in for Workspace Desktop Edition.

### Important

Provisioning only adds attributes to existing Case Data Business Attribute, but does not create the attribute if it is absent. The Case Data Business Attribute should be created during the Workspace Desktop Edition installation process.

The tool creates the following attributes:

- `WebEngagement.CurrentWebPage` (display name is Current Web Page)
- `WebEngagement.EngagementStartPage` (display name is Engagement Start Page)

Location in Genesys Administrator: **Provisioning > Routing/eServices > Business Attributes > Case Data > Attributes Values**

---

## Genesys Chat Server application

Provisioning corrects the Chat Server application connected to the Web Engagement Cluster application. It creates an option in the **[endpoints:Web Engagement Server Tenant ID]** section of the Chat Server application with the following value:

- Option name: **webme**
- Option value: Webengagement\_Chat

### Important

The connection to the Chat Server application does not have to be direct - it can be through a Genesys application with the type Application Cluster. See [Configuring a Connection to a Cluster of Chat Servers \(Optional\)](#) for details.

Location in Genesys Administrator: **Provisioning > Environment > Applications**

## Genesys Stat Server application

Provisioning adds some statistics and filters into the options of the Stat Server application connected to the Web Engagement Cluster application.

If absent, the following statistics are added:

- Chat\_Current\_In\_Queue
- Chat\_Current\_Waiting\_Processing\_In\_Queue
- Chat\_Total\_Entered\_Queue
- Webcallback\_Current\_In\_Queue
- Webcallback\_Current\_Waiting\_Processing\_In\_Queue
- Webcallback\_Total\_Entered\_Queue
- Webengagement\_Total\_Entered\_Queue
- Webengagement\_Total\_Moved\_From\_Queue

If absent, the following filters are added:

- Webengagement\_chat\_filter
- Webengagement\_filter
- Webengagement\_voice\_filter
- Webengagement\_rule\_TEMPLATE

Location in Genesys Administrator: **Provisioning > Environment > Applications**

---

## Genesys Web Engagement Server application

- The **queueKey** in the **[chat]** section is populated with the *Web Engagement Server Tenant ID:webme* value. For example, 1:webme. This change is paired with changes in the connected Chat Server.

Location in Genesys Administrator: **Provisioning > Environment > Applications**

## Running the Provisioning Tool

### Prerequisites

- The configuration applications for the Web Engagement servers are created in Configuration Server.
- The connections for the Web Engagement Server application include the Chat Server (or an Application Cluster object which contains a connection to Chat Server), Interaction Server, and the Stat Server applications.
- If you are using Genesys Generic Server templates, you have a **webengagement** section created in the annex for the Web Engagement Server applications and the **type** option is set to gwseserver. See [Creating a Node Application](#) for details.

### Start

1. Navigate to the Web Engagement installation directory and open the *Web Engagement installation directory\tools\provisioning* folder.
2. From the command line, run the following command: `webengagement_provisioning.bat -host Configuration Server host name or IP address -port Configuration Server port -user user ID login into Configuration Server -password password for the specified user ID -app Application name for Web Engagement Server`  
For Linux, use the same command line, but instead of `webengagement_provisioning.bat`, specify `webengagement_provisioning.sh`.

### Important

User and password options may be optional, according to your Configuration Server settings.

You can also run provisioning with the **overwrite** option. In overwrite mode, the provisioning tool replaces old objects with new objects. Already existing GWE-specific objects will be removed and new objects will be created instead. You will lose any changes you have made manually on GWE-specific objects. The command will look like this:

```
webengagement_provisioning.bat -host Configuration Server host name or IP address -port Configuration Server port -user user -password password -app Application name for Web Engagement Server -overwrite
```

### Important

User and password options may be optional, according to your Configuration Server settings.

3. The provisioning script starts. If the provisioning is successful, the following message is displayed:  
Provisioning script successfully finished his work

**End**

**Note:** The **webengagement\_provisioning.bat** and **webengagement\_provisioning.sh** files contain usage information that is printed when you run them. If you execute these files without parameters, this information will still be printed, but you will also receive an execution error.



---

# Tuning Your JVM

Web Engagement Server allows you to tune your JVM parameters, which is especially important with respect to the virtual memory allocated to the JVM. You can modify your JVM parameters in the **setenv.bat** or **setenv.sh** file that is located at **Web Engagement installation directory\server\**.

## Tuning Your Memory Allocation

Out of the box, the **setenv.bat** and **setenv.sh** files have the following memory settings:

```
set MEMORY_MIN=4096m
set MEMORY_MAX=4096m
```

If you want to change these settings, update the appropriate values and restart the Web Engagement Server.

**Note:** In older versions of GWE, memory management in **setenv.bat** and **setenv.sh** files was done explicitly through the **Xms** and **Xmx** parameters:

```
-Xms512m -Xmx1024m -XX:MaxPermSize=250m
```

Therefore, if you are using an older version of GWE, correct these parameters instead of **MEMORY\_MIN** and **MEMORY\_MAX**.

## Enabling the JMX Port

You can monitor the JVM where your Web Engagement Server is running by using some of the standard Java tools distributed with the JDK, such as JConsole or Visual VM. Since these tools work through the JMX port, the Web Engagement Server needs to open this port on startup.

Out of the box, the **setenv.bat** and **setenv.sh** files contain all of the settings you need to turn on JMX access. Here's what **setenv.sh** looks like:

```
:: Uncomment to enable JMX Remote
:: set JMX_PORT=7199
:: set JAVA_OPTS=%JAVA_OPTS% -Dcom.sun.management.jmxremote ^
:: -Dcom.sun.management.jmxremote.port=%JMX_PORT% ^
:: -Dcom.sun.management.jmxremote.local.only=false ^
:: -Dcom.sun.management.jmxremote.ssl=false ^
:: -Dcom.sun.management.jmxremote.authenticate=false
```

As you can see, the JMX parameters are turned off by default. To enable JMX access, you need to uncomment these settings.

**Note:** Opening the JMX port without adequate security can be risky. See the [JMX monitoring and management documentation](#) for information about how to tune your security settings when using

JMX to monitor your JVM.

## Applying Parameters to Your Windows Service

It is important to understand that any changes you make to the **setenv.bat** file will not be applied to the Windows service that starts Web Engagement Server. Here's how to apply the parameters from **setenv.bat** to your Windows service:

1. Make all necessary changes to **setenv.bat**.
2. Ensure that your Web Engagement Server can be started successfully from the command line.
3. Execute the **server.bat remove** command to remove your existing Windows service.
4. Execute the **server.bat install** command to install a new Windows service.

After you finish these steps, your Windows service will be synchronized with **server.bat**.

---

# Installing the Plug-in for Workspace Desktop Edition

The Genesys Web Engagement Plug-in for Workspace Desktop Edition allows you to enable chat and web callback engagement features in Workspace Desktop Edition. See [Genesys Web Engagement Plug-in for Workspace Desktop Edition Help](#) for details.

To install this plug-in, complete the following procedures:

1. [Installing the Plug-in for Workspace Desktop Edition](#)
2. [Importing the Plug-in for Workspace Desktop Edition Template](#)
3. [Adding a Connection to the Web Engagement Cluster](#).
4. [Adding a Connection to the Web Engagement Cluster using a load balancer option](#) (an alternative approach to [Adding a Connection to the Web Engagement Cluster](#)).
5. Genesys Web Engagement can also work with agents who are Team Leads. For details about how to configure Team Leads, see the following topics in the Workspace Desktop Edition Deployment Guide:
  - [Procedure: Enabling agents to be Team Leads](#)
  - [Monitoring Chat Interactions](#)

## Installing the Plug-in for Workspace Desktop Edition

### Prerequisites

- Your environment includes Workspace Desktop Edition. See [Genesys environment prerequisites](#) for compliant versions. For more information about installing Workspace Desktop Edition, refer to the [Workspace Desktop Edition Deployment Guide](#).

### Start

1. In your installation package, locate and double-click the **setup.exe** file.
2. Click **Next**. The **Select Installed Application** screen appears.
3. Select your Workspace Desktop Edition application.
4. Click **Next**. The **Ready to Install** screen appears.
5. Click **Install**. The Genesys Installation Wizard indicates it is performing the requested operation for the Genesys Web Engagement Plug-in for Workspace Desktop Edition. When through, the **Installation Complete** screen appears.
6. Click **Finish** to complete your installation. As a result of the installation, the following files are copied to the Workspace Desktop Edition installation directory:

- InteractionWorkspace\Genesyslab.Desktop.Modules.WebEngagement.dll
- InteractionWorkspace\Genesyslab.Desktop.Modules.WebEngagement.module-config
- InteractionWorkspace\Newtonsoft.Json.Net35.dll

**End**

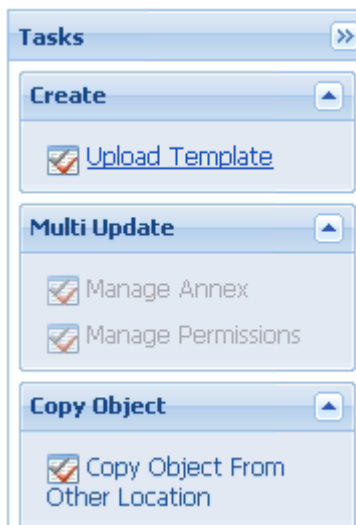
## Importing the Plug-in for Workspace Desktop Edition Template

### Prerequisites

- You completed [Installing the Plug-in for Workspace Desktop Edition](#)

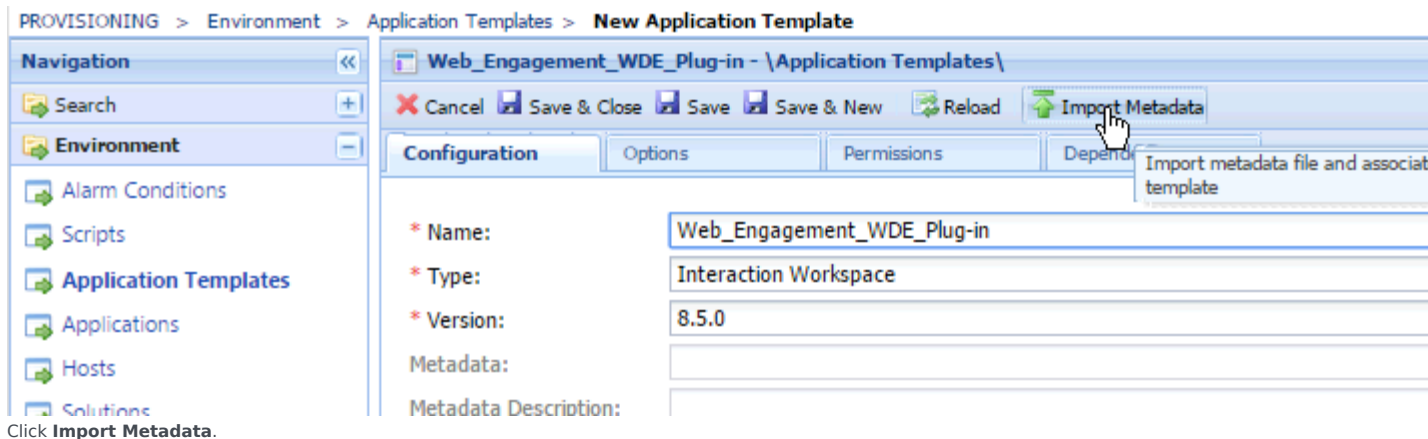
### Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Application Templates**.
2. In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.
4. Browse to the **Web\_Engagement\_WDE\_Plug-in.apd** file located in the **Templates** folder in your installation package. The **Configuration** tab for the new template opens.
5. Click **Import Metadata**.



Click **Import Metadata**.

6. Select the **Web\_Engagement\_WDE\_Plug-in.xml** metadata file and click **Open**. The metadata fields in the **Configuration** tab are now filled.
7. Click **Save & Close**.

**End**

## Adding a Connection to the Web Engagement Cluster

### Prerequisites

- You completed [Importing the Plug-in for Workspace Desktop Edition Template](#)

### Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the Workspace Desktop Edition application, and click **Edit...**
2. In the Connections section, click **Add**. The **Browse Applications** window opens.
3. Select the Web Engagement Cluster application and click **OK**. The cluster is added to the list of Connections.
4. Click **Save & Close**.

**End**

**Note:** If you are using Web Engagement 8.5.000.11, make sure that the **webengagement** section of your Web Engagement Cluster application's Annex contains a **type** option with a value of **backendserver**.

## Adding a Connection to the Web Engagement Cluster using a

## load balancer option

(This is an alternative approach to [Adding a Connection to the Web Engagement Cluster](#).)

### Prerequisites

- You completed [Importing the Plug-in for Workspace Desktop Edition Template](#)
- Your Workspace Desktop Edition application already has a connection to an application cluster other than the Web Engagement Cluster.

### Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the Workspace Desktop Edition application, and click **Edit....**
2. Select the **Options** tab and click **New**.
3. Set the following values:
  - **Location:** Options
  - **Section:** settings
  - **Name:** loadbalancer
  - **Value:** The address of your load balancer for the Web Engagement Cluster — for example, <http://198.51.100.12:8000>.
4. Click **OK**. The option is added to the **[settings]** section.
5. Click **Save & Close**.

### End

## Configuring Role-Based Access Control

Complete this procedure to allow specific users or groups to manage Web Engagement in Workspace Desktop Edition.

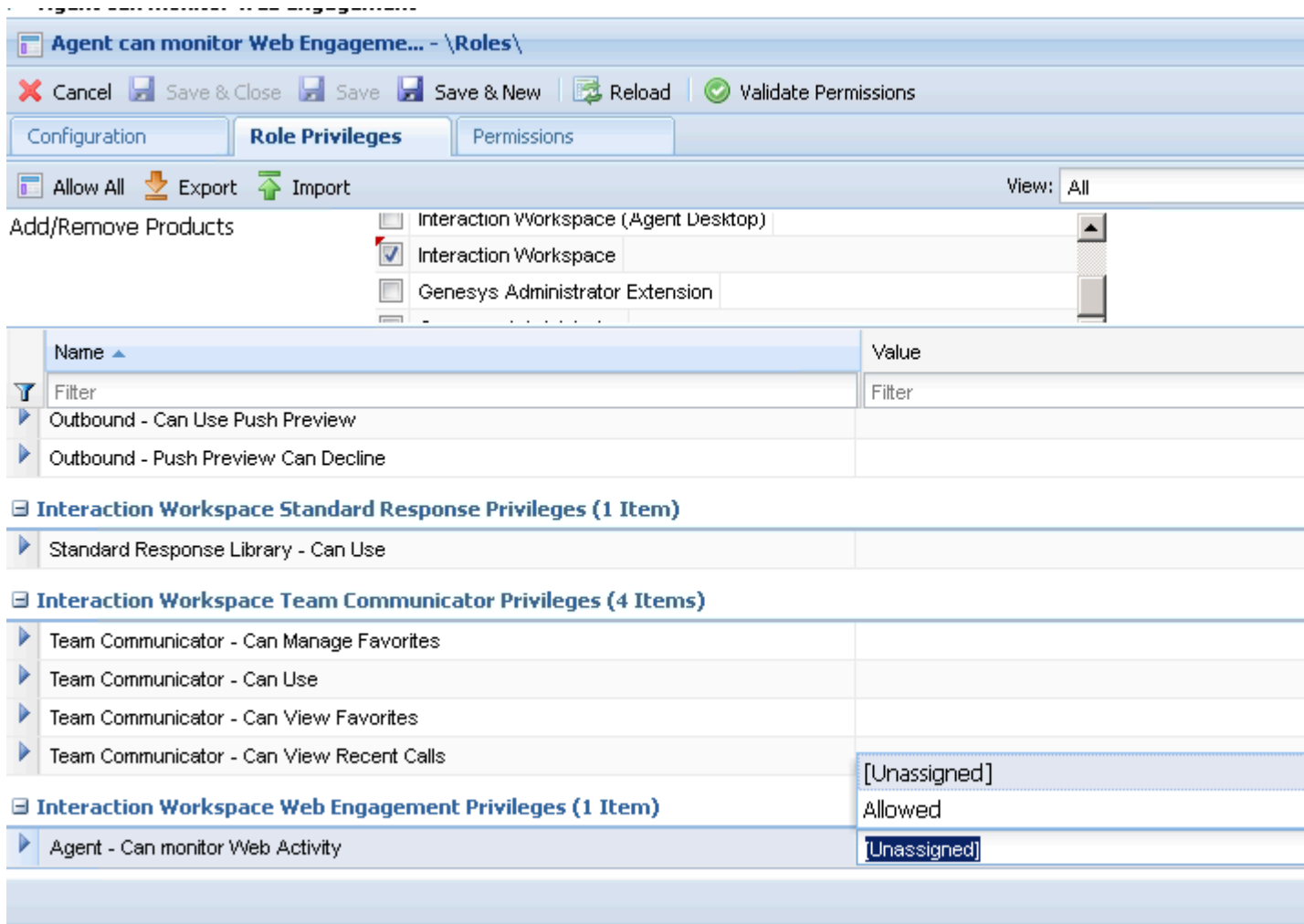
### Prerequisites

- You completed [Importing the Plug-in for Workspace Desktop Edition Template](#)

### Start

1. In Genesys Administrator, navigate to **Provisioning > Accounts > Roles**.
  2. Edit or create a Role responsible for managing Web Engagement in Workspace Desktop Edition. For instance, create the Agent can Monitor Web Engagement role by clicking the **New** button.
  3. Select the **Role Privileges** tab.
-

- In the **Add/Remove Products** top panel, enable Workspace Desktop Edition and expand the Workspace Desktop Edition Web Engagement Privileges section.
- Set the Allowed value for the **Agent - Can Monitor Web Activity** option.



Select Allowed

- In the Members section of the **Configuration** tab, add the users or groups who should get this role.
- Click **Save & close**.

**End**

**Next Steps**

- Installing the Plug-in for Genesys Administrator Extension

---

# Installing the Plug-in for Genesys Administrator Extension

The Genesys Web Engagement Plug-in for Genesys Administrator Extension allows you to use the [Script Generator tool](#) to create a JavaScript code snippet for your web pages to enable Web Engagement monitoring. The plug-in also allows you to [create categories](#), which you need if you implement the [Simple Engagement model](#).

Once installed, the plug-in adds a new Web Engagement section to the Configuration menu in GAX.

## Windows

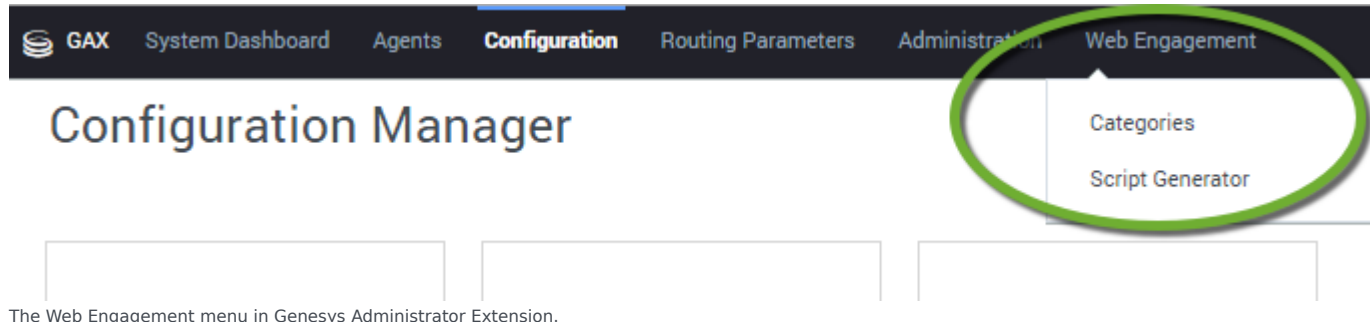
### Prerequisites

- Your environment includes Genesys Administrator Extension. See [Genesys environment prerequisites](#) for compliant versions. For more information about installing the Genesys Administrator Extension, refer to the [Genesys Administrator Extension Deployment Guide](#).
- Installation must be driven from the host where you intend to install the plug-in.

### Start

1. In Genesys Administrator, navigate to Provisioning > Environment > Applications, select the Genesys Administrator Extension server, and stop it.
2. In your installation package, locate and double-click the `setup.exe` file.
3. Click *Next*. The *Choose Destination Location* screen appears.
4. The destination location is the temporary folder where the plug-in files will be unpacked before they are copied into the GAX structure. You can browse to the desired location or use the default location and click *Next*. The *Ready to Install* screen appears.
5. Click *Install*. The Genesys Installation Wizard indicates it is performing the requested operation for the Genesys Web Engagement Plug-in for GAX. When through, the *Installation Complete* screen appears.
6. Click *Finish* to complete your installation. As a result of the installation, a new file called `gax-plugin-webme.jar` is copied to the `<GAX installation>\webapp\WEB-INF\lib\` folder.
7. In Genesys Administrator, navigate to Provisioning > Environment > Applications, select the Genesys Administrator Extension server, and start it.
8. Open Genesys Administration Extension in a web browser with the following URL:  
`http://<GAX_host_name>:<GAX_port>/gax/`.  
`<GAX_host_name>` — The name or IP address of the host for Genesys Administration Extension.  
`<GAX_port>` — The default port for Genesys Administration Extension.  
GAX now includes the **Web Engagement** menu with the following Web Engagement items: Categories and Script Generator.





The Web Engagement menu in Genesys Administrator Extension.

## End

## Linux

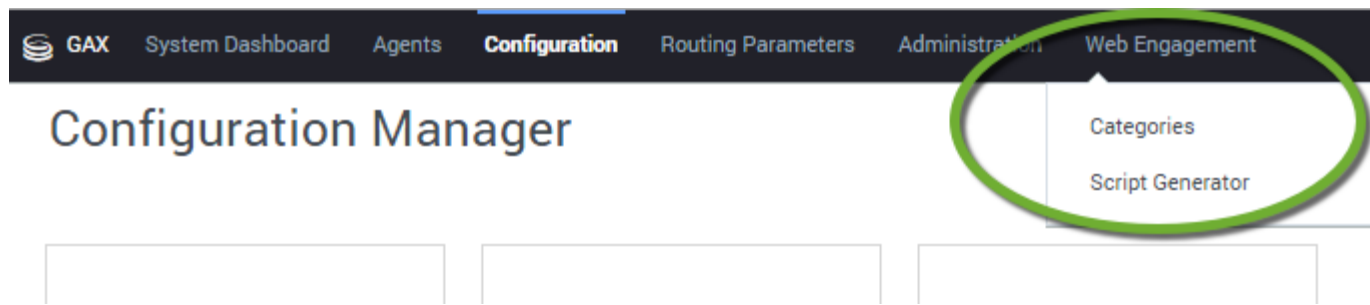
### Prerequisites

- Your environment includes Genesys Administrator Extension. See [Genesys environment prerequisites](#) for compliant versions. For more information about installing the Genesys Administrator Extension, refer to the [Genesys Administrator Extension Deployment Guide](#).
- Installation must be driven from the host where you intend to install the plug-in.

### Start

1. In Genesys Administrator, navigate to Provisioning > Environment > Applications, select the Genesys Administrator Extension server, and stop it.
2. Navigate to your installation package and run the `install.sh` file.
3. Enter the full path to the GAX installation directory.
4. Wait until the installation is finished.
5. In Genesys Administrator, navigate to Provisioning > Environment > Applications, select the Genesys Administrator Extension server, and start it.
6. Open Genesys Administration Extension in a web browser with the following URL:  
`http://<GAX_host_name>:<GAX_port>/gax/`.  
 <GAX\_host\_name> — The name or IP address of the host for Genesys Administration Extension.  
 <GAX\_port> — The default port for Genesys Administration Extension.

GAX now includes the **Web Engagement** menu with the following Web Engagement items: Categories and Script Generator.



The Web Engagement menu in Genesys Administrator Extension.

## End

**Note:** You must make sure that any Person objects that are configured for access to Genesys Administrator Extensions are [set up with Access to Web Engagement Categories and Web Engagement Script Generator privileges](#).

## Next Steps

- Now that you have installed Genesys Web Engagement and its plug-ins, you can [install the Web Engagement Reporting Server](#).

# Tuning Role-Based Access in Genesys Administrator

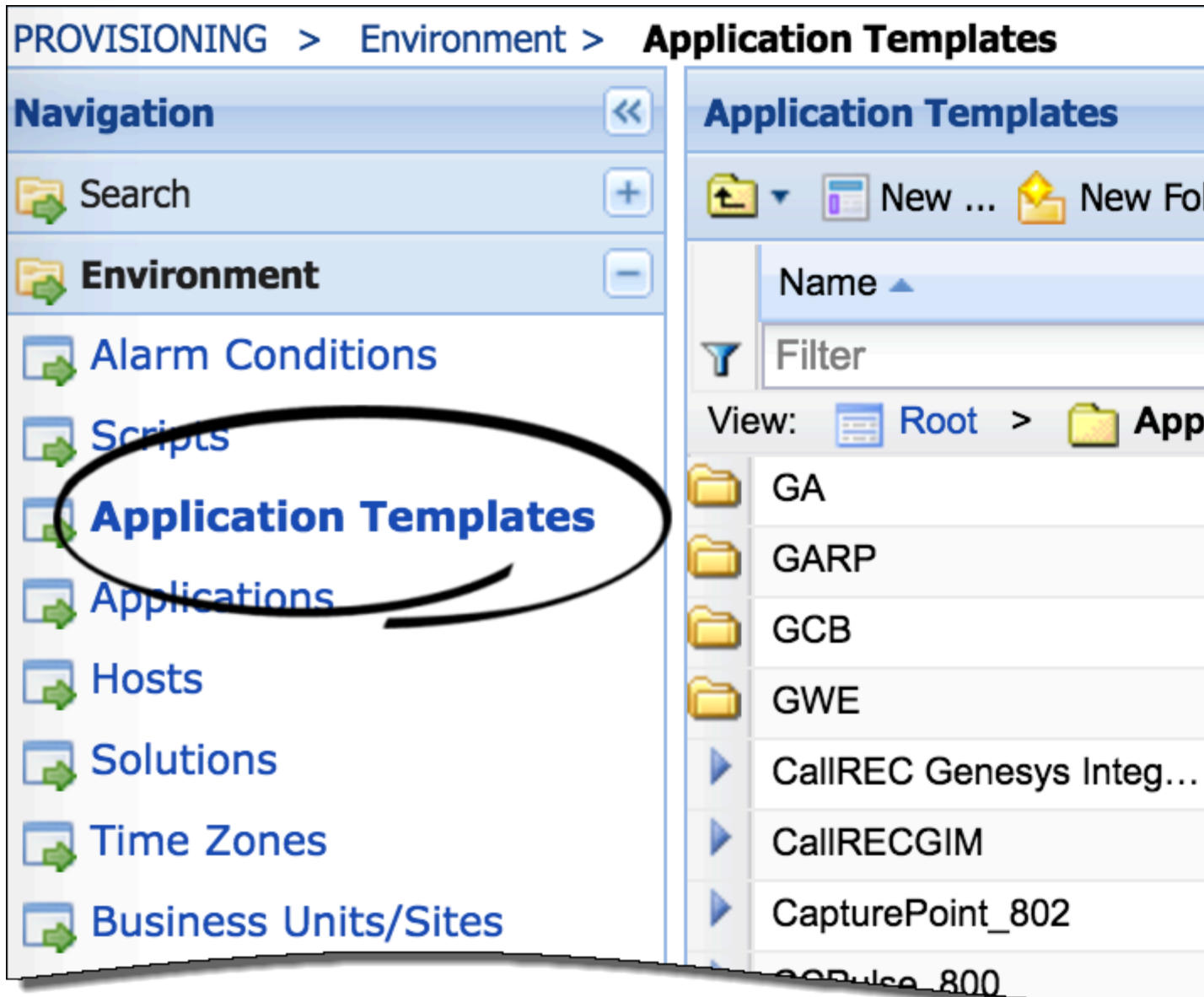
Web Engagement versions 8.5.000.26 and higher support role-based access for the following privileges:

- **Access to Web Engagement Categories**—allows users to create, read, update, or delete Web Engagement categories (starting with 8.5.000.26)
- **Access to Web Engagement Script Generator**—allows users to generate instrumentation scripts, which can then be copied and pasted onto the customer's site (starting with 8.5.000.26)
- **Access to Web Engagement Pacing Configuration**—allows users to configure Web Engagement pacing settings in Agent Group objects (starting with 8.5.000.33)

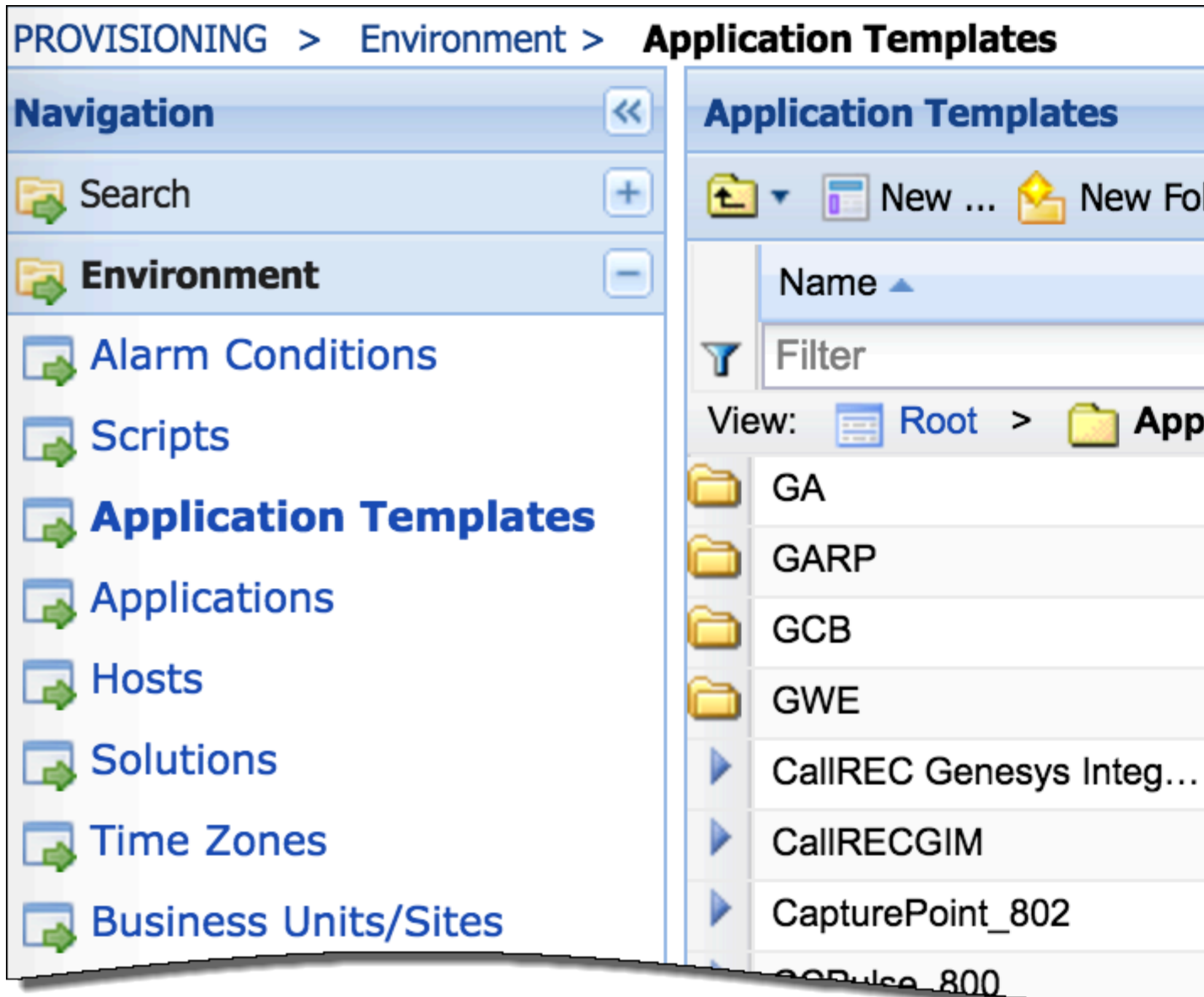
**Note:** Once you have enabled these privileges, as shown below, you must grant the appropriate privileges to any users who need to use the related functionality.

Here is how to set up these privileges:

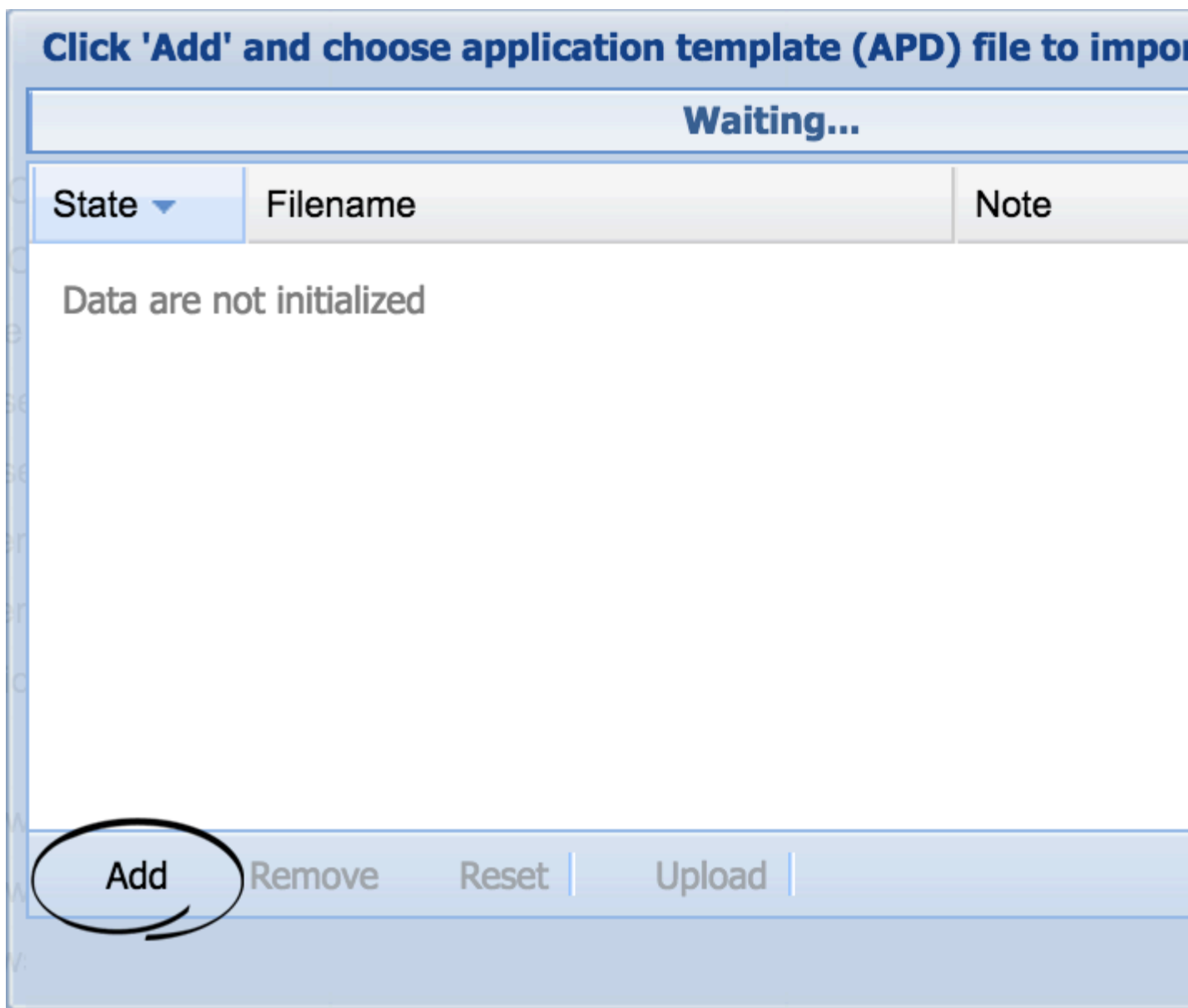
1. Open Genesys Administrator.
2. Go to **Environment/Application Templates**.



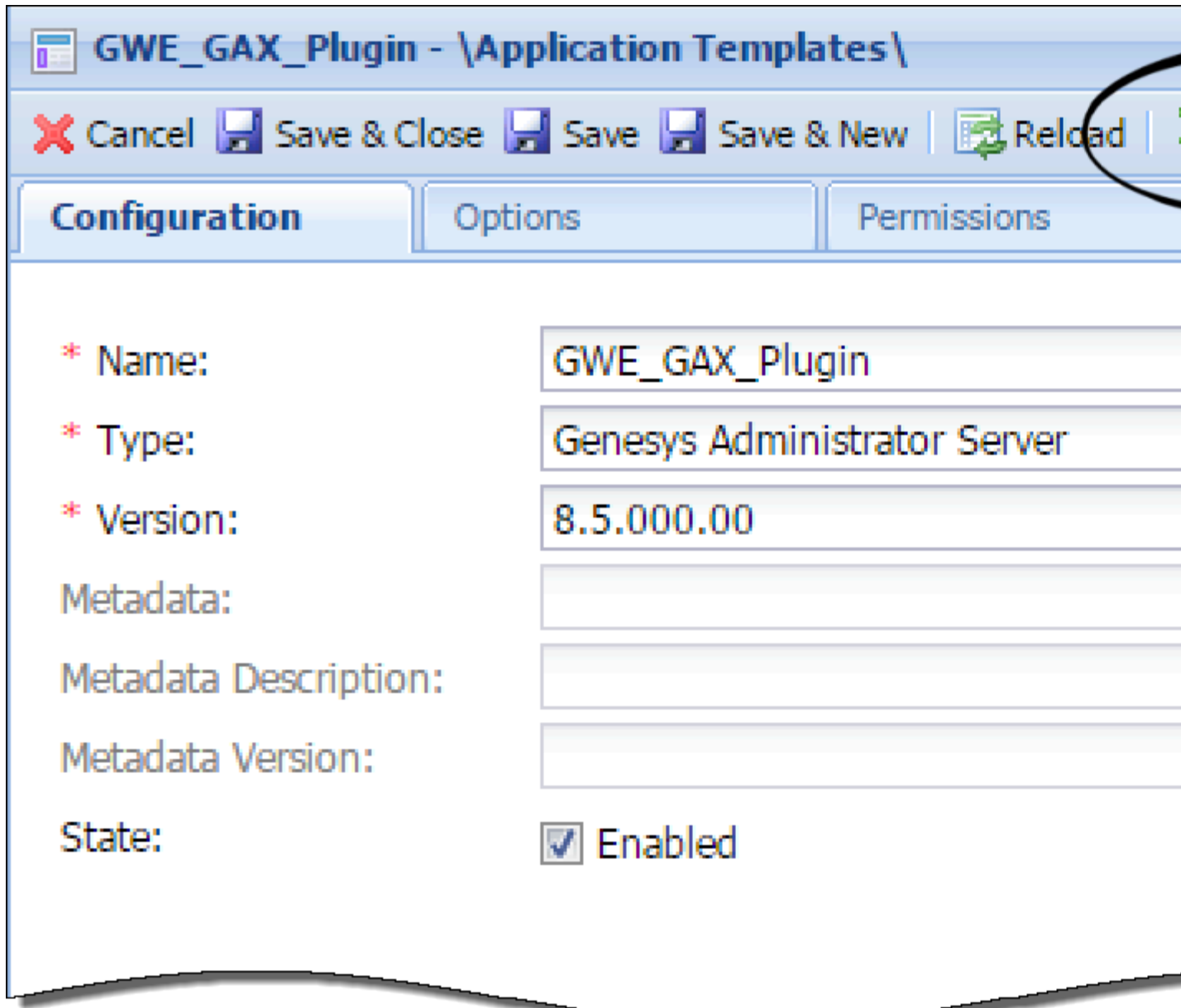
3. Click the **Upload Template** button.



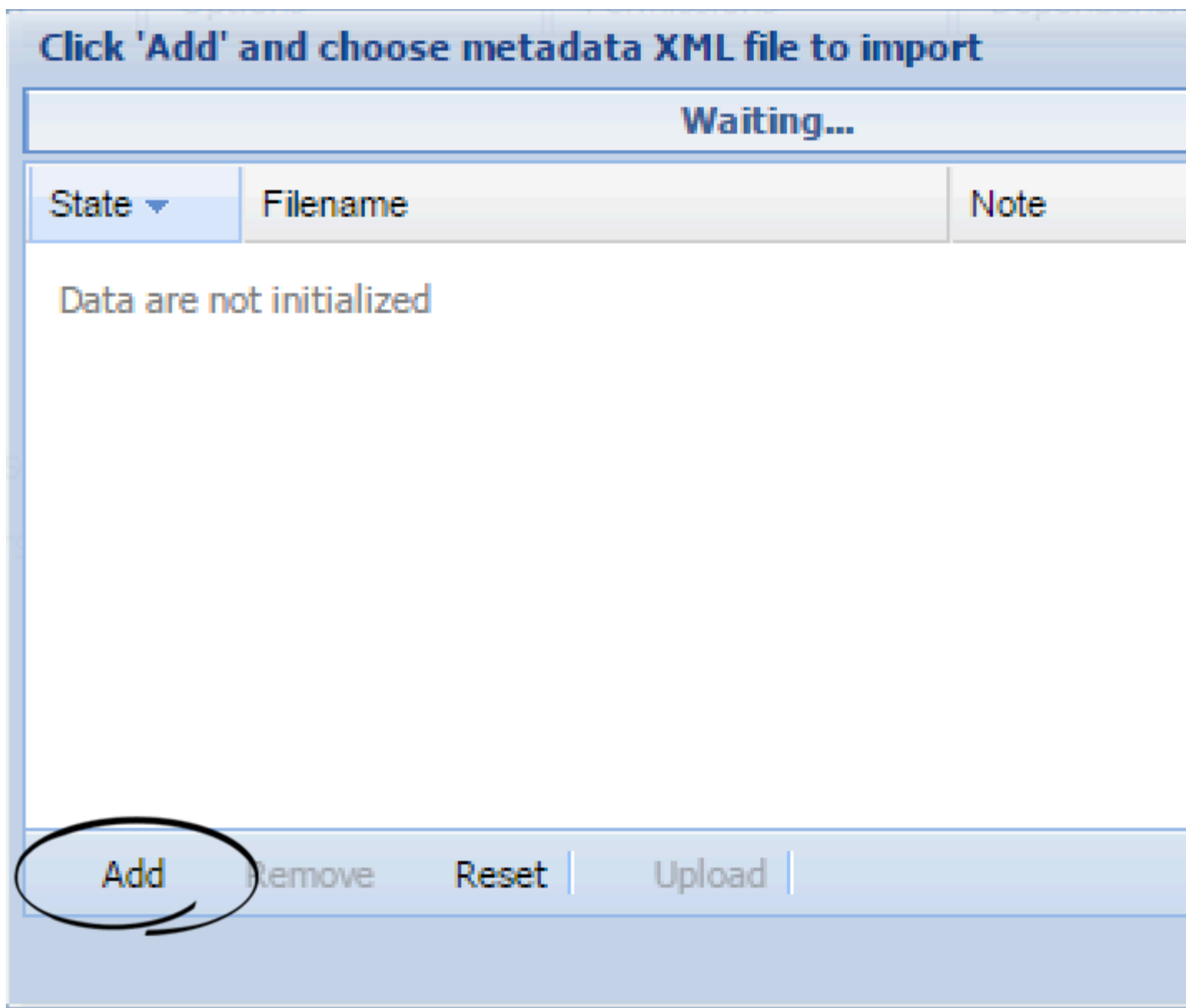
4. Click the **Add** button and chose the path to **Web\_Engagement\_GAX\_Plug-in.apd**, which is located in the IP directory of the Web Engagement Plug-in for Genesys Administrator Extension.



5. Specify the Type as **Genesys Administrator Server** and click **Import Metadata**.

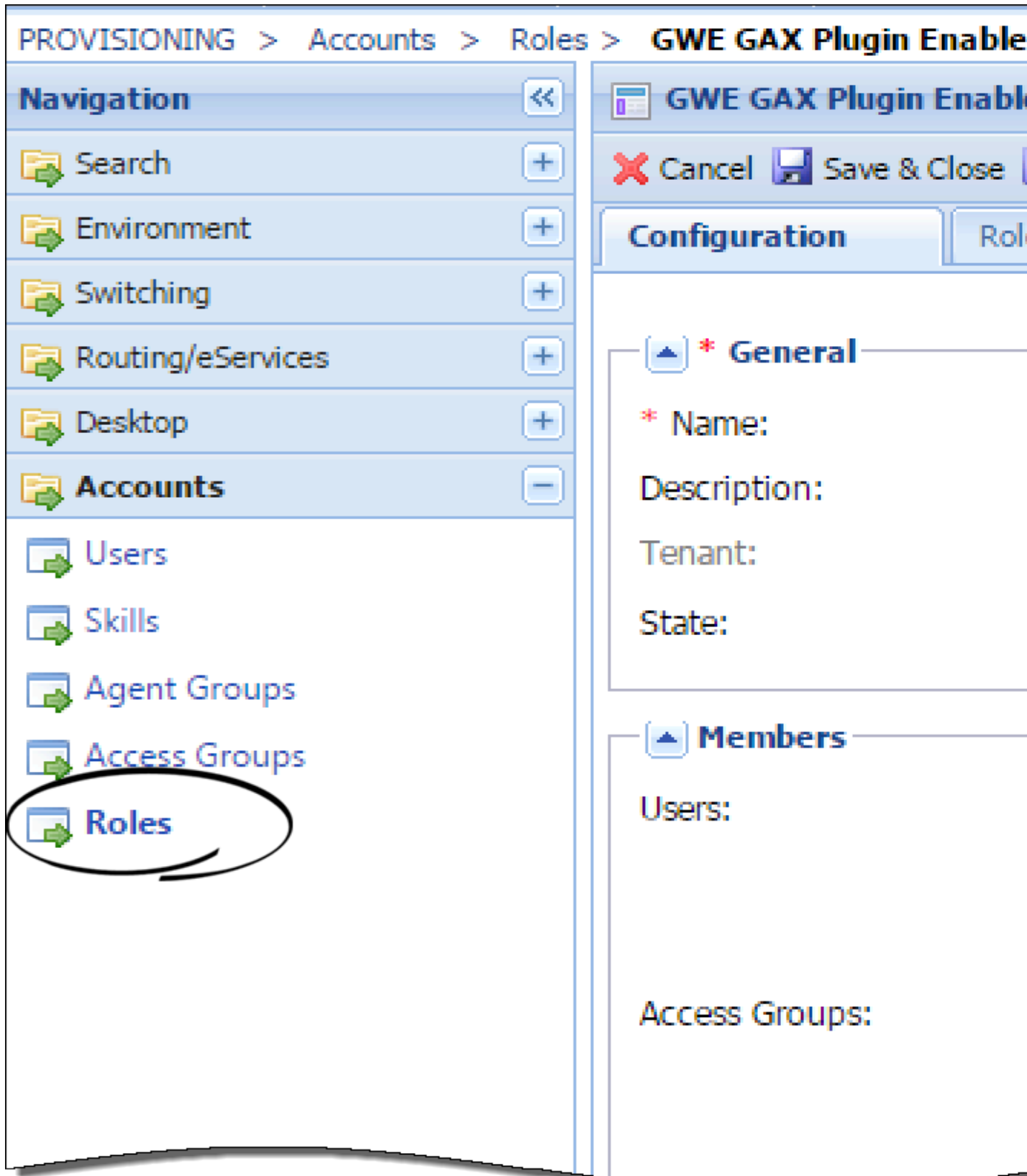


6. Click **Add** and choose the path to **Web\_Engagement\_GAX\_Plug-in.xml**.

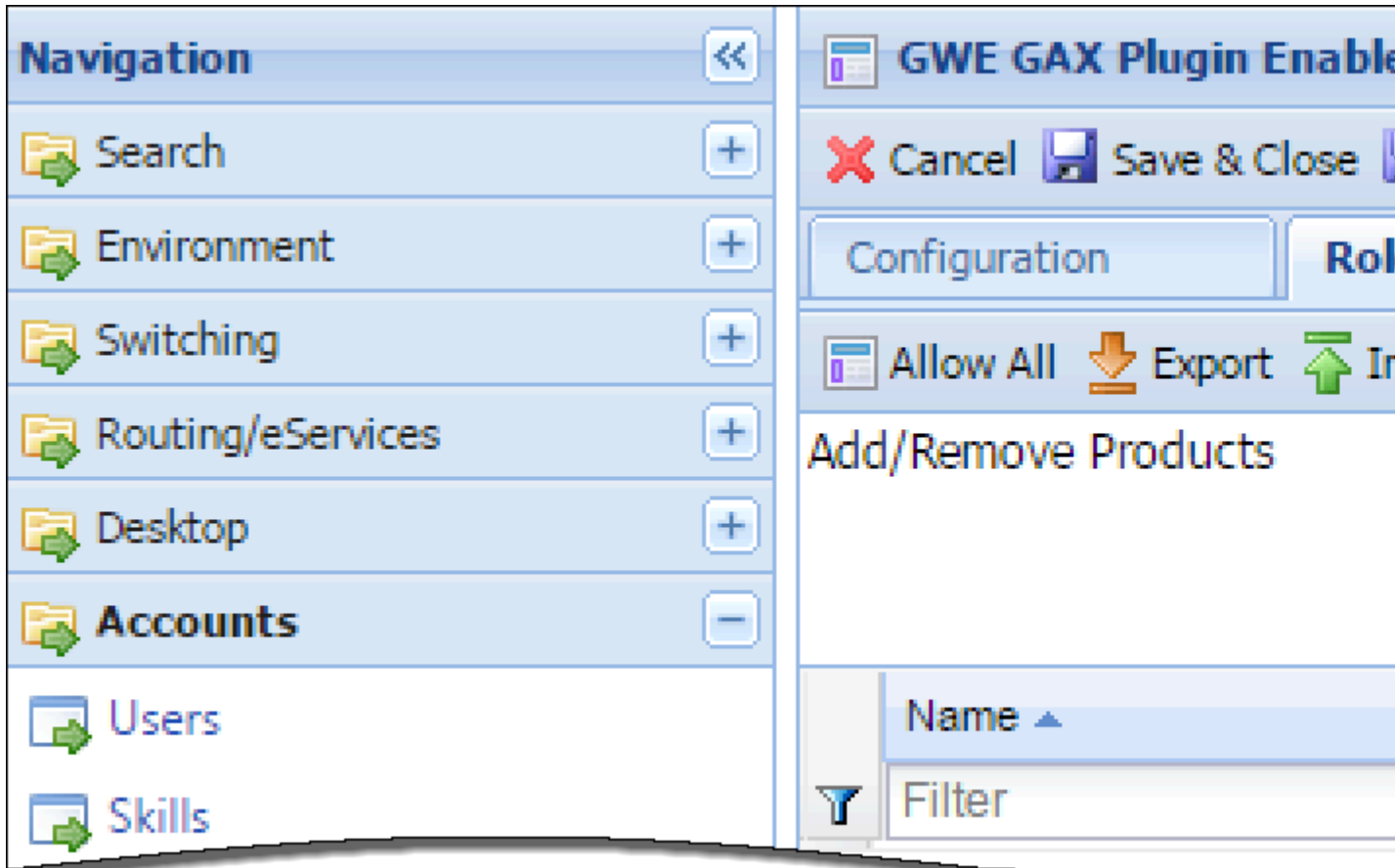


7. When the XML file is imported, click the **Save & Close** button in the **Upload Template** dialog.
8. Go to **Account/Roles** and open your role (or create a new one).





- 9. Go to the **Role Privileges** panel and click the **Genesys Administrator Extension** product to add privileges.



The list of privileges should include **Access to Web Engagement Categories**, **Access to Web Engagement Script Generator**, and (starting with version 8.5.000.33) **Access to Web Engagement Pacing Configuration**. To enable a privilege, set it to Allowed.

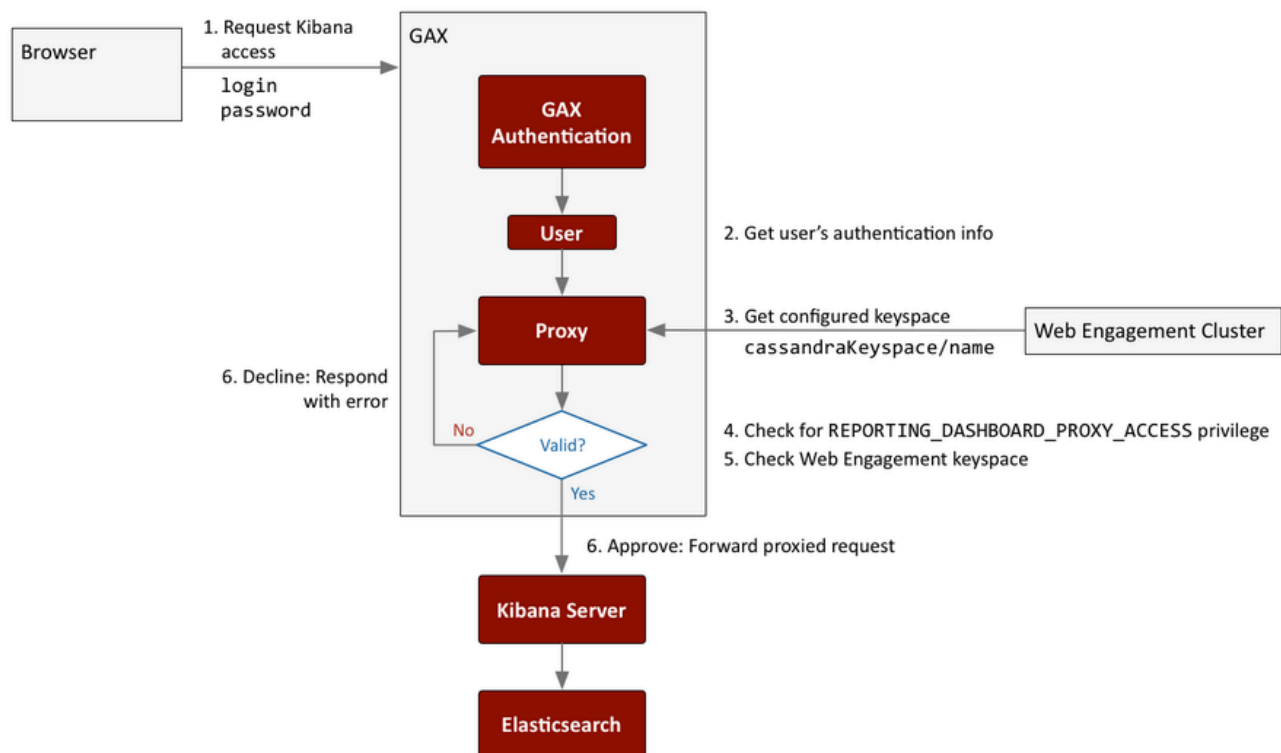
- 10. Click the **Save & Close** button.

# Installing the Reporting Dashboard Plug-in (optional)

Reporting Dashboard Proxy for Genesys Administrator Extension is a GAX plugin that proxies incoming HTTP requests sent from browser-based Kibana clients to the Kibana server, which is controlled by the Web Engagement Server. You must use this plugin when your Extended View reporting dashboards are used in a browser that was launched in a secure zone that is different from the zone containing the Web Engagement Server—for example, if the browser is outside of your VPN in situations where Web Engagement Server is available only through the VPN.

## Authentication

Genesys recommends that you locate database-related components in secure zones, and that you forbid access from less secure zones—especially from the public Internet. This requires the appropriate access restrictions, which must follow a workflow like the one shown here:



As you can see from the diagram, all access to the Kibana server (which is controlled by Web

Engagement server) is routed through the Reporting Dashboard Proxy. Direct access must be prohibited at the network configuration level.

Access requests through the Proxy can be approved or declined, depending on the results of the authentication process. Users can only authenticate successfully if they:

- Are logged in as a GAX user
- Have the **REPORTING\_DASHBOARD\_PROXY\_ACCESS** privilege

To control access to the Kibana server, follow the procedure described at [Tuning Role-Based Access in Genesys Administrator](#), using the **Reporting\_Dashboard\_Proxy\_For\_GAX.apd** template file instead of **Web\_Engagement\_GAX\_Plug-in.apd**.

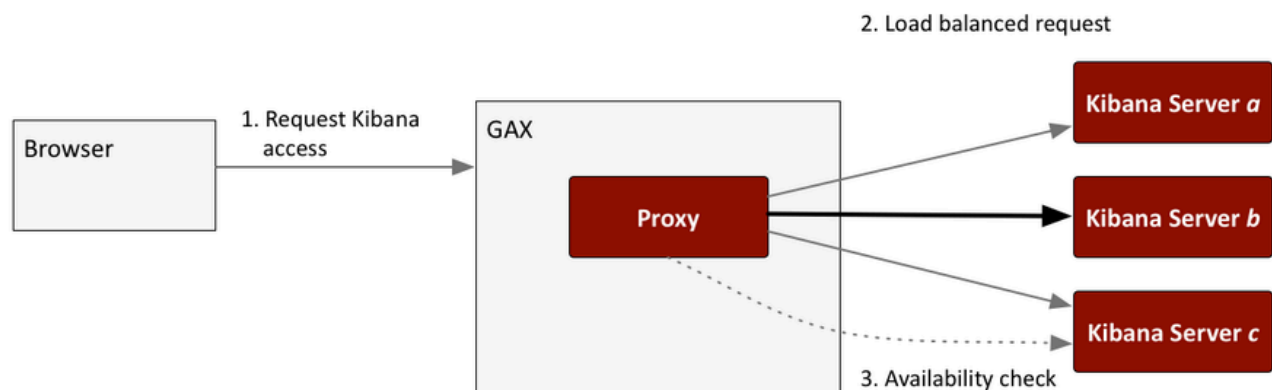
## Keyspace-based authorization

When Web Engagement Server is started, the proxy retrieves its keyspace configuration information and stores it for use during the authorization process. If an access request is authenticated, the proxy compares the keyspace specified in the request body to see if it matches.

As shown in steps **5** and **6** of the Authentication Workflow diagram, if the keyspace matches, the request succeeds. If not, the proxy returns an error.

## Load balancing

The Proxy handles load balancing, forwarding each request to the next available Kibana server. The server URLs are stored in a list, as shown in the table of [configuration options](#). The following diagram shows the load balancing workflow.



## Failover

The proxy checks in with the Kibana servers every 10 seconds, as shown in step **3** of the above load balancing diagram. If a Kibana server doesn't respond, the proxy excludes it from the load balancing process. When the server becomes available again, the proxy reincorporates it into the load balancing process.

## Installation

### Prerequisites

- Your environment includes Genesys Administrator Extension. See [Genesys environment prerequisites](#) for compliant versions. For more information about installing the Genesys Administrator Extension, refer to the [Genesys Administrator Extension Deployment Guide](#).
- Installation must be driven from the host where you intend to install the plug-in.

1. In Genesys Administrator, navigate to Provisioning > Environment > Applications, select the Genesys Administrator Extension server, and stop it.
2. In your installation package, locate and double-click the `setup.exe` file.
3. Click *Next*. The *Choose Destination Location* screen appears.
4. The destination location is the temporary folder where the plug-in files will be unpacked before they are copied into the GAX structure. You can browse to the desired location or use the default location and click *Next*. The *Ready to Install* screen appears.
5. Click *Install*. The Genesys Installation Wizard indicates it is performing the requested operation for the Genesys Reporting Dashboard Proxy. When through, the *Installation Complete* screen appears.
6. Click *Finish* to complete your installation. As a result of the installation, a new file called `gax-plugin-dashboard-proxy-webme.jar` is copied to the `<GAX installation>\webapp\WEB-INF\lib\` folder.
7. In Genesys Administrator, navigate to Provisioning > Environment > Applications, select the Genesys Administrator Extension server, and start it.

## Configuration

Proxy configuration information is stored in the GAX Server application, under the **wcc-kibana-proxy-plugin** section. It includes the following configuration option:

option	description	example
kibana-servers	List of the Kibana nodes that will be used for forwarding	192.168.1.1:5601,192.168.1.1:5602

# Reporting

Starting with Web Engagement release 8.5.000.26, GWE uses [Genesys Data Processing Server](#) for reporting.

If you are using an earlier release of Web Engagement, install [Web Engagement Reporting Server](#).

# Installing Genesys Data Processing Server

Genesys Data Processing Server processes the complex, high-volume data produced by select Genesys products for a variety of uses. When used for reporting, it provides the users of these products with the ability to transform and aggregate this data, and perform sophisticated calculations on it. The resulting metrics are used for reporting and visualization within Genesys Pulse.

The following instructions describe how to install Data Processing Server.

**Note:** Many of the procedures involved in configuring the Genesys Data Processing Server are similar to the ones that are used to configure other application clusters, such as Web Engagement Server. The description of how to [Install the Genesys Web Engagement Server](#) contains information that may be worth consulting while you are configuring Data Processing Server.

**Important:** Genesys recommends that you use a dedicated Cassandra data center for reporting purposes. This will minimize the risk of Cassandra-related faults in operational environments that are running under a heavy load.

**Important:** Genesys recommends that you put each Cassandra node for your reporting data center on the same host as a Data Processing Server node. In other words, the number of nodes in Cassandra reporting data center should be equal to the number of nodes in the Data Processing Server cluster. This will speed up calculations on your analytical data.

## Before you begin

Because Genesys Data Processing Server must process a lot of data, it needs to run on top of a high-speed—and highly scalable—cluster computing system. Genesys has chosen [Apache Spark](#) for this task.

Spark supports several types of clustering. Fortunately, Data Processing Server works well with the simplest one, [Spark Standalone Mode](#). This mode provides high availability by using a dedicated master node. A typical cluster deployment will consist of one master node and several worker nodes and is usually started by Data Processing Server in the background.

Genesys recommends that you configure Data Processing Server in Genesys Administrator, defining a single Application Cluster object and the appropriate number of individual node objects. You can configure these objects and their options using the procedures provided on the rest of this page.

## Recommended versions

Genesys recommends that you use specific Data Processing Server versions with your products. For more information, consult the appropriate page for your product:

- [Genesys Web Engagement](#)

## Data Processing Server nodes

As we just mentioned, the Spark cluster consists of one master node and several worker nodes. Any Data Processing Server node can be the master node in the cluster, as this role is defined by the value of the Spark `startMode` option in the node's configuration options. Here is more information about the two types of node:

- The **Master** node is represented by a Spark `startMode` value of `both`, which indicates that both a Spark Master and a Spark Worker instance will be started on the node. Please note that the Spark `host` option should be set in agreement with `startMode` so that the hostname used in the Spark `host` setting belongs to the node that runs the Spark Master instance. To avoid problems with connectivity inside the Spark cluster, this hostname should be the primary one in the network configuration for this host. In other words, Java's `InetAddress.getLocalHost` should return the same value for the Data Processing Server Master node.
- The **Worker** node is represented by a `startMode` value of `worker`. Only a Spark worker instance will be launched at this node.

There is one additional mode available for the Spark `startMode` option. A mode of `off` means that no Spark processes will be launched on this host and that the role of the node in the Data Processing Server cluster is undefined. This mode is for use in situations where you want to have an externally managed Spark cluster, and limits you to one Data Processing Server node, which serves as an entry point for the Spark cluster. You cannot deploy Data Processing Server with multiple nodes if you have set `startMode` to `off`. Also, if you use this mode, you must have an advanced understanding of how to work with and manage a Spark cluster.

## Configuring Data Processing Server

We have included information about Data Processing Server-related `configuration options` at the end of this page.

## Deploying Data Processing Server

To deploy Data Processing Server, follow these steps:

1. [Importing the Data Processing Server cluster template](#)
  2. [Creating the cluster application](#)
  3. [Configuring the cluster application](#)
  4. [Importing the Data Processing Server template](#)
  5. [Creating a node application](#)
  6. [Configuring a node application](#)
  7. [Tuning analytical Cassandra nodes to skip indexing](#)
  8. [Adding nodes to a cluster](#)
-



9. [Updating your product's cluster application](#)
10. [Data Processing Server data storage](#)
11. [Installing Data Processing Server](#)
12. [Installing Dashboards and Widgets into Pulse](#)
13. [Deploying and Scheduling Job Packages](#)
14. [Data Processing Server data flow](#)

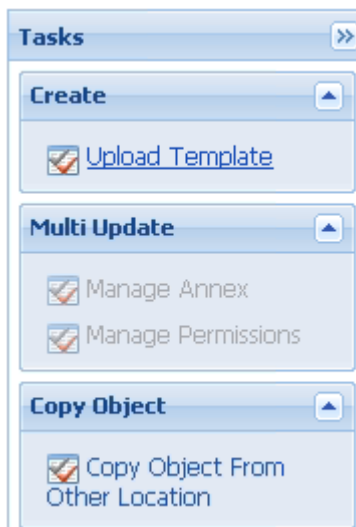
**Note:** For more information on how to work with templates and application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

## Importing the Data Processing Server cluster template

**Note:** For more information on how to work with templates in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Application Templates**.
2. In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.
4. Browse to the **Data\_Processing\_Cluster.apd** file. The **New Application Template** panel opens.
5. Click **Save & Close**.

---

**End**

## Creating the cluster application

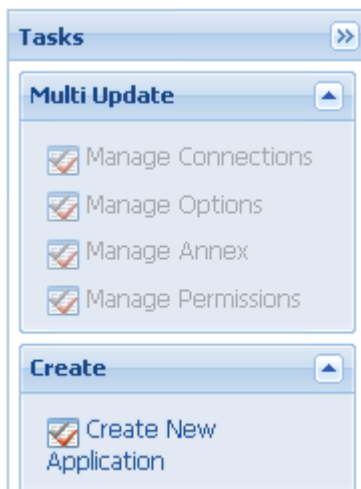
**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Importing the Data Processing Server cluster template](#).

### Start

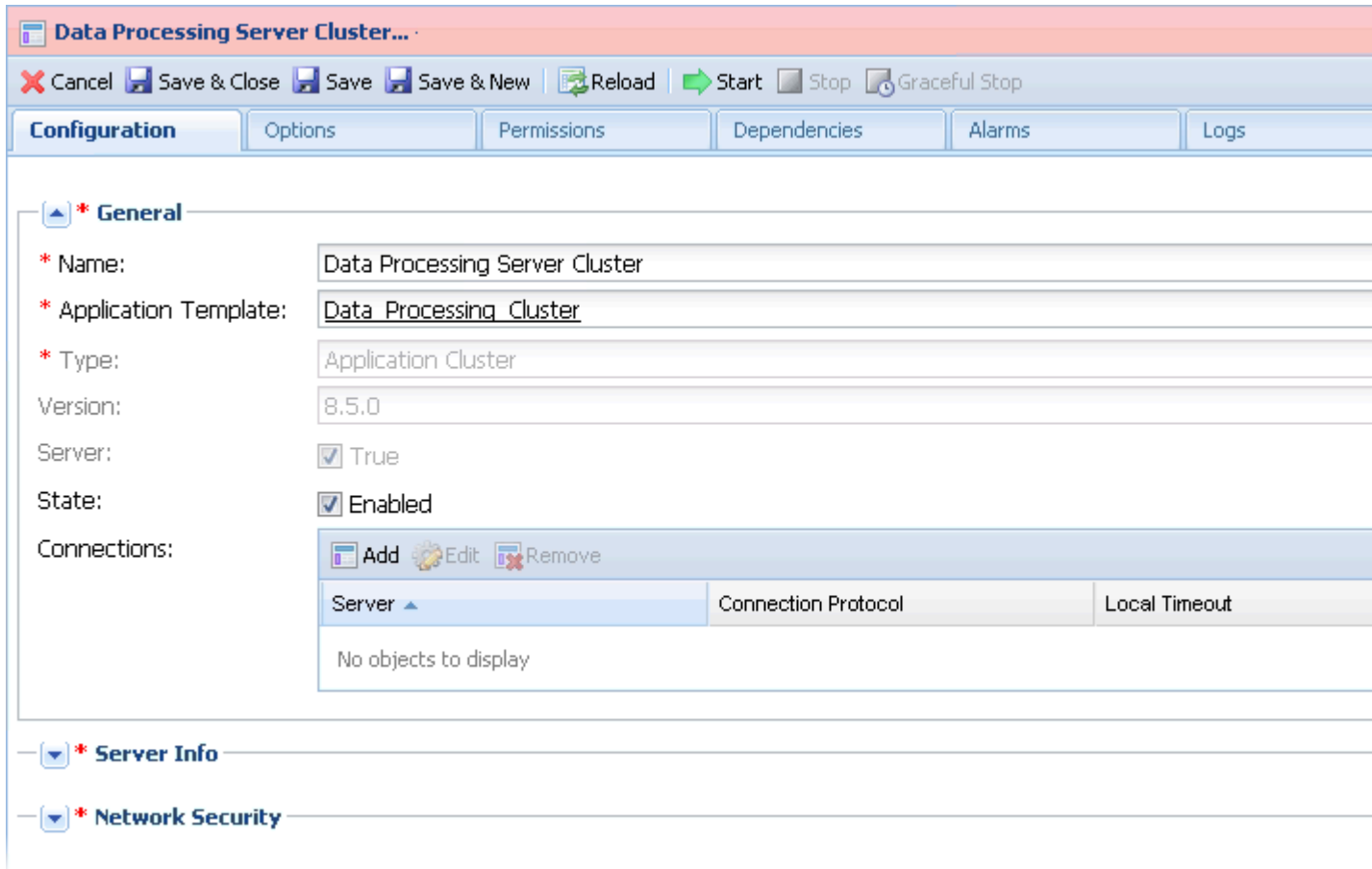
1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.



Create New Application link.

3. In the **Select Application Template** panel, click **Browse for Template** and select the Data Processing Server cluster template that you imported in [Importing the Data Processing Server cluster template](#). Click **OK**.
4. The template is added to the **Select Application Template** panel. Click **Next**.
5. In the **Select Metadata file** panel, click **Browse** and select the **Data\_Processing\_Cluster.xml** file. Click **Open**.
6. The metadata file is added to the **Select Metadata file** panel. Click **Next**.
7. In the **Specify Application parameters** tab:
  - Enter a name for your application. For instance, `Data_Processing_Server_Cluster`.
  - Make sure **State** is enabled.
  - Select the **Host** on which the Data Processing Server cluster will reside.

- Click **Create**.
- The **Results** panel opens.
  - Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The Data Processing Server cluster application form opens and you can start configuring the Data Processing Server cluster application.



Data Processing Server Cluster app opened in Genesys Administrator.

**End**

## Configuring the cluster application

**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Creating the cluster application](#).

## Start

1. If your Cluster application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the Data Processing Server cluster and click **Edit...**
2. Expand the **Server Info** pane.
3. If your **Host** is not defined, click the lookup icon to browse to the hostname of your application.
4. Ensure the **Working Directory** and **Command Line** fields contain "." (period).

Configuration	Options	Permissions	Dependencies	Alarms	Logs
* Working Directory:	.				
* Command Line:	.				
Command Line Arguments:					
* Startup Timeout:	90				
* Shutdown Timeout:	90				
Backup Server:	[Unknown Backup Server]				
* Redundancy Type:	Not Specified				
* Timeout:	10				
* Attempts:	1				
Auto Restart:	<input type="checkbox"/> True				
Log On As SYSTEM :	<input checked="" type="checkbox"/> True				
* Log On Account:	[Unknown Log On Account]				

Commands

5. Click **Save**.
6. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
  - Enter the **Port**. For instance, 10081.
  - Choose http for the **Connection Protocol**.
  - Click **OK**. The HTTP port with the default identifier appears in the list of **Listening ports**.
7. Genesys recommends that you use *external* Cassandra and that you configure it by following the guidelines at [Working with External Cassandra](#), applying the steps on that page to the Data Processing Server cluster instead of your product's server cluster.

Data Processing Server supports the same Cassandra security features as Web Engagement, with the exception of mutual TLS. Refer to the GWE [Cassandra security article](#) for instructions on setting up Cassandra security for GDPS.

However, if you choose to use *embedded* Cassandra, follow these steps:

1. Select the **Options** tab.
2. Make sure that the value of the `clusterName` option in the `[cassandraEmbedded]` section of the Data Processing Server cluster application is the same as the value of the `clusterName` option in your product's cluster application.
3. In the `[cassandraEmbedded]` section, you can take the default values for all of the options except `seedNodes`, which requires a comma-separated list of seed nodes, where each seed node is represented by an IP address or fully qualified domain name (FQDN). For example:
  - 192.168.0.1,192.168.0.3
  - host1.mydomain.com,host2.mydomain.com.

**Important:** The list of seed nodes should include at least one node from each operational and analytical data center.

4. In the `[cassandraKeyspace]` section, you may need to tune the `replicationStrategyParams` option, which by default is set to 'AnalyticalDC':1. This value indicates a replication factor of **1** for nodes that belong to the AnalyticalDC data center. Note that the data center name is important for multi-data center configurations. When using the default endpoint snitch (GossipingPropertyFileSnitch), you can specify the data center name in **Data Processing Server installation folder/resources/cassandra-rackdc.properties**. For more information, refer to <http://docs.datastax.com/en/cassandra/2.2/cassandra/architecture/archsnitchGossipPF.html>.
5. Make sure that the value of the `name` option in the `[cassandraKeyspace]` section of the Data Processing Server cluster application is **not** the same as the value of the `name` option in your product's cluster application. This will ensure that your keyspace settings do not churn—for example, that the TTL properties from an operational keyspace will not be applied to an analytical keyspace.
8. In the `[spark]` section, specify the value of the `host` option using the fully-qualified domain name (FQDN) or IP address for your Data Processing Server.

**Important:** Spark is very sensitive about host names and sometimes even minor network configuration problems may result in cluster connectivity problems. The best way to ensure correct behavior is to verify that the `spark.host` option uses the same name as the Java `InetAddress.getLocalHost().getHostName()` method would return for this host.
9. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.

**End**

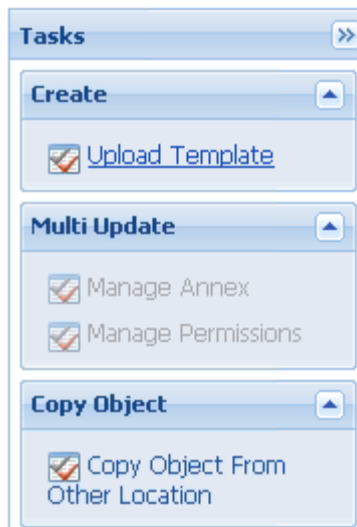
## Importing the Data Processing Server template

### Prerequisites

- You completed [Configuring the cluster application](#).

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Application Templates**.
2. In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.
4. Browse to the **Data\_Processing\_Server.apd** file and select it. The **New Application Template** panel opens.
5. Click **Save & Close**.

**End**

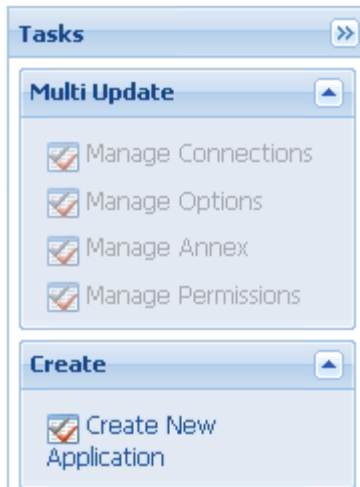
## Creating a node application

### Prerequisites

- You completed [Importing the Data Processing Server template](#).

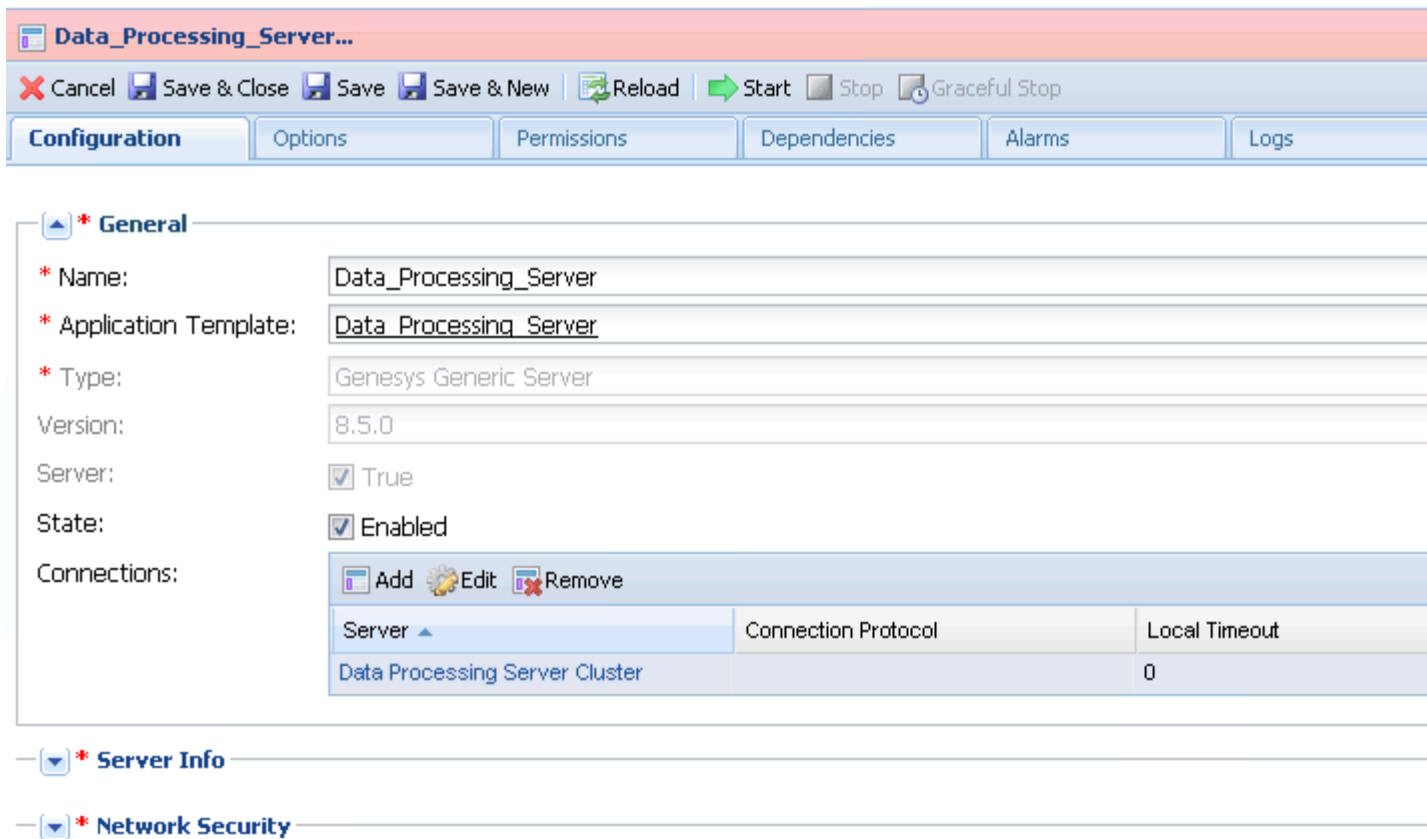
### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.



Create New Application link.

3. In the **Select Application Template** panel, click **Browse for Template** and select the Data Processing Server template that you imported in [Importing the Data Processing Server template](#). Click **OK**.
4. The template is added to the **Select Application Template** panel. Click **Next**.
5. In the **Select Metadata file** panel, click **Browse** and select the **Data\_Processing\_Server.xml** file. Click **Open**.
6. The metadata file is added to the **Select Metadata file** panel. Click **Next**.
7. In **Specify Application parameters**:
  - Enter a name for your application. For instance, `Data_Processing_Server`.
  - Make sure **State** is enabled.
  - Select the **Host** on which the node will reside.
  - Click **Create**.
8. The **Results** panel opens.
9. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.
10. Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The `Data_Processing_Server` application form opens and you can start configuring the node application.



Node app opened in Genesys Administrator.

## End

## Configuring a node application

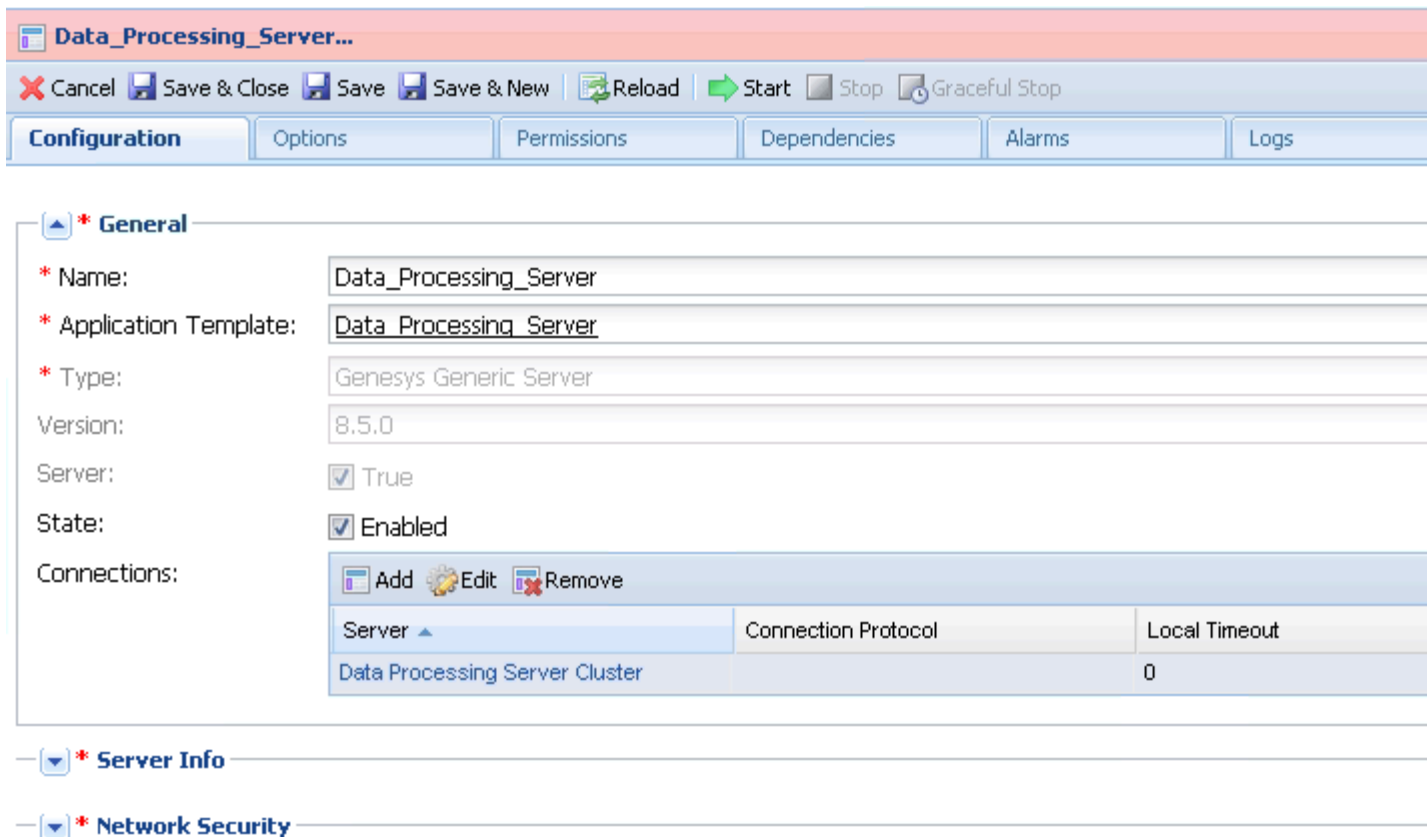
### Prerequisites

- You completed [Creating a node application](#).

### Start

- If your node application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the node and click **Edit...**
- In the Connections section of the **Configuration** tab, click **Add**. The **Browse for applications** panel opens. Select the Data Processing Server cluster application you defined above, then click **OK**.





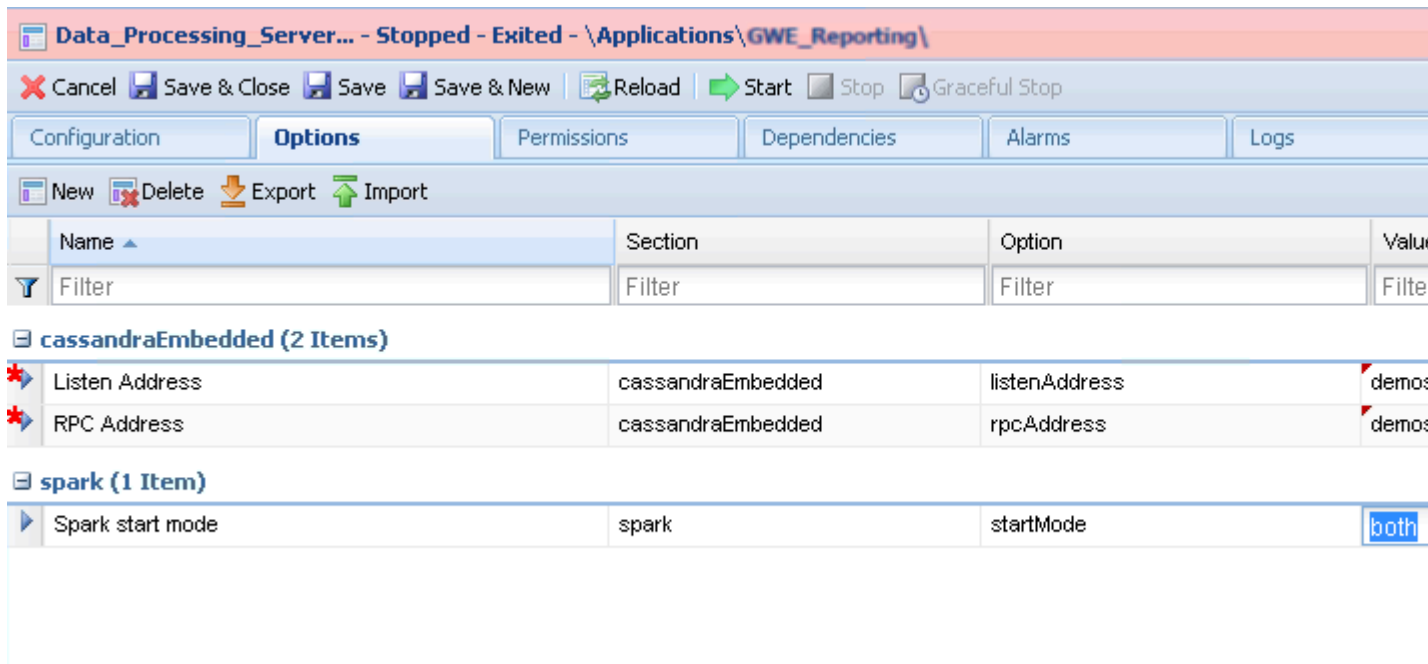
Node connection to Cluster

3. Expand the **Server Info** pane.
4. If your **Host** is not defined, click the lookup icon to browse to the hostname of your application.
5. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
  - Enter the **Port**. For instance, 10081.
  - Choose http for the **Connection Protocol**.
  - Click **OK**. The HTTP port with the default identifier appears in the list of **Listening ports**.
6. Click **Save**.
7. Genesys recommends that you use *external* Cassandra and that you configure it by following the guidelines at [Working with External Cassandra](#), applying the steps on that page to the Data Processing Server cluster instead of your product's server cluster.
 

However, if you choose to use *embedded* Cassandra, select the **Options** tab.

  - In the **View** field, select **Advanced View (Options)**.
  - In the **[cassandraEmbedded]** section, set the values for **listenAddress** and **rpcAddress**, using a fully qualified domain name or the appropriate IP address.
8. In the **[spark]** section, set the value for **startMode**. If you are configuring a master node, set this value to **both**. For other nodes, set it to **worker**.

**Note:** You should only have one master node configured for your Data Processing Server cluster. If you have a single-node cluster, then your single node must be configured as a master node.



Data Processing Server node options

## End

## Tuning analytical Cassandra nodes to skip indexing

For performance reasons, Web Engagement Server's *operational* Cassandra nodes use custom Cassandra indexes that rely on the services of Elasticsearch. However, the *analytical* Cassandra nodes used by Data Processing Server do not require those indexes, as all analytical reads of Cassandra tables are full scan reads. Because of this, Genesys recommends that these indexes be removed from your analytical nodes.

If you are using embedded Cassandra, Web Engagement will automatically avoid creating these indexes. But if you are using external Cassandra, you must carry out the following procedure to get rid of them.

**Note:** You must do these steps *before* you replicate the Cassandra data from your operational data centers.

## Start

1. Copy the required libraries to the **Cassandra lib folder** for each Cassandra node in your analytical data centers.
2. Modify your Cassandra startup scripts to include the **genesys-es-dummy** system property

- On Windows, append the following line to **bin/cassandra.in.bat**:

```
set JAVA_OPTS=%JAVA_OPTS% -Dgenesys-es-dummy=true
```

- On Linux, append the following line to **bin/cassandra.in.sh**:

```
export JVM_OPTS="$JVM_OPTS -Dgenesys-es-dummy=true"
```

## End

## Adding nodes to a cluster

To create more nodes:

### Start

1. Follow the instructions above for [Creating a node application](#), but use a different name for the new node.
2. [Configure the new node application](#), as shown above, but point to a different port.

### End

## Updating your product's cluster application

Genesys recommends that you use a dedicated data center for reporting. In order to do this, you must do the following to your product's cluster application:

- Modify the [seedNodes](#) option in the [\[cassandraEmbedded\]](#) section so that it is in sync with the **seedNodes** option in the Data Processing Server Cluster application.
- Modify the [replicationStrategyParams](#) option in the [\[cassandraKeyspace\]](#) section so that it includes replication to the reporting data center. For example:

```
'OperationalDC':1,'AnalyticalDC':1
```

**Important:** You only need to do this to your product's cluster application. Note also that your Data Processing Server Cluster application should specify replication strategy parameters for the corresponding analytical (reporting) data center only. In other words, all of the data that comes into the analytical data center should be left there, rather than being propagated to an operational data center.

**Important:** If you have more than one operational data center, then you must replicate **each** of your operational data centers to analytical data centers.

## Data Processing Server data storage

Data Processing Server stores several types of information:

- Aggregated data results
- General configuration data
- Tenant-specific configuration data
- Default Pulse dashboards and widgets
- Meta-information

All of this information is stored in a database layer that is indexed by Elasticsearch. By default, you can access the reporting database layer via HTTP, using the URL of the correctly configured Load Balancer:

- For embedded Cassandra, the Load Balancer should redirect requests to **one of your product's hosts** on port 9200 (or the port ID you have specified in the **http.port** option of the [\[elasticsearch\]](#) section of your product's cluster application).
- For external Cassandra, the Load Balancer should redirect requests to one of the **Cassandra hosts** used by your product (Operational DC) on port 9200 (or the port ID you have specified in the **http.port** option of the [\[elasticsearch\]](#) section of your product's cluster application).

We will refer to this URL as *the Reporting Data URL*. When you send your browser or HTTP client requests to the Reporting Data URL, you should receive HTTP Response 200.

## Installing Data Processing Server

Install the Data Processing Server on Windows or Linux.

**Note:** For more information on how to install apps that you have configured in Genesys Administrator, consult [Generic Installation Procedures](#).

### The Pulse Collector

Data Processing Server uses data gathered by a Pulse Collector. This Pulse Collector must only be installed on one node in the Data Processing Server cluster.

**Although the procedures in the next section tell you how to set up your initial Pulse Collector installation, if you decide later that you want to install the Pulse Collector on a different node, you must follow these extra steps:**

- Turn the collector off at the node it was originally installed on:
-

- Stop the node
- Set the **PULSE\_COLLECTOR\_ENABLED** variable in your **setenv.bat** or **setenv.sh** file to false
- Remove the **pulse-collector.war** file from the **webapps** folder
- Restart the node
- Turn the collector on at another node:
  - Stop the node
  - Set the **PULSE\_COLLECTOR\_ENABLED** variable in your **setenv.bat** or **setenv.sh** file to true
  - Set the value of **DATA\_PROCESSING\_ES\_URL**—also in your **setenv.bat** or **setenv.sh** file—to the value of your [Reporting Data URL](#)
  - Copy the **pulse-collector.war** file from the **etc** folder to the **webapps** folder
  - Restart the node

## Windows

### Prerequisites

- [Configuring a node application](#)
- A supported JDK is installed. See [Java Requirements](#) for details.

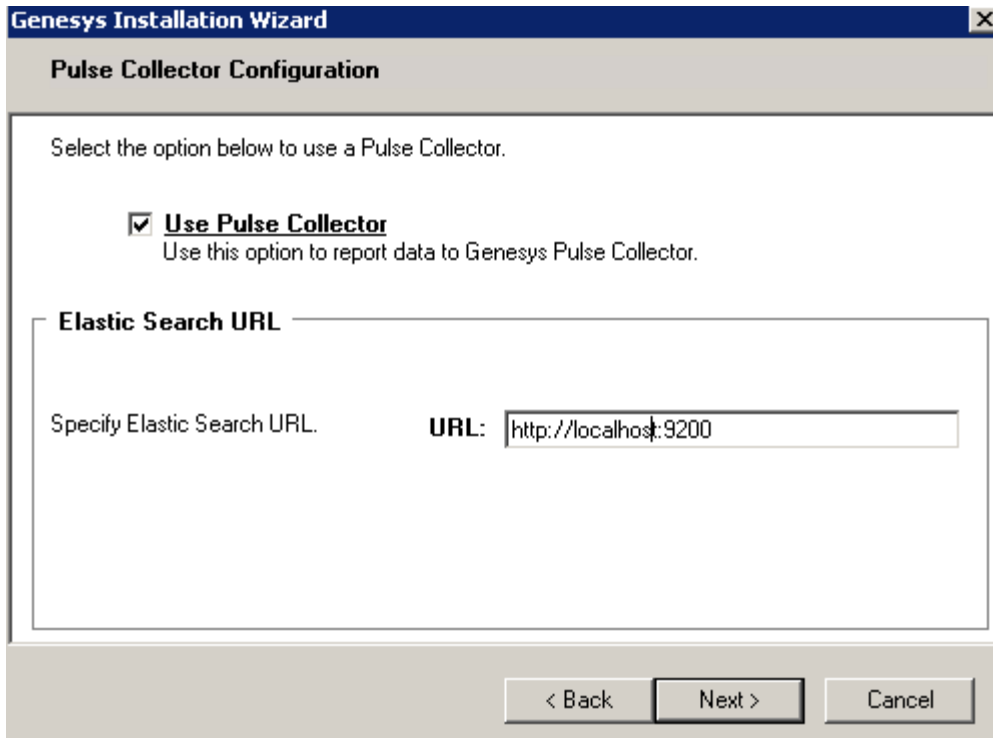
### Start

1. In your installation package, locate and double-click the **setup.exe** file. The Install Shield opens the welcome screen.
  2. Click **Next**. The **Connection Parameters to the Configuration Server** screen appears.
  3. Under **Host**, specify the host name and port number where Configuration Server is running. (This is the main "listening" port entered in the **Server Info** tab for Configuration Server.)
  4. Under **User**, enter the user name and password for logging on to Configuration Server.
  5. Click **Next**. The **Select Application** screen appears.
  6. Select the Data Processing Server Application—that is, the Node app you created above—that you are installing. The **Application Properties** area shows the **Type**, **Host**, **Working Directory**, **Command Line executable**, and **Command Line Arguments** information previously entered in the **Server Info** and **Start Info** tabs of the selected Application object.  
**Note:** For multi-node clusters, you must install the Data Processing Server Application into exactly the same directory on every node. For example, if the path for Node 1 is /genesys/gdps/gdps\_n1, it cannot be /genesys/gdps/gdps\_n2 for any of the other nodes. This requires manual intervention, since the installation package offers a default installation path based on the application name, which is therefore different for each node.
  7. Click **Next**. The **Choose Destination Location** screen appears.
  8. Under **Destination Folder**, keep the default value or browse for the desired installation location. Note that you probably do not want to use the Windows Program Files folder as your destination folder.
  9. Click **Next**. The **Backup Configuration Server Parameters** screen appears.
-

10. If you have a backup Configuration Server, enter the **Host name** and **Port**.
11. In the **Pulse Collector Configuration** window, select **Use Pulse Collector**:

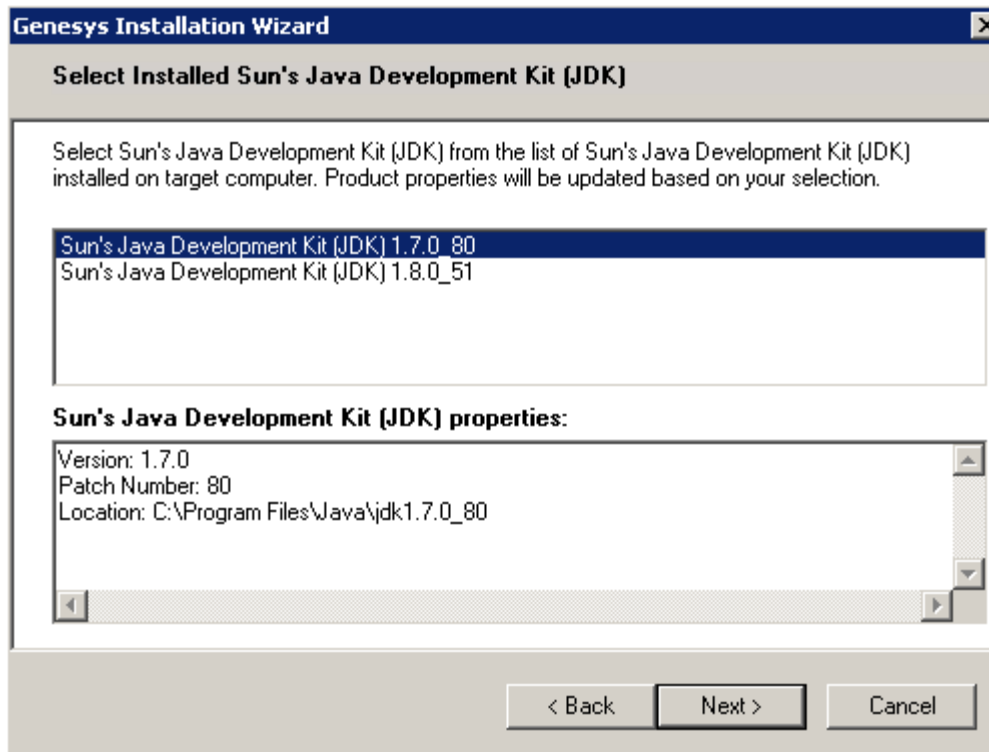


12. In the resulting **Elasticsearch URL** field, specify the host and port of the Data Processing Server cluster node on which you want to install the Pulse Collector. **Note:** You should only install the Pulse Collector on one node at a time, as mentioned [above](#).



Pulse Collector Host and Port

13. Click **Next**. Select the appropriate JDK:



Select JDK

14. Click **Next**. The **Ready to Install** screen appears.
15. Click **Install**. The Genesys Installation Wizard indicates it is performing the requested operation for Data Processing Server. When through, the **Installation Complete** screen appears.
16. Click **Finish** to complete your installation of the Data Processing Server.
17. Inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.
18. **Apply the parameters to your Windows service.**

## End

**Note:** Genesys recommends that you frequently clear the Spark temporary directory—for example, once a week or before you start Data Processing Server. You can find it in the system temporary directory with a name template of spark-\*. The default location for this directory is **system\_disk:\Users\user\_name\AppData\Local\Temp** directory. You can also use the system disk clean-up procedure.



## Linux

### Prerequisites

- [Configuring a node application](#)
- A supported JDK is installed. See [Java Requirements](#) for details.

### Start

1. Open a terminal in the Genesys Web Engagement CD/DVD or the Genesys Web Engagement IP, and run the **install.sh** file. The Genesys Installation starts.
2. Enter the hostname of the host on which you are going to install.
3. Enter the connection information to log in to Configuration Server:
  - The hostname. For instance, `demosrv.genesyslab.com`.
  - The listening port. For instance, `2020`.
  - The user name. For instance, `demo`.
  - The password.
4. If you have a backup Configuration Server, enter the **Host name** and **Port**.  
If the connection settings are successful, a list of keys and Web Engagement applications is displayed.
5. Enter the key for the Data Processing Server application—that is, the Node app you created above in Configuration Server.
6. Use the key for Genesys Pulse to indicate whether or not to enable the Pulse Collector
7. If you have enabled the Pulse Collector, enter the Reporting Data Elasticsearch URL.
8. Enter the location where Data Processing Server is to be installed on your web server.  
**Note:** This location must match the previous settings that you entered in Configuration Server.  
**Note:** For multi-node clusters, you must install the Data Processing Server Application into exactly the same directory on every node. For example, if the path for Node 1 is `/genesys/gdps/gdps_n1`, it cannot be `/genesys/gdps/gdps_n2` for any of the other nodes. This requires manual intervention, since the installation package offers a default installation path based on the application name, which is therefore different for each node.
9. If the installation is successful, the console displays the following message:  
Installation of Genesys Data Processing Server, version 8.5.x has completed successfully.
10. Inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.

### End

**Note:** Genesys recommends that you frequently clear the Spark temporary directory—for example, once a week or before you start Data Processing Server. You can find it in the system temporary directory with a name template of `spark-*`. The default location for this directory is **/tmp**.

---

## Installing dashboards and widgets into Pulse

At this point, you must follow the instructions for installing your product's Pulse dashboards and widgets:

- [Genesys Web Engagement](#)

## Deploying and scheduling job packages

After you have installed **and started** the Data Processing Server, you need to deploy and schedule jobs for it to process.

To do this, execute the following two scripts that are provided in **Data Processing Server Installation directory/deploy/package**.

### Important

- You must install [curl](#) in order to use these scripts.
- You must run the scripts on the master node.

Once you have installed [curl](#), make sure it's available from **Data Processing Server Installation directory/deploy/package**, and run the following scripts:

1. **deploy-package.bat** or **deploy-package.sh** deploys your jobs and requires the URL (host and port) for your Data Processing Server.

If the script executes successfully, you will receive the following response: {"status" : "Ok"}

Here is a sample command for Windows:

```
deploy-package.bat reporting-host:10081
```

And one for Linux:

```
GDPS_URL=http://reporting-host:10081 ./deploy-package.sh
```

2. **schedule-package.bat** or **schedule-package.sh** schedules your jobs and requires:

- The URL (host and port) for your Data Processing Server
- The [Reporting Data URL](#)
- The name of the Cassandra keyspace that is used by your product. In the case of Web Engagement, for example, this is the [keyspace name](#) described in the [\[cassandraKeyspace\]](#) configuration option section.

If the script executes successfully, you will receive the following response: {"<schedule\_guid>" : "Job scheduled"}

Here is a sample command for Windows:

```
schedule-package.bat reporting-host:10081 gwe-cluster-lb:9200 gpe
```

And one for Linux:

```
GDPS_URL=http://reporting-host:10081 ES_URL=http://gwe-cluster-lb:9200 KEYSACE=gpe
./schedule-package.sh
```

In this sample, we can see that:

- Data Processing Server is running on reporting-host at port 10081.
- The Load Balancer running on gwe-cluster-lb at port 9200 represents a Reporting Data URL.
- The GWE keyspace name is gpe.

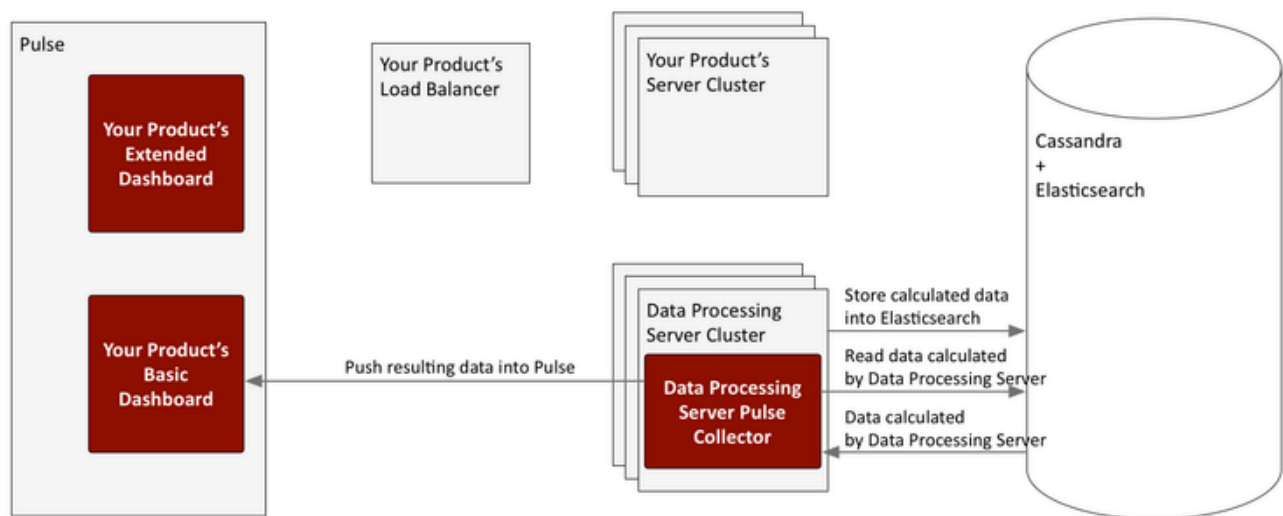
## Data Processing Server data flow for reporting

Data Processing Server supports two types of Pulse reporting:

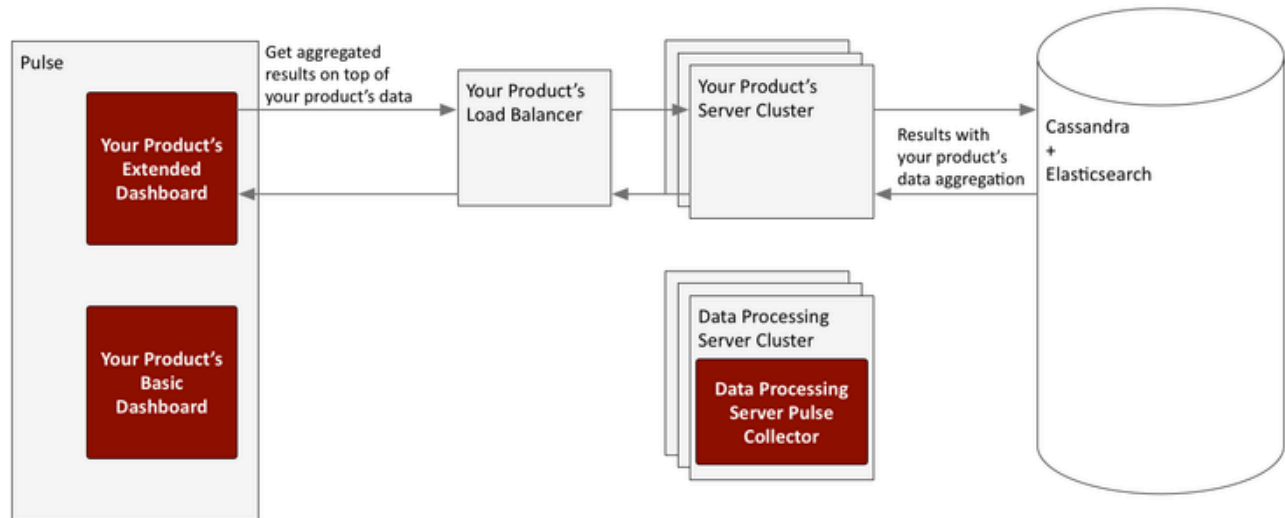
- The **basic views**, also known as the basic dashboards, are native Pulse widgets that are driven by a Pulse collector which is included with the Data Processing Server. These widgets can be used in other dashboard views created in Pulse.
- The **extended views (dashboards)** are bespoke metric views of your product's data sources hosted directly within Pulse. These widgets cannot be used with other Pulse dashboards.

The following diagrams show the data flow for each type of reporting.

### Basic dashboard



## Extended dashboard



## Configuration options

These configuration option sections can be useful in setting up Data Processing Server and your Spark cluster.

### log

The `[log]` configuration options are applied to the Data Processing Server environment in a way that is similar to how they are used with Web Engagement Server.

### cassandraEmbedded

If you have set up your Data Processing Server with an embedded Cassandra node, the `[cassandraEmbedded]` section provides its configuration options. Their meanings when applied to the Data Processing Server environment are similar to the way they are used with Web Engagement Server.

### cassandraKeyspace

Data Processing Server stores some of its data (packages and schedule) in a dedicated Cassandra keyspace. The `[cassandraKeyspace]` section provides its configuration options. All of the options mean pretty much the same as they do when used with Web Engagement Server, although some of their values—such as for the **name**—will be different.

## spark

Data Processing Server launches a dedicated Spark cluster and all of the Data Processing Server nodes need to share the coordinates of the Spark Master node. In addition to this, each individual node has options that can be used to configure the mode with which Spark starts on its box.

### host

**Description:** The name of the Spark Master host. The value should be the same as what Java's `InetAddress.getLocalHost()` would return for the specified host.

**Default Value:** None

**Valid Values:** *hostname of the Spark Master node*

**Mandatory:** No

**Changes Take Effect:** After start/restart

### port

**Description:** The port number of the Spark Master host.

**Default Value:** 7077

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

### startMode

**Description:** The mode that will be used when starting Spark. If set to off, Spark will not be started by Data Processing Server, and will instead have its state managed externally. If set to worker, only a worker node will be started. If set to both, both a worker node and a master node are started. **Note:** Genesys recommends that you set this option for each node to clearly specify the role. However, you can set the Cluster object to worker mode and override that value for the master node by setting that node to both.

**Default Value:** worker

**Valid Values:** off, worker, or both

**Mandatory:** No

**Changes Take Effect:** After start/restart

### masterWebPort

**Description:** The number of the TCP port that the Spark Master web UI will listen on. Note that this option is provided for cases when the default port has already been used by another service.

**Default Value:** 8080

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

### workerWebPort

**Description:** The number of the TCP port that the Spark Worker web UI will listen on. Note that this

---

option is provided for cases when the default port has already been used by another service.

**Default Value:** 8081

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### executorMemory

**Description:** Use this option to manage the amount of memory used by Spark for executing tasks on each node. Genesys recommends at least two gigabytes per node, but more memory can improve performance if hardware allows. For information about the format, consult the Spark documentation.

**Default Value:** None

**Valid Values:** Valid memory limit

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### sparkHeartbeatTimeout

**Description:** The timeout value in seconds between two heartbeat calls to the Spark metrics API.

**Default Value:** 60

**Valid Values:** Positive integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### sparkStartTimeout

**Description:** The timeout value in seconds between a Spark start or restart and the first time its API is checked. On slower machines, it makes sense to increase this value so that Spark has enough time to start successfully (without initiating a restart cycle).

**Default Value:** 20

**Valid Values:** Positive integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### uri

**Description: Advanced.** For situations when Spark is running externally, you must set the URI instead of the host and port. The URI must include the protocol, in addition to the host and port.

**Default Value:** None

**Valid Values:** Valid Spark URI

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### spark.context

**Advanced.** This entire section is copied into **SparkContext**, so it can be used to tune the Spark options. You must have an in-depth understanding of Spark configuration if you are going to use this section.

---

# Using Web Engagement with Data Processing Server

Web Engagement uses Genesys Data Processing Server for reporting, so that you can digest and visualize the complexities of customer and agent activity.

Most of the steps required to install and configure Data Processing Server are carried out on **Data Processing Server itself**. This page contains the small amount of supplementary information you need on the Web Engagement side.

## Recommended versions

Genesys recommends that you use specific Data Processing Server and Web Engagement Server versions together, as shown in the following table:

Data Processing Server	Web Engagement Server	Supported Pulse Version
8.5.000.20+	8.5.000.23+	8.5.102.xx
8.5.000.29	8.5.000.33, 8.5.000.34	8.5.104.xx, 8.5.105.xx
8.5.000.30+	8.5.000.36+	8.5.104.xx, 8.5.105.xx

## Installing dashboards and widgets into Pulse

As you are installing and configuring Genesys Data Processing Server, you will reach a point where you are prompted to install your Pulse dashboards and widgets. Follow these instructions and then return to the Data Processing Server installation page.

### Introduction

Web Engagement comes with a configuration tool that is located in **Web Engagement installation directory/tools/pulse**. You can use this tool to install your dashboards and widgets into Pulse in two different ways:

- **Default configuration**—Only requires a few parameters that drive an automated configuration process
- **Manual configuration**—Requires more extensive inputs, but provides more advanced configuration options

**Note:** If you need to import Pulse templates, you must set the **http-read-only** option to false to make Elasticsearch available for writing.

## Prerequisites

- The **enabled** option in the **[kibana]** section of the Web Engagement cluster application is set to true.
- Web Engagement Server is running.
- The **Reporting Data URL** is accessible.
- If you have changed Pulse-related configuration data in the manual configuration mode or upgraded your widgets in the default configuration mode, you must **remove your Pulse templates** so Data Processing Server can create new ones.

## Default configuration

1. Run the default configuration scenario with **gdps-config-tool.jar**:
  - a. Go to **Web Engagement installation directory/tools/pulse**.
  - b. Execute

```
java -jar gdps-config-tool.jar <Reporting Data URL> <gax_host> <gax_username>
<gax_password> <Web Engagement data URL>
```

Where <Web Engagement data URL> is the address of the load balancer that redirects requests to the Web Engagement Server on the port specified by the **port** option in the **[kibana]** section of the Web Engagement cluster application.

**Note:** If you are using **Reporting Dashboard Proxy for Genesys Administrator Extension**, use `http://GAX Host:GAX Port/gax/api/wcc-kibana-proxy` to force all Kibana requests through the proxy.

For example,

```
java -jar gdps-config-tool.jar http://gwe-cluster-lb:9200 http://gax-host:8040 default
password http://gwe-cluster-lb:5601
```

**Note:** The default scenario uses a predefined index name. To use a custom index name for your tenant run **gdps-config-tool.jar** with **-Dalias.name=Web\_Engagement\_keyspace\_name**, where *Web\_Engagement\_keyspace\_name* is the value of the **name** option from the **[cassandraKeyspace]** section of the Web Engagement cluster application. For example:

```
java -Dalias.name=gwekeyspace -jar gdps-config-tool.jar http://gwe-cluster-lb:9200
http://gax-host:8040 default password http://gwe-cluster-lb:5601
```

**Note:** By default, **gdps-config-tool.jar** creates a new system configuration when it is first run, but does not overwrite the system configuration if you run it again. To overwrite an existing system configuration, use the **overwriteSystem** flag. For example:

```
java -jar gdps-config-tool.jar http://gwe-cluster-lb:9200 http://gax-host:8040 default
password http://gwe-cluster-lb:5601 overwriteSystem
```

**Note:** For improved security, Genesys recommends that you use the local **es-http-proxy** instead of accessing ElasticSearch directly. In situations where the GDPS Pulse Collector is running on the same host as ElasticSearch, you can set the **es-http-proxy** URL to `http://<elastic_search_node>:<port>`. By default this URL is set to `http://localhost:9292`. For example:

```
java -jar gdps-config-tool.jar http://localhost:9292 http://localhost:8040 default
password http://localhost:5601
```

2. If the configuration process was successful, the console will display **DONE**.
 

**Note:** If an error has occurred, you can try to run the command again, as there are occasions when the configuration tool can't access an http resource, or when an http response times out.
3. Notify Data Processing Server about the configuration changes using one of these methods:
  - a. Send a GET request:



```
http://<Data Processing Server Host>:<Data Processing Server Port>/pulse-collector/  
gdps/configuration/init
```

For example,

```
http://example.com:9999/pulse-collector/gdps/configuration/init
```

If the response is Collector has been re-initialized with new configuration, the new configuration was applied successfully.

- b. Reboot Data Processing Server.

**Note:** You do not need to reboot Genesys Administrator Extensions Server, Pulse, or the Web Engagement Server after you have installed your Web Engagement-specific dashboards and widgets. All of the new widget templates, dashboards, and related elements will be added automatically.

## Manual configuration

For manual configuration, run **gdps-config-tool.jar** with **-Dindex.tool.mode=true**.

For more information on how to do this, run `java -Dindex.tool.mode=true -jar gdps-config-tool.jar`.

### Example 1

Export dashboards and configuration to the **./exported** directory, as shown in this example for Web Engagement:

```
java -Dindex.tool.mode=true -Dworking.dir="./exported" -Des.import.config=true -jar gdps-  
config-tool.jar export gpe.kibana http://gwe-host-lb:9200
```

### Example 2

Import the collector configuration for a new tenant, in the **newtenant** keyspace:

1. Create a JSON file with tenant-related configuration information in the **./config** folder:

```
./config/gdps.collector-tenantConfiguration.json  
{  
  "tenantAliasName": "newtenant",  
  "gaxUrl": "http://gaxurl:8040",  
  "gaxUser": "gaxUser",  
  "gaxPass": "gaxPass"  
}
```

2. Import the new configuration, as shown in this example for Web Engagement:

```
java -Dindex.tool.mode=true -Dworking.dir="./config" -Des.import.config=true -jar gdps-  
config-tool.jar import newtenant http://gwe-host-lb:9200
```

**Important:** When you import configuration data from a folder, the configuration tool imports all of the JSON files that are contained in that folder. Make sure that your folder only contains the files you need.

## Removing Pulse templates

If you have changed Pulse-related configuration data in the manual configuration mode or upgraded your widgets in the default configuration mode, you must remove your Pulse templates so Data Processing Server can create new ones.

You can remove your templates by using the **Add A Widget** menu of any Pulse dashboard. Here is a list of the Pulse templates that you need to remove for Web Engagement:

- GWE - Anonymous Vs Authenticated
- GWE - Conversions Funnel
- GWE - Mobile Vs Desktop
- GWE - New Vs Returning
- GWE - Rates
- GWE - Web Traffic Today

[Return to the Genesys Data Processing Server installation page](#)

## CSRF white list configuration

Beginning with version 8.5.220.20, the GAX Server includes a CSRF (Cross-site request forgery) filter which restricts access to GAX for some HTTP Requests.

In order to pass your Kibana requests through the Reporting Dashboard GAX Plugin, you must add the following configuration information to the **GAX Server/conf/trustedurl.properties** file:

```
urls=/wcc-kibana-proxy/**
```

---

# Installing the Web Engagement Reporting Server

**Starting with release 8.5.000.26, Web Engagement uses Genesys Data Processing Server for reporting. You must only use Web Engagement Reporting Server if you are running an earlier release.**

**Note:** Because many of the procedures involved in configuring the Reporting Server are similar to the ones you have used to configure Web Engagement Server, you may want to consult [Installing the Genesys Web Engagement Server](#) while you are configuring Reporting Server.

**Important:** Genesys recommends that you use a dedicated Cassandra data center for reporting purposes. This will minimize the risk of Cassandra-related faults in operational environments that are running under a heavy load.

## Important

Starting in 8.5.0, Embedded Cassandra mode is deprecated in Web Engagement; support for this mode will be discontinued in 9.0.

## Before You Begin

Because Web Engagement Reporting Server must process a lot of data, it needs to run on top of a high-speed—and highly scalable—cluster computing system. Genesys has chosen [Apache Spark](#) for this task.

Spark supports several types of clustering. Fortunately, Reporting Server works well with the simplest one, [Spark Standalone Mode](#). This mode provides high availability by using a dedicated master node. A typical cluster deployment will consist of one master node and several worker nodes and is usually started by Reporting Server in the background.

Genesys recommends that you configure Reporting Server in Genesys Administrator Extensions (GAX), defining a single Application Cluster object and the appropriate number of individual node objects. You can configure these objects and their options using the procedures provided on the rest of this page.

## Reporting Server Nodes

As we just mentioned, the Spark cluster consists of one master node and several worker nodes. Any Reporting Server node can be the master node in the cluster, as this role is defined by the value of the Spark [startMode](#) option in the node's configuration options. Here is more information about the two types of node:

- The **Master** node is represented by a Spark **start mode** of **both**, which indicates that both a Spark Master and a Spark Worker instance will be started on the node. Please note that the Spark **host** option should be set in agreement with **start mode** so that the hostname used in the Spark **host** setting belongs to the node that runs the Spark Master instance. To avoid problems with connectivity inside the Spark Cluster, this hostname should be the primary one in the network configuration for this host. In other words, Java's **InetAddress.getLocalHost** should return the same value for the Reporting Server Master node.
- The **Worker** node is represented by a **start mode** of **worker**. Only a Spark worker instance will be launched at this node.

There is one additional mode available for the Spark **start mode** option. A mode of **off** means that no Spark processes will be launched on this host and that the role of the node in the Reporting Server cluster is undefined. This mode is for use in situations where you want to have an externally managed Spark cluster, and limits you to one Reporting Server node, which serves as an entry point for the Spark cluster. You cannot deploy Reporting Server with multiple nodes if you have set the start mode to **off**. Also, if you use this mode, you must have an advanced understanding of how to work with and manage a Spark cluster.

## Recommended Versions

Genesys recommends that you use specific Web Engagement Reporting Server and Web Engagement Server versions together, as shown in the following table:

Web Engagement Reporting Server	Web Engagement Server
8.5.000.14	8.5.000.14
8.5.000.17	8.5.000.15

## Configuring Reporting Server

We have included information about Reporting Server-related **configuration options** at the end of this page.

## Deploying the Web Engagement Reporting Server

To deploy Reporting Server, follow these steps:

1. [Importing the Reporting Server Cluster Template](#)
2. [Creating the Cluster Application](#)
3. [Configuring the Cluster Application](#)
4. [Importing the Reporting Server Template](#)
5. [Creating a Node Application](#)

6. [Configuring a Node Application](#)
7. [Tuning analytical Cassandra nodes to skip indexing](#)
8. [Adding Nodes to a Cluster](#)
9. [Updating the Web Engagement Cluster Application](#)
10. [Reporting Server Data Storage](#)
11. [Installing the Reporting Server](#)
12. [Installing Dashboards and Widgets into Pulse](#)
13. [Deploying and Scheduling Job Packages](#)
14. [Web Engagement Reporting Data Flow](#)

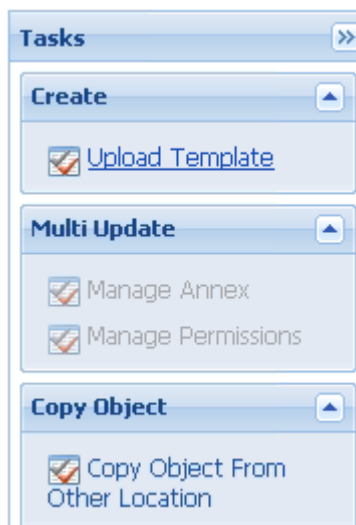
**Note:** For more information on how to work with templates and application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

## Importing the Reporting Server Cluster Template

**Note:** For more information on how to work with templates in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Application Templates**.
2. In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.

4. Browse to the **Reporting\_Server\_Cluster.apd** file. The **New Application Template** panel opens.
5. Click **Save & Close**.

## End

## Creating the Cluster Application

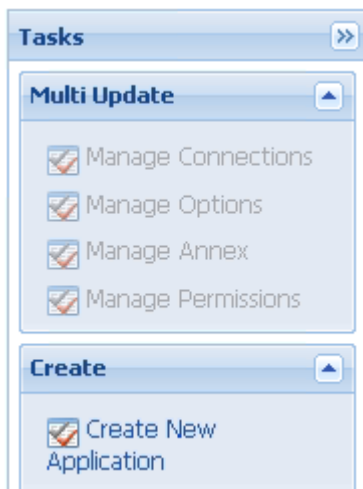
**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Importing the Reporting Server Cluster Template](#).

### Start

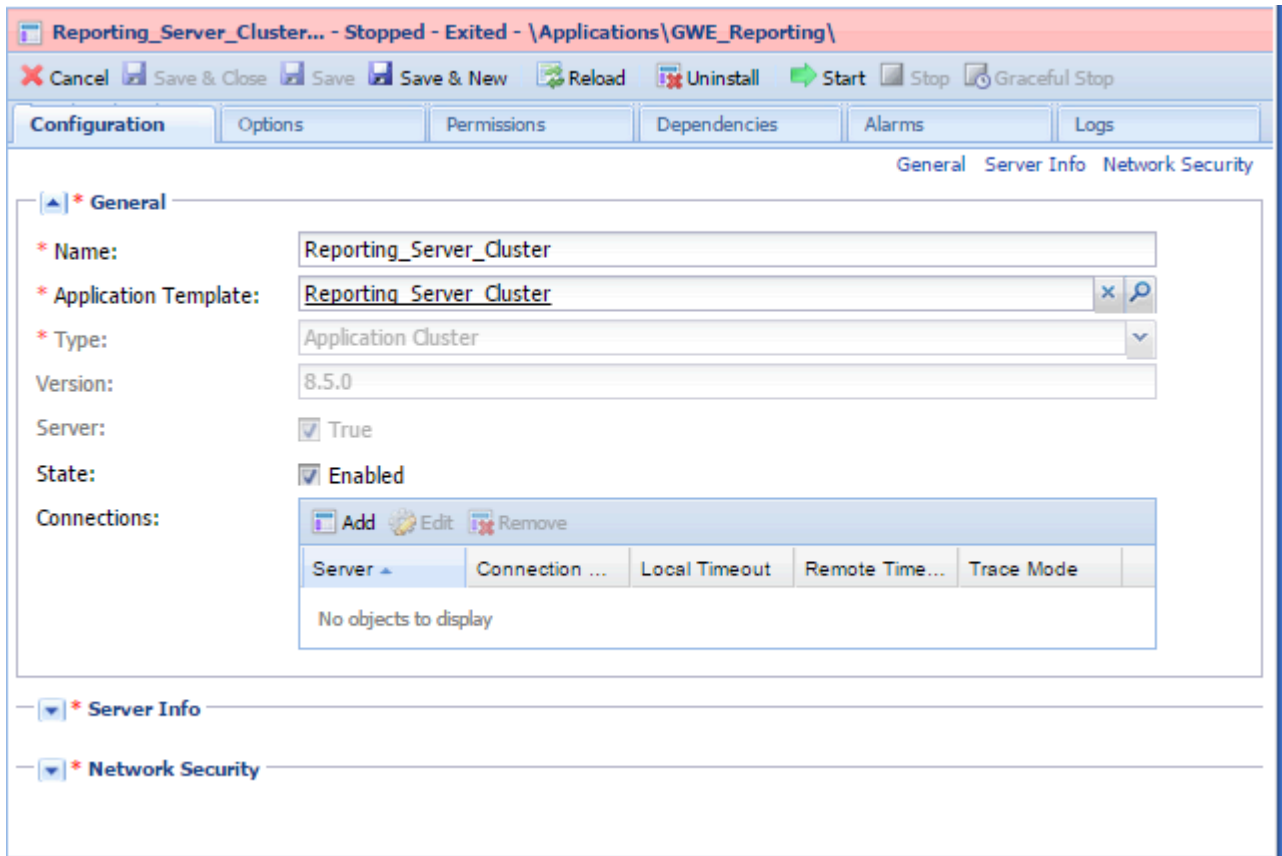
1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.



Create New Application link.

3. In the **Select Application Template** panel, click **Browse for Template** and select the Reporting Server Cluster template that you imported in [Importing the Reporting Server Cluster Template](#). Click **OK**.
4. The template is added to the **Select Application Template** panel. Click **Next**.
5. In the **Select Metadata file** panel, click **Browse** and select the **Reporting\_Server\_Cluster.xml** file. Click **Open**.
6. The metadata file is added to the **Select Metadata file** panel. Click **Next**.
7. In the **Specify Application parameters** tab:
  - Enter a name for your application. For instance, Reporting\_Server\_Cluster.

- Make sure **State** is enabled.
  - Select the **Host** on which the Reporting Server Cluster will reside.
  - Click **Create**.
8. The **Results** panel opens.
  9. Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The Reporting Server Cluster application form opens and you can start configuring the Reporting Server Cluster application.



Reporting Server Cluster app opened in Genesys Administrator.

**End**

## Configuring the Cluster Application

**Note:** For more information on how to work with application objects in Genesys Administrator, consult [Generic Configuration Procedures](#).

### Prerequisites

- You completed [Creating the Cluster Application](#).

## Start

1. If your Cluster application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the Reporting Server Cluster and click **Edit...**
2. Expand the **Server Info** pane.
3. If your **Host** is not defined, click the lookup icon to browse to the hostname of your application.
4. Ensure the **Working Directory** and **Command Line** fields contain "." (period).

Configuration	Options	Permissions	Dependencies	Alarms	Logs
* Working Directory:	.				
* Command Line:	.				
Command Line Arguments:					
* Startup Timeout:	90				
* Shutdown Timeout:	90				
Backup Server:	[Unknown Backup Server]				
* Redundancy Type:	Not Specified				
* Timeout:	10				
* Attempts:	1				
Auto Restart:	<input type="checkbox"/> True				
Log On As SYSTEM :	<input checked="" type="checkbox"/> True				
* Log On Account:	[Unknown Log On Account]				

Commands

5. Click **Save**.
6. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
  - Enter the **Port**. For instance, 10081.
  - Choose http for the **Connection Protocol**.
  - Click **OK**. The HTTP port with the default identifier appears in the list of **Listening ports**.
7. To configure Embedded Cassandra, select the **Options** tab.
  1. Make sure that the value of the **clusterName** option in the **[cassandraEmbedded]** section of the Reporting Server Cluster application is the same as the value of the **clusterName** option in the Web Engagement Cluster application.
  2. In the **[cassandraEmbedded]** section, you can take the default values for all of the options except **seedNodes**, which requires a comma-separated list of seed nodes, where each seed node is



---

represented by an IP address or fully qualified domain name (FQDN). For example:

- 192.168.0.1, 192.168.0.3
- host1.mydomain.com, host2.mydomain.com.

**Important:** The list of seed nodes should include at least one node from each operational and analytical data center.

3. In the `[cassandraKeyspace]` section, you may need to tune the `replicationStrategyParams` option, which by default is set to `'AnalyticalDC':1`. This value indicates a replication factor of **1** for nodes that belong to the `AnalyticalDC` data center. Note that the data center name is important for multi-data center configurations. When using the default endpoint snitch (`GossipingPropertyFileSnitch`), you can specify the data center name in **Reporting Server installation folder/resources/cassandra-rackdc.properties**. For more information, refer to <http://docs.datastax.com/en/cassandra/2.2/cassandra/architecture/archsnitchGossipPF.html>.
4. Make sure that the value of the `name` option in the `[cassandraKeyspace]` section of the Reporting Server Cluster application is the same as the value of the `name` option in the Web Engagement Cluster application.
8. To configure external Cassandra, follow the guidelines at [Working with External Cassandra](#), applying the steps on that page to the Reporting Server Cluster instead of the Web Engagement Server Cluster.
9. In the `[spark]` section, specify the value of the `host` option using the fully-qualified domain name (FQDN) or IP address for your Reporting Server.  
**Important:** Spark is very sensitive about host names and sometimes even minor network configuration problems may result in cluster connectivity problems. The best way to ensure correct behavior is to verify that the `spark.host` option uses the same name as the Java `InetAddress.getLocalHost().getHostName()` method would return for this host.
10. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.

**End**

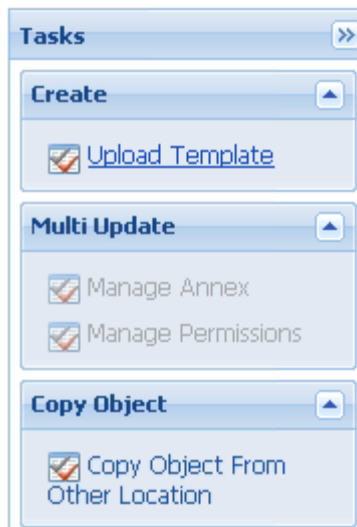
## Importing the Reporting Server Template

### Prerequisites

- You completed [Configuring the Cluster Application](#).

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Application Templates**.
2. In the **Tasks** panel, click **Upload Template**.



Upload Template link in the Tasks panel

3. In the **Click 'Add' and choose application template (APD) file to import** window, click **Add**.
4. Browse to the **Reporting\_Server.apd** file and select it. The **New Application Template** panel opens.
5. Click **Save & Close**.

## End

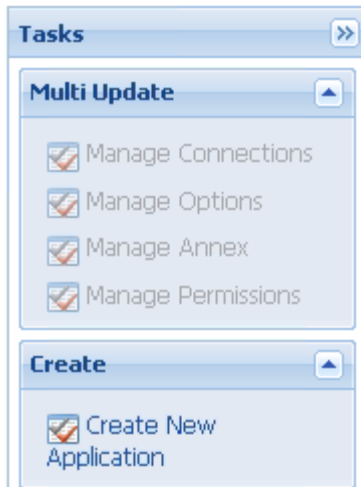
## Creating a Node Application

### Prerequisites

- You completed [Importing the Reporting Server Template](#).

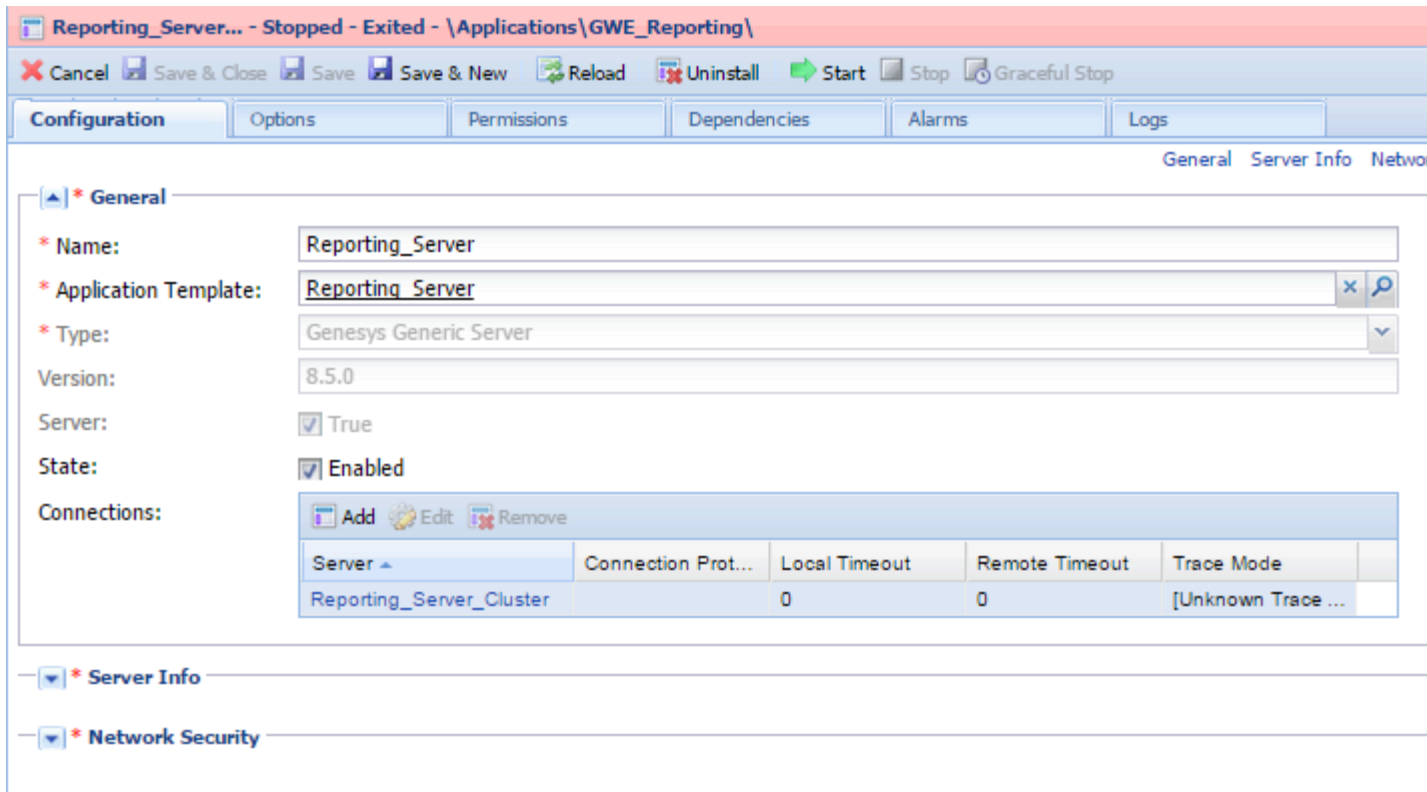
### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.



Create New Application link.

3. In the **Select Application Template** panel, click **Browse for Template** and select the Reporting Server template that you imported in **Importing the Reporting Server Template**. Click **OK**.
4. The template is added to the **Select Application Template** panel. Click **Next**.
5. In the **Select Metadata file** panel, click **Browse** and select the **Reporting\_Server.xml** file. Click **Open**.
6. The metadata file is added to the **Select Metadata file** panel. Click **Next**.
7. In **Specify Application parameters**:
  - Enter a name for your application. For instance, Reporting\_Server.
  - Make sure **State** is enabled.
  - Select the **Host** on which the node will reside.
  - Click **Create**.
8. The **Results** panel opens.
9. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.
10. Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The Reporting\_Server application form opens and you can start configuring the node application.



Node app opened in Genesys Administrator.

## End

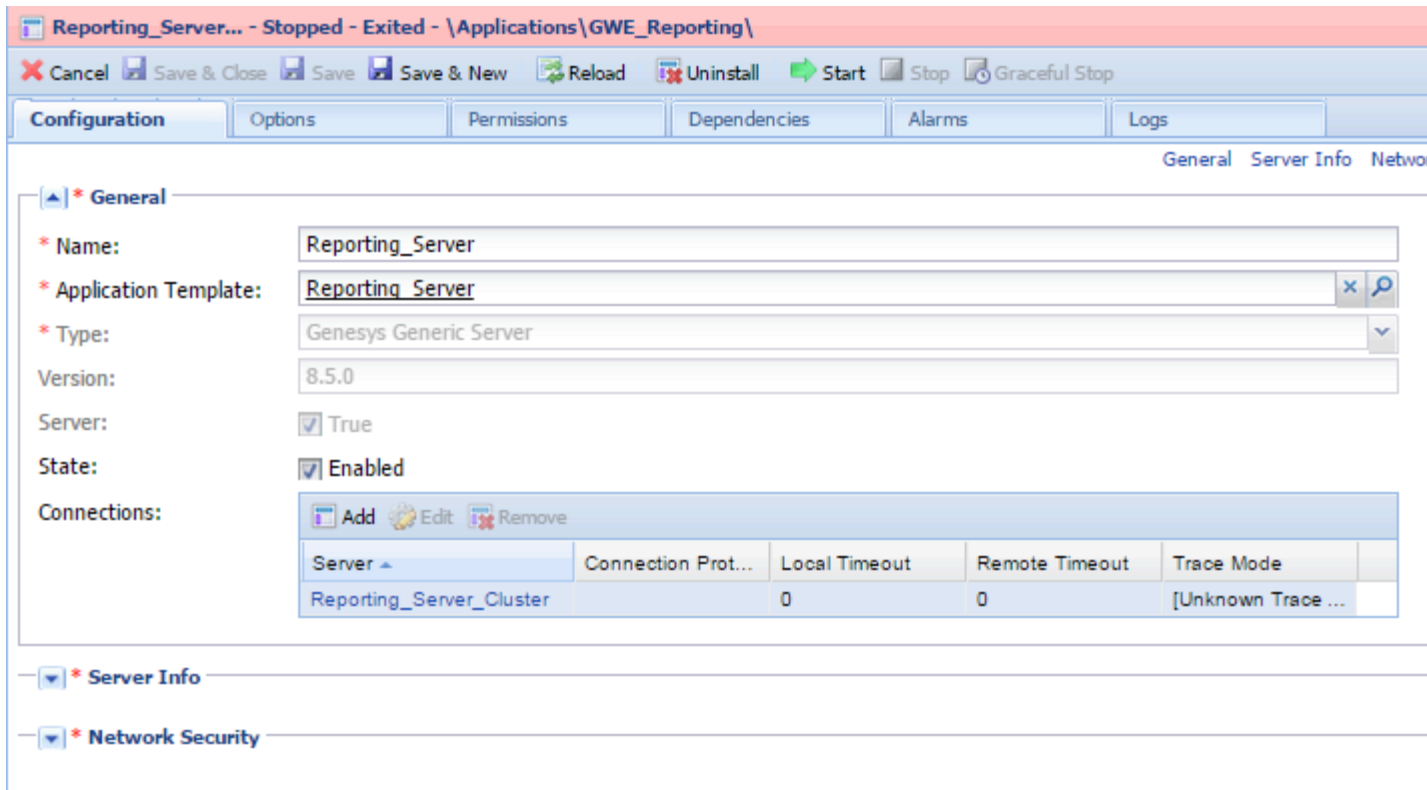
## Configuring a Node Application

### Prerequisites

- You completed [Creating a Node Application](#).

### Start

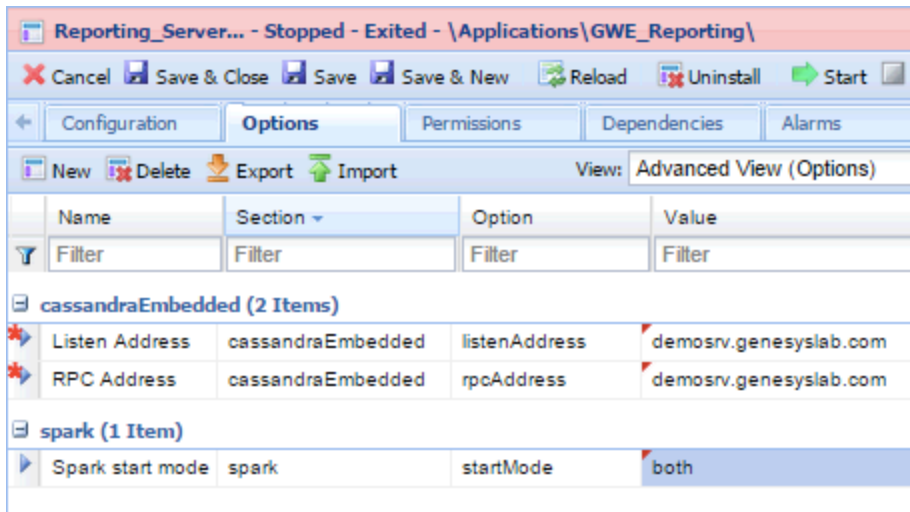
- If your node application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the node and click **Edit....**
- In the Connections section of the **Configuration** tab, click **Add**. The **Browse for applications** panel opens. Select the Reporting Server Cluster application you defined above, then click **OK**.



Node connection to Cluster

3. Expand the **Server Info** pane.
4. If your **Host** is not defined, click the lookup icon to browse to the hostname of your application.
5. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
  - Enter the **Port**. For instance, 10081.
  - Choose http for the **Connection Protocol**.
  - Click **OK**. The HTTP port with the default identifier appears in the list of **Listening ports**.
6. Click **Save**.
7. To configure Embedded Cassandra and Spark, select the **Options** tab.
  - In the **View** field, select **Advanced View (Options)**.
  - In the **[cassandraEmbedded]** section, set the values for **listenAddress** and **rpcAddress**, using a fully qualified domain name or the appropriate IP address.
  - In the **[spark]** section, set the value for **startMode**. If you are configuring a master node, set this value to **both**. For other nodes, set it to **worker**.
 

**Note:** You should only have one master node configured for your Reporting Server cluster. If you have a single-node cluster, then your single node must be configured as a master node.



Cassandra Listen and RPC Addresses

**End**

## Tuning analytical Cassandra nodes to skip indexing

For performance reasons, Web Engagement Server's *operational* Cassandra nodes use custom Cassandra indexes that rely on the services of Elasticsearch. However, the *analytical* Cassandra nodes used by Data Processing Server do not require those indexes, as all analytical reads of Cassandra tables are full scan reads. Because of this, Genesys recommends that these indexes be removed from your analytical nodes.

If you are using embedded Cassandra, Web Engagement will automatically avoid creating these indexes. But if you are using external Cassandra, you must carry out the following procedure to get rid of them.

**Note:** You must do these steps *before* you replicate the Cassandra data from your operational data centers.

**Start**

1. Copy the required libraries to the **Cassandra lib folder** for each Cassandra node in your analytical data centers.
2. Modify your Cassandra startup scripts to include the **genesys-es-dummy** system property
  - On Windows, append the following line to **bin/cassandra.in.bat**:

```
set JAVA_OPTS=%JAVA_OPTS% -Dgenesys-es-dummy=true
```

- On Linux, append the following line to **bin/cassandra.in.sh**:

```
export JVM_OPTS="$JVM_OPTS -Dgenesys-es-dummy=true"
```

**End**

## Adding Nodes to a Cluster

To create more nodes:

**Start**

1. Follow the instructions above for [Creating a Node Application](#), but use a different name for the new node.
2. [Configure the new node application](#), as shown above, but point to a different port.

**End**

## Updating the Web Engagement Cluster Application

Genesys recommends you to use a dedicated data center for reporting. In order to do this, you must do the following to the Web Engagement Cluster application:

- Modify the `seedNodes` option in the `[cassandraEmbedded]` section so that it is in sync with the `seedNodes` option in the Reporting Server Cluster application.
- Modify the `replicationStrategyParams` option in the `[cassandraKeyspace]` section so that it includes replication to the reporting data center. For example:

```
'OperationalDC':1,'AnalyticalDC':1
```

**Important:** You only need to do this for the Web Engagement Cluster application. The Reporting Server Cluster application does not need to be replicated to the operational data center.

## Reporting Server Data Storage

Reporting Server stores several types of information:

- Aggregated data results
- General configuration data
- Tenant-specific configuration data
- Default Pulse dashboards and widgets
- Meta-information

All of this information is stored in a database layer that is indexed by Elasticsearch. By default, you

---

can access the reporting database layer via HTTP, using the URL of the correctly configured Load Balancer:

- For embedded Cassandra, the Load Balancer should redirect requests to the **Web Engagement Server hosts** on port 9200 (or the port ID you have specified in the **http.port** option of the [\[elasticsearch\]](#) section of the Web Engagement Cluster application).
- For external Cassandra, the Load Balancer should redirect requests to the **Cassandra hosts** on port 9200 (or the port ID you have specified in the **http.port** option of the [\[elasticsearch\]](#) section of the Web Engagement Cluster application).

We will refer to this URL as *the Reporting Data URL*. When you send your browser or HTTP client requests to the Reporting Data URL, you should receive HTTP Response 200.

## Installing the Reporting Server

Install the Reporting Server on Windows or Linux.

**Note:** For more information on how to install apps that you have configured in Genesys Administrator, consult [Generic Installation Procedures](#).

### The Pulse Collector

Reporting Server uses data gathered by a Pulse Collector. This Pulse Collector must only be installed on one node in the Reporting Server cluster.

**Although the procedures in the next section tell you how to set up your initial Pulse Collector installation, if you decide later that you want to install the Pulse Collector on a different node, you must follow these extra steps:**

- Turn the collector off at the node it was originally installed on:
  - Stop the node
  - Set the **PULSE\_COLLECTOR\_ENABLED** variable in your **setenv.bat** or **setenv.sh** file to false
  - Remove the **pulse-collector.war** file from the **webapps** folder
  - Restart the node
- Turn the collector on at another node:
  - Stop the node
  - Set the **PULSE\_COLLECTOR\_ENABLED** variable in your **setenv.bat** or **setenv.sh** file to true
  - Set the value of **REPORTING\_ES\_URL**—also in your **setenv.bat** or **setenv.sh** file—to the value of your [Reporting Data URL](#)
  - Copy the **pulse-collector.war** file from the **etc** folder to the **webapps** folder
  - Restart the node



## Windows

### Prerequisites

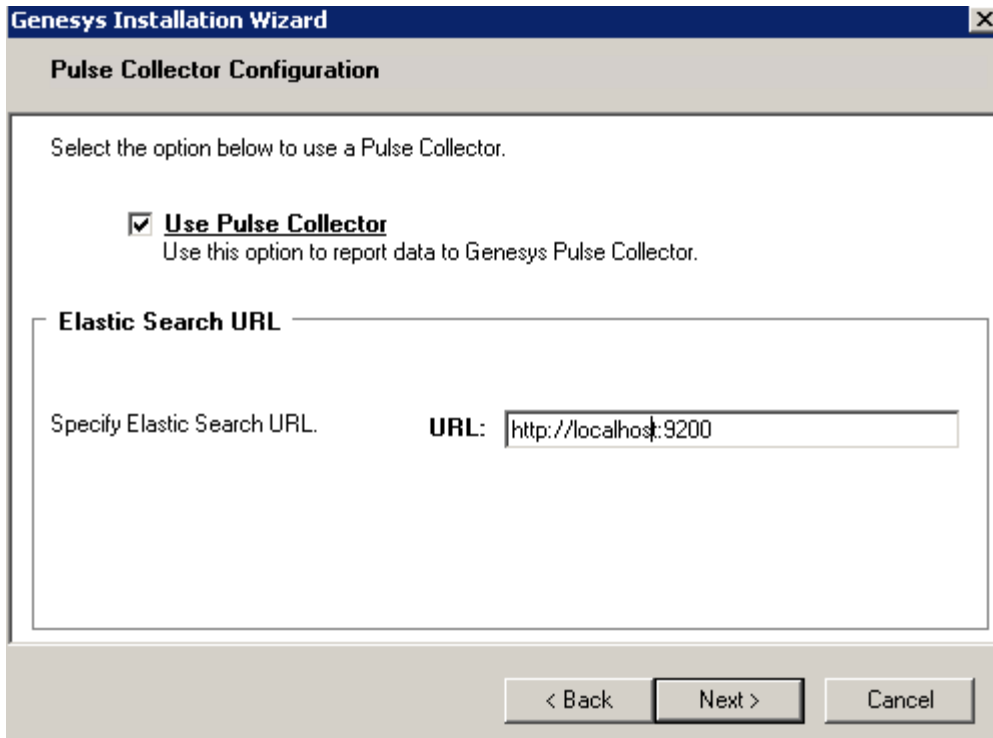
- [Configuring a Node Application](#)
- A supported JDK is installed. See [Java Requirements](#) for details.

### Start

1. In your installation package, locate and double-click the **setup.exe** file. The Install Shield opens the welcome screen.
2. Click **Next**. The **Connection Parameters to the Configuration Server** screen appears.
3. Under **Host**, specify the host name and port number where Configuration Server is running. (This is the main "listening" port entered in the **Server Info** tab for Configuration Server.)
4. Under **User**, enter the user name and password for logging on to Configuration Server.
5. Click **Next**. The **Select Application** screen appears.
6. Select the Reporting Server Application—that is, the Node app you created above—that you are installing. The **Application Properties** area shows the **Type**, **Host**, **Working Directory**, **Command Line executable**, and **Command Line Arguments** information previously entered in the **Server Info** and **Start Info** tabs of the selected Application object.
7. Click **Next**. The **Choose Destination Location** screen appears.
8. Under **Destination Folder**, keep the default value or browse for the desired installation location. Note that you probably do not want to use the Windows Program Files folder as your destination folder.
9. Click **Next**. The **Backup Configuration Server Parameters** screen appears.
10. If you have a backup Configuration Server, enter the **Host name** and **Port**.
11. In the **Pulse Collector Configuration** window, select **Use Pulse Collector**:

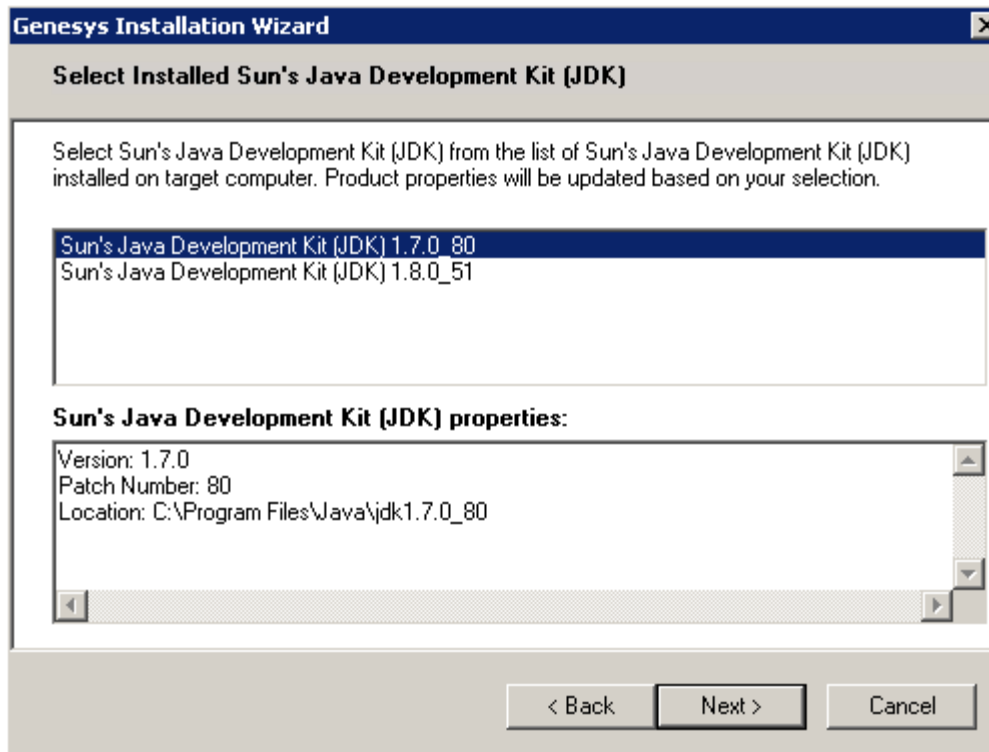


12. In the resulting **Elasticsearch URL** field, specify the host and port of the Reporting Server cluster node on which you want to install the Pulse Collector. **Note:** You should only install the Pulse Collector on one node at a time, as mentioned [above](#).



Pulse Collector Host and Port

13. Click **Next**. Select the appropriate JDK:



Select JDK

14. Click **Next**. The **Ready to Install** screen appears.
15. Click **Install**. The Genesys Installation Wizard indicates it is performing the requested operation for Reporting Server. When through, the **Installation Complete** screen appears.
16. Click **Finish** to complete your installation of the Reporting Server.
17. Inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.
18. **Apply the parameters to your Windows service.**

## End

**Note:** Genesys recommends that you frequently clear the Spark temporary directory—for example, once a week or before you start Reporting Server. You can find it in the system temporary directory with a name template of spark-\*. The default location for this directory is **system\_disk:\Users\user\_name\AppData\Local\Temp** directory. You can also use the system disk clean-up procedure.

## Linux

### Prerequisites

- [Configuring a Node Application](#)
- A supported JDK is installed. See [Java Requirements](#) for details.

### Start

1. Open a terminal in the Genesys Web Engagement CD/DVD or the Genesys Web Engagement IP, and run the **install.sh** file. The Genesys Installation starts.
2. Enter the hostname of the host on which you are going to install.
3. Enter the connection information to log in to Configuration Server:
  - The hostname. For instance, `demosrv.genesyslab.com`.
  - The listening port. For instance, `2020`.
  - The user name. For instance, `demo`.
  - The password.
4. If you have a backup Configuration Server, enter the **Host name** and **Port**.  
If the connection settings are successful, a list of keys and Web Engagement applications is displayed.
5. Enter the key for the Reporting Server application—that is, the Node app you created above in Configuration Server.
6. Use the key for Genesys Pulse to indicate whether or not to enable the Pulse Collector
7. If you have enabled the Pulse Collector, enter the Reporting Data Elasticsearch URL.
8. Enter the location where Reporting Server is to be installed on your web server.  
**Note:** This location must match the previous settings that you entered in Configuration Server.
9. If the installation is successful, the console displays the following message:  
`Installation of Web Engagement Reporting Server, version 8.5.x has completed successfully.`
10. Inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.

### End

**Note:** Genesys recommends that you frequently clear the Spark temporary directory—for example, once a week or before you start Reporting Server. You can find it in the system temporary directory with a name template of `spark-*`. The default location for this directory is **/tmp**.

## Installing Dashboards and Widgets into Pulse

Web Engagement comes with a configuration tool that is located in **Web Engagement installation directory/tools/pulse**. You can use this tool to install Web Engagement dashboards and widgets into Pulse in two different ways:

- **Default Configuration**—Only requires a few parameters that drive an automated configuration process
- **Manual Configuration**—Requires more extensive inputs, but provides more advanced configuration options

### Prerequisites

- The **enabled** option in the **[kibana]** section of the Genesys Web Engagement cluster application is set to **true**.
- Web Engagement is running.
- The **Reporting Data URL** is accessible.

### Default Configuration

#### Start

1. Run the default configuration scenario with **gdps-config-tool.jar**:
  - a. Go to **Web Engagement installation directory/tools/pulse**.
  - b. Execute

```
java -jar gdps-config-tool.jar <Reporting Data URL> <gax_host> <gax_username>
<gax_password> <Web Engagement Data URL>
```

Where <Web Engagement Data URL> is the address of the Load Balancer, which redirects requests to the Web Engagement Servers on the port specified by the **port** option in the **[kibana]** section of the Web Engagement Cluster application. For example:

```
java -jar gdps-config-tool.jar http://gwe-cluster-lb:9200 http://gax-host:8040 default
password http://gwe-cluster-lb:5601
```

**Note:** <Web Engagement Data URL> is optional and can be skipped. If this URL is skipped, the Expanded Kibana View will be disabled. For example:

```
java -jar gdps-config-tool.jar http://gwe-cluster-lb:9200 http://gax-host:8040 default
password overwriteSystem
```

**Note:** The default scenario uses an index of **gpe**. To use a custom index name for your GWE tenant run **gdps-config-tool.jar** with **-Dindex.name=Web\_Engagement\_keyspace\_name**, where *Web\_Engagement\_keyspace\_name* is the value of the **name** option from the **[cassandraKeyspace]** section of the Web Engagement Cluster application. For example:

```
java -Dindex.name=gwekeyspace -jar gdps-config-tool.jar http://gwe-cluster-lb:9200
http://gax-host:8040 default password http://gwe-cluster-lb:5601
```

**Note:** By default, **gdps-config-tool.jar** creates a new system configuration when it is first run, but does not overwrite the system configuration if you run it again. To overwrite an existing system configuration, use the **overwriteSystem** flag. For example:

```
java -jar gdps-config-tool.jar http://gwe-cluster-lb:9200 http://gax-host:8040 default
password http://gwe-cluster-lb:5601 overwriteSystem
```

2. If the configuration process was successful, the console will display **DONE**.  
**Note:** If an error has occurred, you can try to run the command again, as there are occasions when the configuration tool can't access an http resource, or when an http response times out.
3. Notify Reporting Server about the configuration changes using one of these methods:
  - a. Send a GET request:

```
http://<Reporting Server Host>:<Reporting Server Port>/pulse-collector/gdps/
configuration/init
```

For example,

```
http://example.com:9999/pulse-collector/gdps/configuration/init
```

If the response is Collector has been re-initialized with new configuration, the new configuration was applied successfully.

- b. Reboot Reporting Server.

**Note:** You do not need to reboot GAX Server, Pulse, or Web Engagement Server after your Web Engagement-specific dashboards and widgets have been installed. All of the new widget templates, dashboards, and related elements will be added automatically.

## End

## Manual Configuration

For manual configuration, run **gdps-config-tool.jar** with **-Dindex.tool.mode=true**.

For more information on how to do this, run `java -Dindex.tool.mode=true -jar gdps-config-tool.jar`.

### Example 1

Export dashboards and configuration to the **./exported** directory:

```
java -Dindex.tool.mode=true -Dworking.dir="./exported" -Des.import.config=true -jar gdps-
config-tool.jar export gpe.kibana http://gwe-host-lb:9200
```

### Example 2

Import the collector configuration for a new tenant, in the **newtenant** keyspace:

## Start

1. Create a JSON file with tenant-related configuration information in the **./config** folder:

```
./config/gdps.collector-tenantConfiguration.json
{
  "tenantAliasName": "newtenant",
```

```
"gaxUrl": "http://gaxurl:8040",  
"gaxUser": "gaxUser",  
"gaxPass": "gaxPass"  
}
```

2. Import the new configuration:

```
java -Dindex.tool.mode=true -Dworking.dir="./config" -Des.import.config=true -jar gdps-  
config-tool.jar import newtenant http://gwe-host-lb:9200
```

**Important:** When you import configuration data from a folder, the configuration tool imports all of the JSON files that are contained in that folder. Make sure that your folder only contains the files you need.

**Note:** If you have changed Pulse-related configuration data in manual mode, you must remove your Pulse Web Engagement templates so that they can be recreated by Reporting Server. You can remove the Web Engagement Pulse templates by using the **Add A Widget** menu of any Pulse dashboard. Here is a list of the templates that you need to remove:

- GWE - Anonymous Vs Authenticated
- GWE - Conversions Funnel
- GWE - Mobile Vs Desktop
- GWE - New Vs Returning
- GWE - Rates
- GWE - Web Traffic Today

## End

## Deploying and Scheduling Job Packages

After you have installed **and started** the Reporting Server, you need to deploy and schedule jobs for it to process.

To do this, execute the following two scripts that are provided in **Reporting Server Installation directory/deploy/gwe**. You need to install **CURL** in order to use these scripts. Once you have installed it, make sure it's available from **Reporting Server Installation directory/deploy/gwe**, and run the following scripts:

1. **deploy-gwe.bat** or **deploy-gwe.sh** deploys your jobs and requires the URL (host and port) for your Reporting Server.

If the script executes successfully, you will receive the following response: {"status" : "0k"}

2. **schedule-gwe.bat** or **schedule-gwe.sh** schedules your jobs and requires:

- The URL (host and port) for your Reporting Server
- The **Reporting Data URL**
- The name of the Cassandra keyspace that is used by Web Engagement

Here is a sample command:

```
schedule-gwe.bat reporting-host:10081 gwe-cluster-lb:9200 gpe
```



In this sample, we can see that:

- Reporting Server is running on reporting-host at port 10081.
- The Load Balancer running on gwe-cluster-lb at port 9200 represents a Reporting Data URL.
- The GWE keyspace name is gpe.

If the script executes successfully, you will receive the following response: {"<schedule\_guid>" : "Job scheduled"}

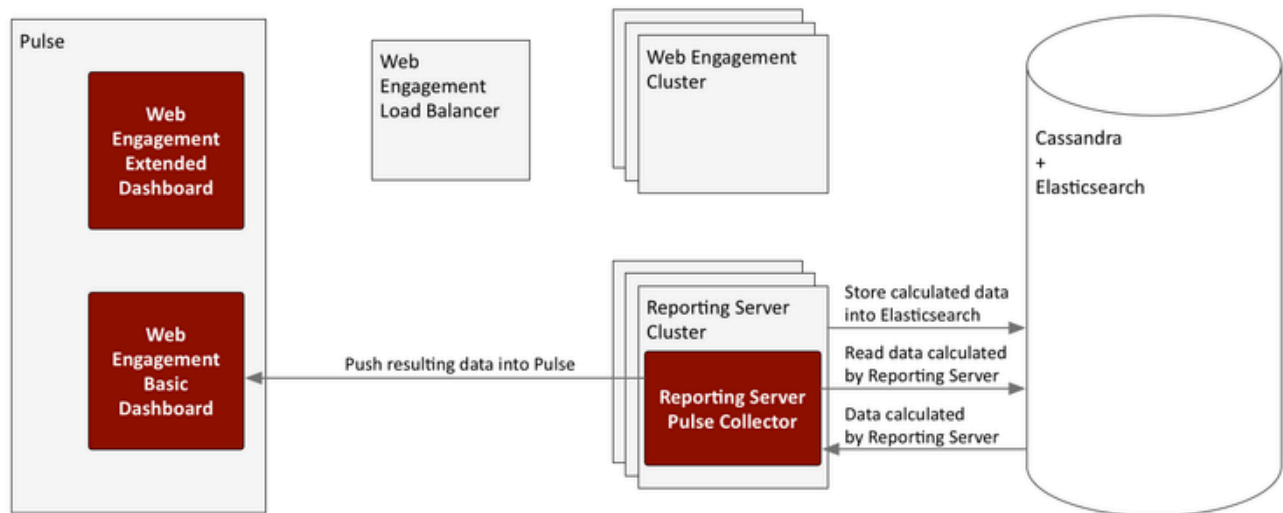
## Web Engagement Reporting Data Flow

Web Engagement supports two types of Pulse reporting:

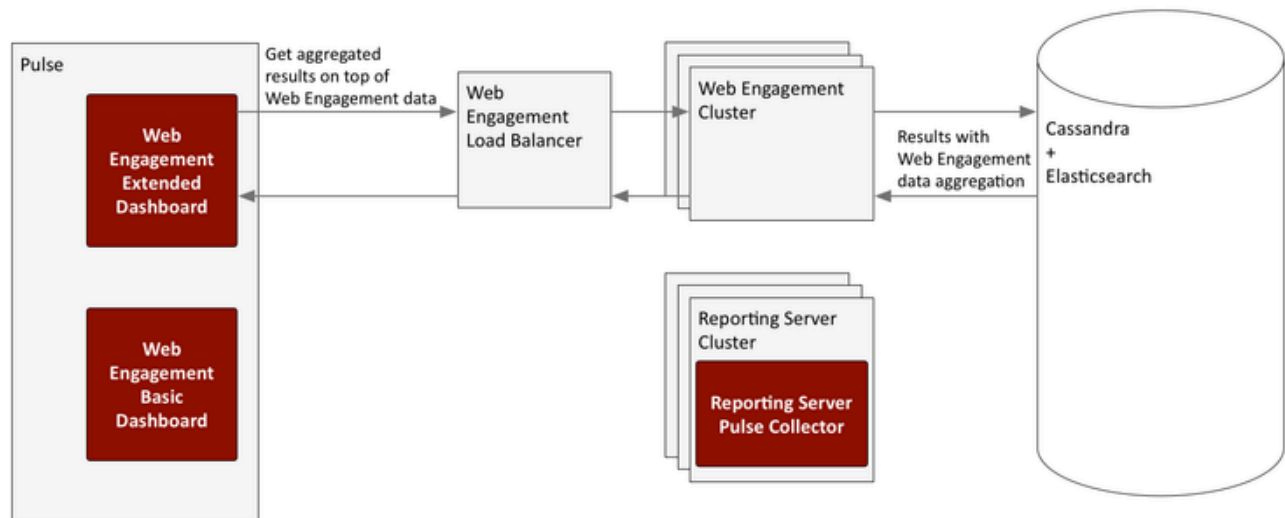
- The **basic views**, also known as the basic dashboards, are native Pulse widgets that are driven by a Pulse collector which is included with the Reporting Server. These widgets can be used in other dashboard views created in Pulse.
- The **extended views (dashboards)** are bespoke metric views of Web Engagement data sources hosted directly within Pulse. These widgets cannot be used with other Pulse dashboards.

The following diagrams show the data flow for each type of reporting.

### Basic Dashboard



## Extended Dashboard



## Configuration Options

These configuration option sections can be useful in setting up Reporting Server and your Spark cluster.

### log

The `[log]` configuration options are applied to the Reporting Server environment in a way that is similar to how they are used with Web Engagement Server.

### cassandraEmbedded

If you have set up your Reporting Server with an embedded Cassandra node, the `[cassandraEmbedded]` section provides its configuration options. Their meanings when applied to the Reporting Server environment are similar to the way they are used with Web Engagement Server.

### cassandraKeyspace

Reporting Server stores some of its data (packages and schedule) in a dedicated Cassandra keyspace. The `[cassandraKeyspace]` section provides its configuration options. All of the options mean pretty much the same as they do when used with Web Engagement Server, although some of their values—such as for the **name**—will be different.

## spark

Reporting Server launches a dedicated Spark cluster and all of the Reporting Server nodes need to share the coordinates of the Spark Master node. In addition to this, each individual node has options that can be used to configure the mode with which Spark starts on its box.

### host

**Description:** The name of the Spark Master host. The value should be the same as what Java's `InetAddress.getLocalHost()` would return for the specified host.

**Default Value:** None

**Valid Values:** *hostname of the Spark Master node*

**Mandatory:** No

**Changes Take Effect:** After start/restart

### port

**Description:** The port number of the Spark Master host.

**Default Value:** 7077

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

### startMode

**Description:** The mode that will be used when starting Spark. If set to off, Spark will not be started by Reporting Server, and will instead have its state managed externally. If set to worker, only a worker node will be started. If set to both, both a worker node and a master node are started. **Note:** Genesys recommends that you set this option for each node to clearly specify the role. However, you can set the Cluster object to worker mode and override that value for the master node by setting that node to both.

**Default Value:** worker

**Valid Values:** off, worker, or both

**Mandatory:** No

**Changes Take Effect:** After start/restart

### masterWebPort

**Description:** The number of the TCP port that the Spark Master web UI will listen on. Note that this option is provided for cases when the default port has already been used by another service.

**Default Value:** 8080

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

### workerWebPort

**Description:** The number of the TCP port that the Spark Worker web UI will listen on. Note that this

---

option is provided for cases when the default port has already been used by another service.

**Default Value:** 8081

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### executorMemory

**Description:** Use this option to manage the amount of memory used by Spark for executing tasks on each node. Genesys recommends at least two gigabytes per node, but more memory can improve performance if hardware allows. For information about the format, consult the Spark documentation.

**Default Value:** None

**Valid Values:** Valid memory limit

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### sparkHeartbeatTimeout

**Description:** The timeout value in seconds between two heartbeat calls to the Spark metrics API.

**Default Value:** 60

**Valid Values:** Positive integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### sparkStartTimeout

**Description:** The timeout value in seconds between a Spark start or restart and the first time its API is checked. On slower machines, it makes sense to increase this value so that Spark has enough time to start successfully (without initiating a restart cycle).

**Default Value:** 20

**Valid Values:** Positive integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### uri

**Description: Advanced.** For situations when Spark is running externally, you must set the URI instead of the host and port. The URI must include the protocol, in addition to the host and port.

**Default Value:** None

**Valid Values:** Valid Spark URI

**Mandatory:** No

**Changes Take Effect:** After start/restart

#### spark.context

**Advanced.** This entire section is copied into **SparkContext**, so it can be used to tune the Spark options. You must have an in-depth understanding of Spark configuration if you are going to use this section.

# Security

Genesys Web Engagement supports HTTPS/SSL/TLS to protect data over the web.

- All connections can be secured, including connections from the browser to the Web Engagement server.
- Applications defined in Configuration Server can have both HTTP and HTTPS connections.

Transport Layer Security (TLS) is supported above Java containers, Jetty and Apache Tomcat. The user data submitted from the browser tier is always sent through secure connections.

## Important

Genesys performs security testing with [OWASP Zed Attack Proxy \(ZAPProxy\)](#) to protect the Genesys Web Engagement solution against known vulnerabilities. For details, see [Security Testing with ZAPProxy](#).

Genesys Web Engagement includes additional security configurations that can be used with your GWE installation:

- [Secure Sockets Layer \(SSL\)](#) — Load SSL certificates and configure Jetty.
- [Transport Layer Security \(TLS\)](#) — Configure TLS for Genesys and Web Engagement servers.
- [Authentication](#) — Enable authentication for the Web Engagement Server, Interaction Workspace, and the Engagement Strategy.
- [Cassandra Security](#) — Establish a secure channel between your client and Cassandra coordinator nodes.

## Next Steps

After you configure security for Genesys Web Engagement, you can [configure features](#) to enable additional functionality.

# Security Tips for Third-Party Components

Web Engagement and Data Processing Server can't do their jobs without the help of these third-party Java-based applications:

- Cassandra
- Elasticsearch
- Spark

Because Genesys doesn't create these products, we can't control their dependencies. This means that they are subject to vulnerabilities that would not otherwise affect Web Engagement. Please pay attention to the following recommendations as you are securing your Web Engagement environment.

## Spark

Genesys Data Processing Server comes with Spark, which runs separately outside of the Data Processing Server JVM process. Be sure to limit your Spark-related connections, by verifying for example, that:

- Ports are opened only for specific IP addresses
- JMX is controlled in the appropriate way

## Cassandra

Genesys recommends that you use an external Cassandra cluster. Be sure that:

- This cluster is located in the secure zone
- Access to Cassandra hosts (default ports 9042, 9160, 7000, 7001) is granted only for a limited set of client connections

By default, Cassandra will not open a JMX port, but if you do need to open one, please consider that an open JMX connection is a security exposure, so pay attention to securing the JMX port by taking care of things like these:

- Limit access to specific IP addresses
- Apply authentication parameters

## Elasticsearch

In Web Engagement 8.5, the Elasticsearch nodes are running in the same JVM as the Cassandra nodes. This means that most Cassandra-related security measures will automatically be applied to Elasticsearch.

However, Elasticsearch also opens additional ports (by default 9200 and 9300), so these ports must also be secured.

---

# Secure Connections to HTTP Clients

The Jetty web server supplied with the Genesys Web Engagement solution includes a pre-configured, self-signed certificate. This allows you to use HTTPS out of the box in a [lab deployment](#).

For a [production deployment](#), you should use a certificate issued by a third-party Certificate Authority. The procedures on this page provide examples of ways to load SSL certificates and configure Jetty. These examples may vary depending on your environment.

## Loading an SSL Certificate and Private Key into a JSSE Keystore

### Important

In a development environment, you can use self-signed certificates, but in a production environment you should use a certificate issued by a third-party Certificate Authority, such as VeriSign.

### Prerequisites

- An SSL certificate, either generated by you or issued by a third-party Certificate Authority. For more information on generating a certificate, see [http://wiki.eclipse.org/Jetty/Howto/Configure\\_SSL](http://wiki.eclipse.org/Jetty/Howto/Configure_SSL).

### Start

1. Depending on your certificate format, do **one** of the following:

- If your certificate is in PEM form, you can load it to a JSSE keystore with the keytool using the following command:  

```
keytool -keystore keystore -importcert -alias alias -file certificate_file -trustcacerts
```

**Where:**

*keystore* is the name of your JSSE keystore.

*alias* is the unique alias for your certificate in the JSSE keystore.

*certificate\_file* is the name of your certificate file. For example, *jetty.crt*.

- If your certificate and key are in separate files, you must combine them into a PKCS12 file before loading it to a keystore.

1. Use the following command in openssl to combine the files:

```
openssl pkcs12 -inkey private_key -in certificate -export -out pkcs12_file
```

**Where:**

*private\_key* is the name of your private key file. For example, *jetty.key*.



*certificate* is the name of your certificate file. For example, `jetty.crt`.

*pkcs12\_file* is the name of the PKCS12 file that will be created. For example, `jetty.pkcs12`.

2. Load the PKCS12 file into a JSSE keystore using `keytool` with the following command:  

```
keytool -importkeystore -srckeystore pkcs12_file -srcstoretype store_type
-destkeystore keystore
```

**Where:**

*pkcs12\_file* is the name of your PKCS12 file. For example, `jetty.pkcs12`.

*store\_type* is the file type you are importing into the keystore. In this case, the type is PKCS12.

*keystore* is the name of your JSSE keystore.

## Important

You will need to set two passwords during this process: keystore and truststore. Make note of these passwords because you will need to add them to your Jetty SSL configuration file.

## End

## Next Steps

- [Configuring Jetty](#)

## Configuring Jetty

This section provides basic information about how to configure SSL in Jetty. For more information, see <http://www.eclipse.org/jetty/documentation/current/configuring-ssl.html>.

## Prerequisites

- You completed [Loading an SSL Certificate and Private Key into a JSSE Keystore](#)

## Start

1. Open the SSL configuration file, ***Web Engagement Root Directory/server/etc/jetty-ssl.xml***, in a text editor.
2. Find the `<New id="sslContextFactory" class="org.eclipse.jetty.http.ssl.SslContextFactory">` element and update all paths and passwords.  
**Note:** You can run Jetty's password utility to obfuscate your passwords. See <http://www.eclipse.org/jetty/documentation/current/configuring-security-secure-passwords.html>.
3. Save your changes.
4. Open the Jetty SSL module configuration file, ***Web Engagement Root Directory/server/modules/***

**ssl.mod**, in a text editor.

5. Comment out all properties settings after the line that says **etc/jetty-ssl.xml** *except for lines containing:*
  - **[files]**
  - **[ini-template]**
6. Save your changes.

**End**

## Choosing a Directory for the Keystore

The keystore file in the example above is given relative to the Jetty home directory. For production, you should keep your keystore in a private directory with restricted access. Even though the keystore has a password, the password may be configured into the runtime environment and is vulnerable to theft.

You can now start Jetty the normal way (make sure that **jcrt.jar**, **jnet.jar** and **jsse.jar** are on your classpath) and SSL can be used with a URL, such as `https://your_IP:8743/`

## Next Steps

- Return to the [Genesys Web Engagement Security](#) page.

---

# Transport Layer Security (TLS) with Genesys Servers

Genesys Web Engagement supports the Transport Layer Security (TLS) protocol to secure data exchanged with other Genesys components. For details about TLS, see the [Genesys Security Deployment Guide](#). You can configure TLS for Web Engagement by completing the procedures on this page.

## Configuring TLS for Genesys Servers

To configure the TLS parameters for Genesys servers such as Configuration Server, Message Server, Interaction Server, and so on, see [Configuring TLS Parameters in Configuration Manager](#).

## Configuring TLS for Web Engagement Servers

To enable TLS support for the Genesys Web Engagement Server, you must do the following:

1. Have properly installed trusted certificates for the Genesys servers.
2. Configure TLS options for the Web Engagement Server application.
3. Configure the appropriate connections between the Web Engagement Server application and the necessary Genesys servers through secure ports.

## Configuring TLS Options

The Genesys Web Engagement Server includes the following TLS-related configuration options in its [\[security\]](#) section.

Option	Default Value	Mandatory	Changes Take Effect	Description
<a href="#">provider</a>	none	no	after restart	Type of trusted storage  Valid values: MSCAPI, PEM or JKS. If empty, TLS support is disabled.
<a href="#">trusted-ca</a>	none	no	after restart	Specifies the name of the trusted store file which holds the public certificate to verify the server.

Option	Default Value	Mandatory	Changes Take Effect	Description
				Applicable for PEM and JKS trusted storage types only. Valid values: valid file name (including path)
<code>truststore-password</code>	none	no	after restart	Password for the JKS trusted storage.  Valid values: any string

See [Configuring Trusted Stores](#) below for details about configuration for a specific type of store (PEM, JKS, MSCAPI).

## Configuring Trusted Stores

### PEM Trusted Store

PEM stands for "Privacy Enhanced Mail", a 1993 IETF proposal for securing email using public-key cryptography. That proposal defined the PEM file format for certificates as one containing a Base64-encoded X.509 certificate in specific binary representation with additional metadata headers.

PEM certificate trusted store works with CA certificate from an X.509 PEM file. It is a recommended trusted store to work on Linux systems.

Complete the steps below to work with the PEM certificate trusted store:

#### Start

1. Configure TLS for Genesys servers to use certificates signed by CA certificate **certificateCA.crt**.
2. Place the trusted CA certificate in PEM format on the Genesys Web Engagement Server application host. To convert a certificate of another format to .pem format you can use the [OpenSSL tool](#). For example:
  - Convert a DER file (.crt .cer .der) to PEM:  

```
openssl x509 -inform der -in certificateCA.crt -out certificateCA.pem
```
  - Convert a PKCS#12 file (.pfx .p12) containing a private key and certificates to PEM:  

```
openssl pkcs12 -in certificateCA.pfx -out certificateCA.pem -nodes
```

You can add **-nocerts** to only output the private key or add **-nokeys** to only output the certificates.
3. In Genesys Administrator, navigate to **Provisioning > Environment > Applications** and open your Web Engagement Server application.
4. Click the **Options** tab and navigate to the [\[security\]](#) section.
5. Set the **provider** option to PEM.
6. Set the **trusted-ca** option to the path and file name for your trusted CA in PEM format on the Genesys Web Engagement Server application host.

7. Click **Save & Close**.

## End

### JKS Trusted Store

A Java KeyStore (JKS) is a repository of security certificates used, for instance, in SSL/TLS encryption. The Java Development Kit provides a tool named **keytool** to manipulate the keystore.

Complete the steps below to work with the JKS certificate trusted store:

## Start

1. Configure TLS for Genesys servers to use certificates signed by CA certificate **certificateCA.crt**.
2. Import the CA certificate to an existing Java keystore using keytool:
  - Run the keytool command with option **-alias** set to **root**:  

```
keytool -import -trustcacerts -alias root -file certificateCa.crt -keystore /path/to/keysore/keystore.jks
```
  - Enter the keystore password in command line prompt - for example:  
Enter keystore password: somepassword
3. In Genesys Administrator, navigate to **Provisioning > Environment > Applications** and open your Web Engagement Server application.
4. Click the **Options** tab and navigate to the **[security]** section.
5. Set the **provider** option to JKS.
6. Set the **trusted-ca** option to the path and file name for your JKS trusted storage type on the Genesys Web Engagement Server application host.
7. Set the **trusted-pwd** option to the password defined for your keystore in Step 2.
8. Click **Save & Close**.

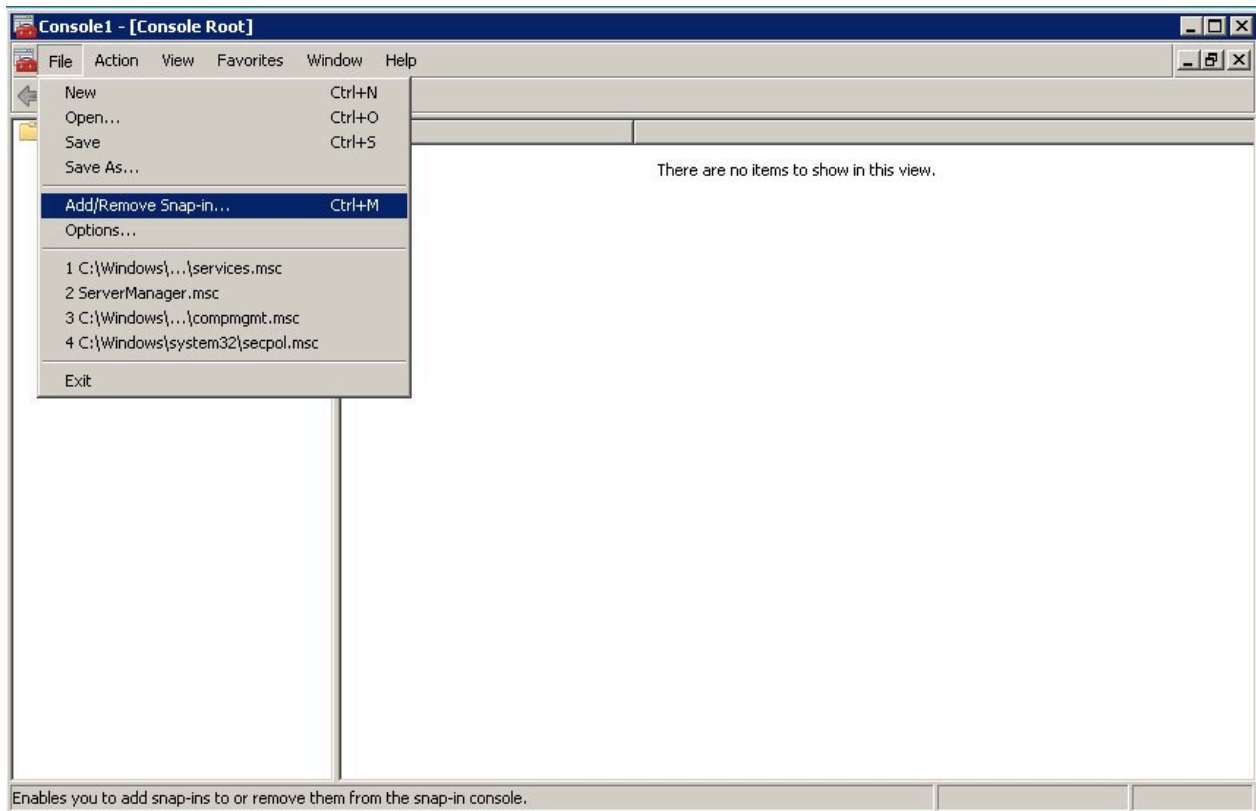
## End

### MSCAPI Trusted Store

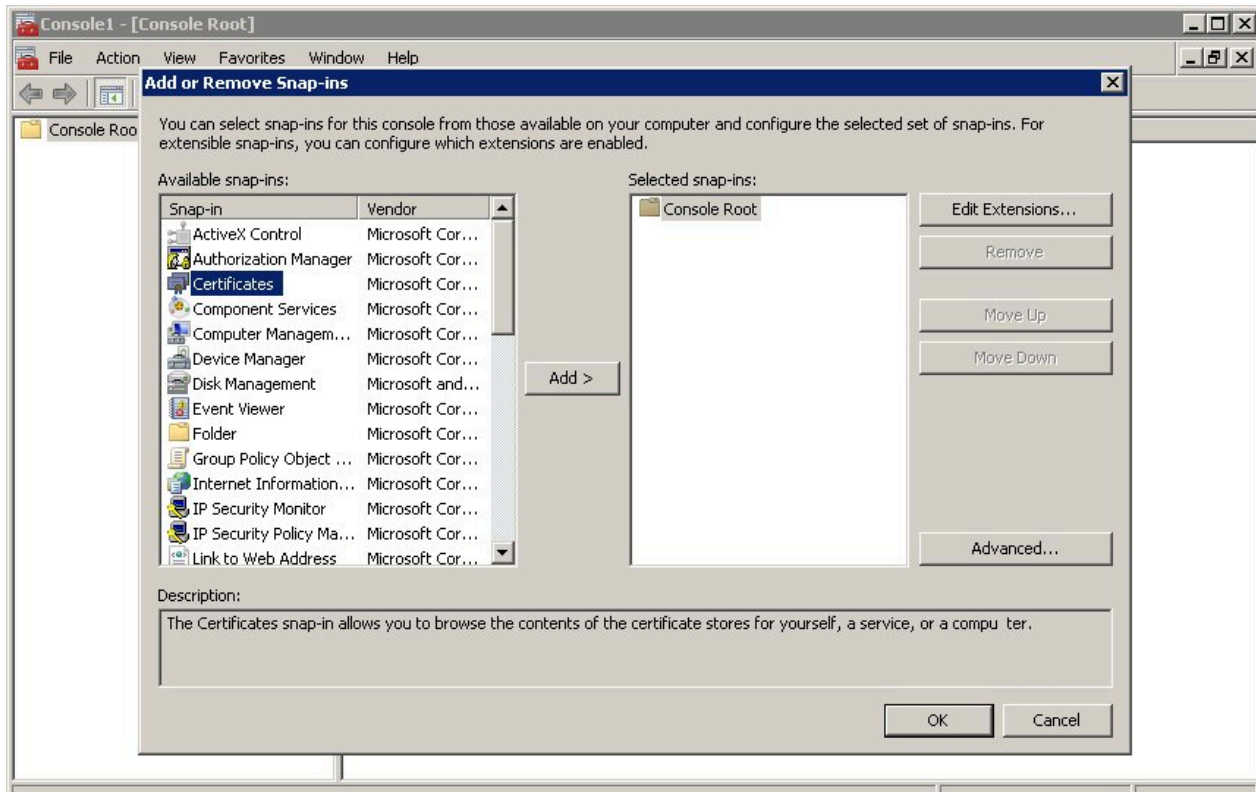
Complete the steps below to work with the MSCAPI certificate trusted store:

## Start

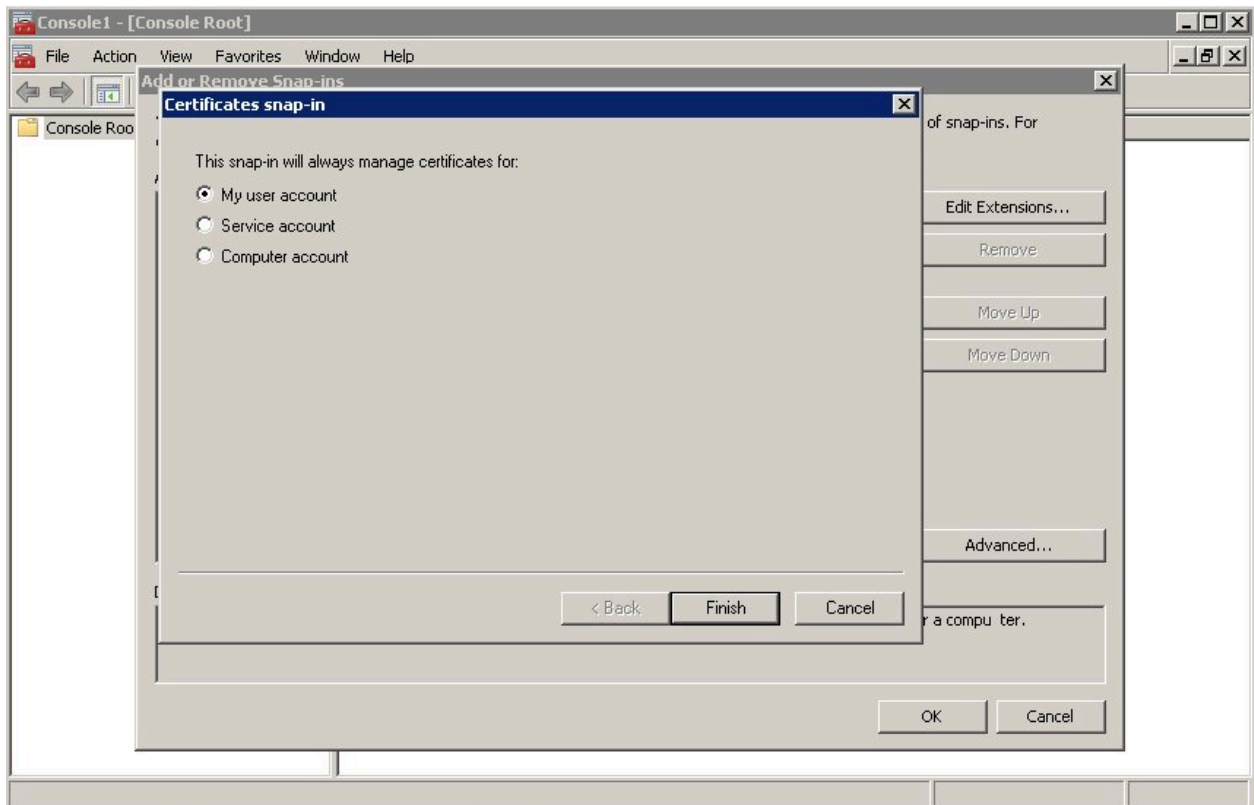
1. Configure and tune TLS for Genesys servers to use certificates signed by the same CA.
2. If the Web Engagement Server is running on a different host, copy the trusted CA certificate to this host.
3. Import the CA certificate to WCS via Certificates Snap-in on the Web Engagement Server host by launching the MMC console. Enter **mmc** at the command line.
4. Select **File > Add/Remove Snap-in...** from the main menu.



5. Select **Certificates** from the list of available snap-ins and click **Add**.

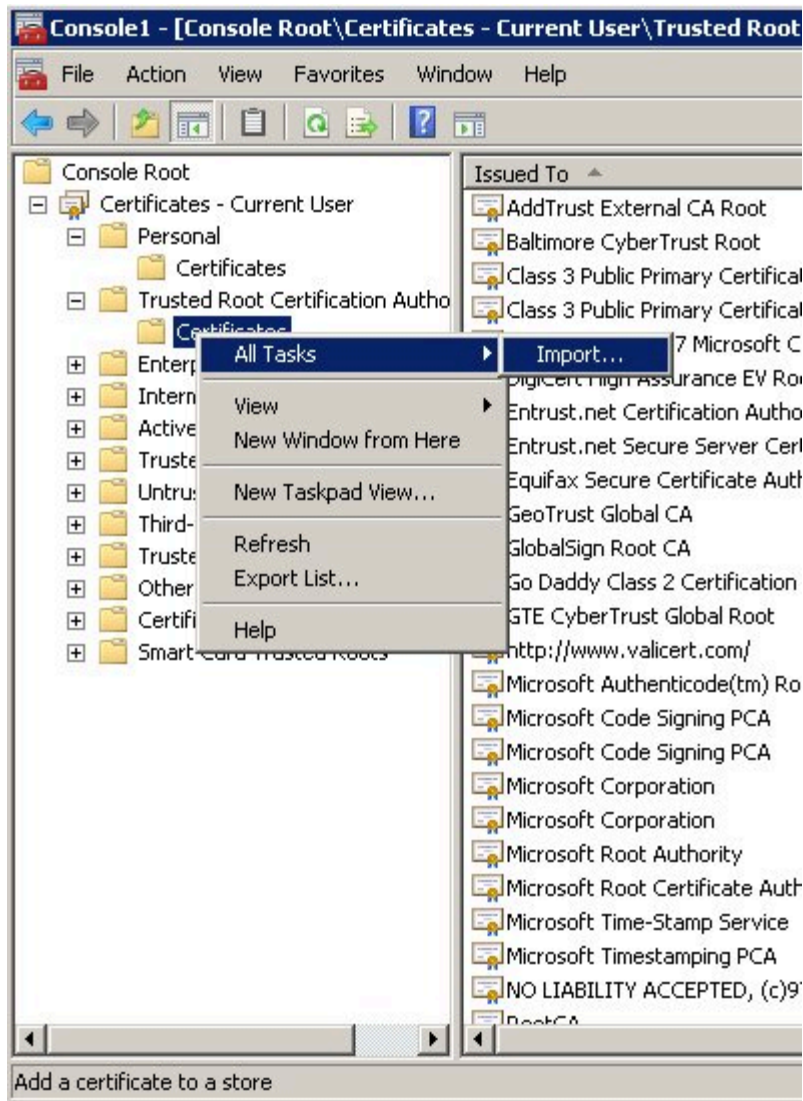


6. Select the account to manage certificates for and click **Finish**. It is important to place certificates under the correct Windows account. Some applications are run as services under the Local Service or System account, while others are run under user accounts. The account chosen in MMC must be the same as the account used by the application which certificates are configured for, otherwise the application will not be able to access this WCS storage.



7. Click **OK**.
8. Import a certificate. Right-click the "Trusted Root Certification Authorities/Certificates" folder and choose All Tasks > Import... from the context menu. Follow the steps presented by the Certificate Import Wizard. Once finished the imported certificate appears in the certificates list.





9. In Genesys Administrator, navigate to **Provisioning > Environment > Applications** and open your Web Engagement Server application.
10. Click the **Options** tab and navigate to the [\[security\]](#) section.
11. Set the **provider** option to MSCAP.
12. Click **Save & Close**.

## End

## Next Steps

- Return to the [Genesys Web Engagement Security](#) page.

---

# Authentication

You can enable secure communications with the **History REST API** by completing the procedures below to implement authentication. If you do enable authentication, then all the clients of the API must use the authentication scheme and credentials. Two common clients of the API are the Genesys Web Engagement Plug-in for Interaction Workspace and the Engagement Strategy. See "Configuring Authentication in Interaction Workspace" and "Configuring Authentication in the Engagement Strategy" for details.

## Configuring Authentication in the Web Engagement Server

Complete the steps below to enable authentication for the History REST API.

### Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the Genesys Web Engagement Server application, and click **Edit...**
2. Click the **Options** tab and scroll down to the **[security]** section.
3. Set the following options:
  - **auth-scheme**
  - **user-id**
  - **password**
4. Click **Save & Close**.

### End

## Configuring Authentication in Interaction Workspace

If you enable authentication for the History REST API and use the Genesys Web Engagement Plug-in for Interaction Workspace, then you must complete the steps below to enable authentication for the plug-in.

### Prerequisites

- You completed "Configuring Authentication in the Web Engagement Server".

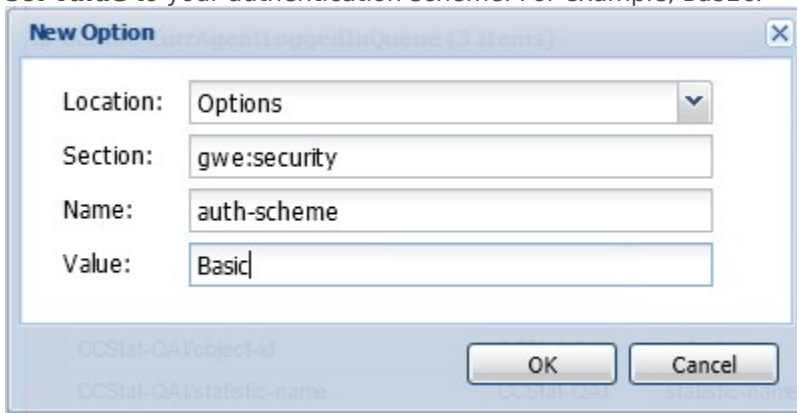
### Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the Interaction Workspace application, and click **Edit...**  
**Note:** Before configuring the authentication options, be sure to read about each option to help

determine the correct values for your deployment:

- **auth-scheme**
- **user-id**
- **password**

2. Click the options tab and then click **New**.
3. In the **New Option** window, configure the following:
  - a. Set **Section** to gwe:security
  - b. Set **Name** to auth-scheme
  - c. Set **Value** to your authentication scheme. For example, Basic.

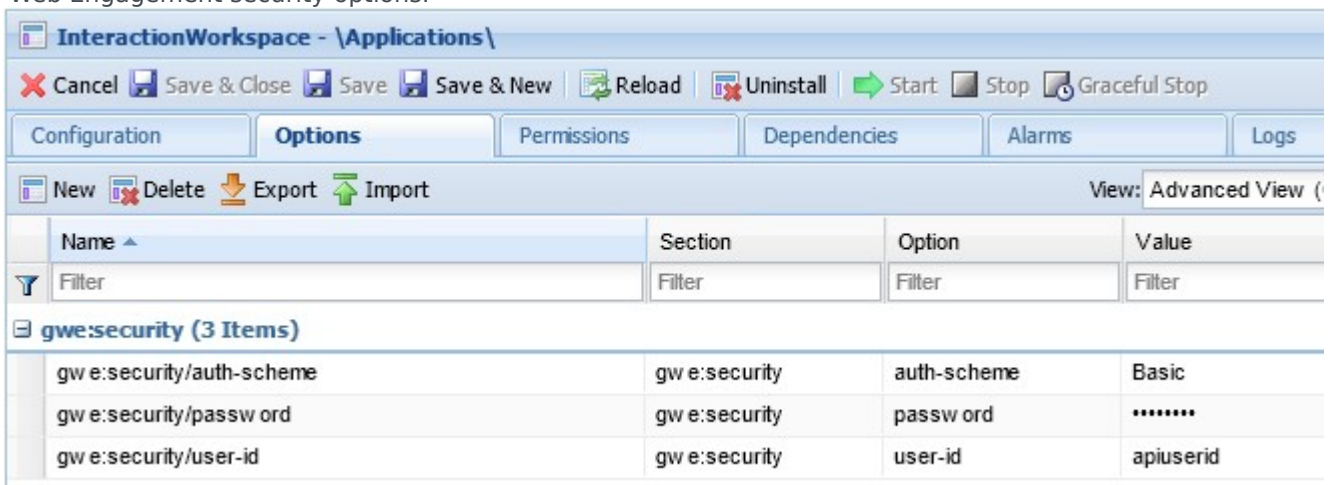


d. Click **OK**

4. Complete steps a-d to configure the remaining security options:

Section	Name	Value
gwe:security	user-id	Your user ID.
gwe:security	password	Your user password.

Your configuration options for Interaction Workspace should now have a new section for the Genesys Web Engagement security options:



---

New security options for Genesys Web Engagement.

5. Click **Save & Close**.

## End

# Configuring Authentication in the Default Engagement Strategy

Complete the steps below to add security credentials to the default SCXML strategy to support authentication for the REST API.

## Prerequisites

- You completed "Configuring Authentication in the Web Engagement Server".
- Your SCXML strategy uses the REST API. See [Customizing the Engagement Strategy](#) for details.

## Start

1. Open the [Engagement Logic strategy](#) in Composer.
2. Open **default.workflow**.
3. Find and set your user and password credentials in the list of properties at the Entry point (Start).
4. Regenerate the SCXML and follow the Web Engagement [application deployment procedure](#).

## End

## Next Steps

- Return to the [Genesys Web Engagement Security](#) page.

# Cassandra Security

Unauthorized access to Cassandra data is possible at several points:

- Direct access via "standard" interfaces: Thrift and CQL
- Access to data traveling through the network
- Access to data files that Cassandra stores on hard drives

Cassandra's default configuration provides mechanisms to secure direct interfaces (through authentication and authorization) and network traffic (through the use of TLS). The data stored on hard drives can be secured either by third-party commercial offerings or with some development investments.

The following sections describe how to provide secure access to:

- [External Cassandra](#)
- [Embedded Cassandra](#)

## External Cassandra

### Securing access interfaces

You can secure your access interfaces based on an authentication and authorization scheme. In other words, Cassandra needs to know:

- Who is trying to access the system
- Whether they are allowed to access the system at all
- If so, which data they should have access to

With the default setup, anybody is allowed to access all the data.

### Authentication

Authentication (who) is managed by the **authenticator** parameter in the **cassandra.yaml** file. By default, GWE 8.5 only provides a login/password authentication scheme with the use of **PasswordAuthenticator**.

### Procedure

#### Start

---

1. Edit **Cassandra installation directory/conf/cassandra.yaml**:
  1. Change the **authenticator** option (described in the [\[cassandraEmbedded\]](#) option section) to PasswordAuthenticator. Note that by default, the **authenticator** option is set to AllowAllAuthenticator.
  2. Tune your **system\_auth** keyspace replication according to the [DataStax system\\_auth documentation](#).
2. Restart the Cassandra client.

The default superuser name and password that you use to start the client is stored in Cassandra:

```
<client startup string> -u cassandra -p cassandra
```
3. Start cqlsh using the superuser name and password (cassandra):

```
./cqlsh -u cassandra -p cassandra
```
4. Deactivate the default superuser, which is called cassandra, as mentioned above. This step is optional but highly recommended.
  1. Create a superuser with another name.
  2. Log in as the newly create superuser.
  3. Change the cassandra user password to something long and incomprehensible, and then forget about it. It won't be used again.
  4. Take away the cassandra user's superuser status.
5. Use the following CQL 3 statements to set up user accounts and then grant permissions to access the database objects:
  - **CREATE USER**
  - **GRANT**
6. Set the new user name and password to the values of the [\[cassandraKeyspace\]](#) **userName** and **password** options for the Web Engagement Cluster application.

## End

## Authorization

Authorization (which data) is managed by the authorizer in the GWE cluster configuration options. Cassandra offers a familiar relational database **GRANT/REVOKE** paradigm to grant or revoke permissions for accessing data.

## Procedure

### Start

Edit **Cassandra installation directory/conf/cassandra.yaml**:

1. Change the **authorizer** option (described in the [\[cassandraEmbedded\]](#) option section) to CassandraAuthorizer. Note that by default, the **authorizer** option is set to AllowAllAuthorizer.
2. Tune your **system\_auth** keyspace replication according to the [DataStax system\\_auth documentation](#). Note that the validity period for permissions caching is 2000 ms.

For more information about permissions see the [DataStax Object permissions documentation](#).

## End

CQL supports these authorization statements:

- [GRANT](#)
- [LIST PERMISSIONS](#)
- [REVOKE](#)

## Resource Access Point Configuration

### Prerequisites

- The Cassandra Resource Access Point applications are created and configured

### Procedure

#### Start

For all Cassandra Resource Access Points:

1. Open the **cassandraClient** (previously **cluster**) configuration option section.
2. Set the **userName** option to the name of an already-created user.
3. Set the **password** option to the user's password.

#### End

## Securing Network Traffic

The client-to-node and node-to-node traffic in your Cassandra deployment may require protection. They can both be secured by using SSL (Secure Sockets Layer) encryption.

### Client-to-Node Encryption

Client-to-node encryption uses SSL to protect data that is traveling from client machines to a database cluster. It does this by establishing a secure channel between the client and the coordinator node.

### Prerequisites

- You must install Java Cryptography Extension (to enable 256-bit encryption).
- All nodes must have all of the relevant SSL certificates. See [Preparing server certificates](#).

## Procedure

### Start

On each node in the [\[cassandraEmbedded\]](#) option section, edit ***Cassandra installation directory/conf/cassandra.yaml***:

1. Set the [client\\_encryption\\_options/enabled](#) option to `true`.
2. Set the appropriate paths to your `.keystore` and `.truststore` files in the [client\\_encryption\\_options/keystore](#) and [client\\_encryption\\_options/truststore](#) options.
3. Provide the required passwords in [client\\_encryption\\_options/keystore\\_password](#) and [client\\_encryption\\_options/truststore\\_password](#). The passwords must match the passwords used when generating the keystore and the truststore.
4. To enable client certificate authentication, set the [client\\_encryption\\_options/require\\_client\\_auth](#) option to `true`.

### End

## Node-to-Node Encryption

Node-to-node encryption uses SSL to protect data being transferred between cluster nodes. This includes node-to-node gossip communication.

## Prerequisites

- You must install Java Cryptography Extension (to enable 256-bit encryption).
- All nodes must have all of the relevant SSL certificates. See [Preparing server certificates](#).

## Procedure

### Start

On each node in the [\[cassandraEmbedded\]](#) option section, edit ***Cassandra installation directory/conf/cassandra.yaml***:

1. Set the [server\\_encryption\\_options/internode\\_encryption](#) option to one of the following:
    - **all**—Cassandra encrypts all internodal traffic
    - **dc**—Cassandra encrypts all traffic between datacenters
    - **rack**—Cassandra encrypts all traffic between racks
  2. Set the appropriate paths to your `.keystore` and `.truststore` files in the [server\\_encryption\\_options/keystore](#) and [server\\_encryption\\_options/truststore](#) options.
  3. Provide the required passwords in [server\\_encryption\\_options/keystore\\_password](#) and [server\\_encryption\\_options/truststore\\_password](#). The passwords must match the passwords used when generating the keystore and the truststore.
-



4. To enable client certificate authentication, set the `server_encryption_options/require_client_auth` option to `true`.

**End**

## Resource Access Point Configuration

**Prerequisites**

- The Cassandra Resource Access Point applications are created and configured.

**Procedure****Start**

For all Cassandra Resource Access Points:

1. Open the properties for the port with an ID of `native`.
2. Set this port to secured.
3. If your Cassandra cluster has its own properly configured security certificate, enable **Mutual TLS** for this port.

**End**

## Web Engagement Application Configuration

**Prerequisites**

- The Web Engagement cluster applications are created and configured.
- The Web Engagement cluster applications have their own properly configured security certificates.

**Procedure****Start**

- For every Web Engagement application:
  1. Make sure its security certificate was correctly configured.
  2. For security fine tuning, read the description of the TLS options in **Configuration Options Reference Manual**.
- For every Web Engagement cluster application:
  - Set the Web Engagement application port to secured.

**End**

---

## Using cqlsh

If you are planning to use the cqlsh standard utility with encryption, please consult the [Datastax documentation](#).

## Embedded Cassandra

### Important

Starting in 8.5.0, Embedded Cassandra mode is deprecated in Web Engagement; support for this mode will be discontinued in 9.0.

## Securing access interfaces

You can secure your access interfaces based on an authentication and authorization scheme. In other words, Cassandra needs to know:

- Who is trying to access the system
- Whether they are allowed to access the system at all
- If so, which data they should have access to

With the default setup, anybody is allowed to access all the data.

### Authentication

Authentication (who) is managed by the **authenticator** parameter in the **cassandra.yaml** file. By default, GWE 8.5 only provides a login/password authentication scheme with the use of **PasswordAuthenticator**.

### Procedure

#### Start

1. Change the `[cassandraEmbedded] authenticator` option for the Web Engagement Cluster application to `PasswordAuthenticator`.  
By default, the **authenticator** option is set to `AllowAllAuthenticator`.
2. Tune your **system\_auth** keyspace replication according to the [DataStax system\\_auth documentation](#).
3. Restart the Cassandra client.  
The default superuser name and password that you use to start the client is stored in Cassandra:  

```
<client startup string> -u cassandra -p cassandra
```

4. Start `cqlsh` using the superuser name and password (cassandra):  
`./cqlsh -u cassandra -p cassandra`
5. Deactivate the default superuser, which is called `cassandra`, as mentioned above. This step is optional but highly recommended.
  1. Create a superuser with another name.
  2. Log in as the newly create superuser.
  3. Change the `cassandra` user password to something long and incomprehensible, and then forget about it. It won't be used again.
  4. Take away the `cassandra` user's superuser status.
6. Use the following CQL 3 statements to set up user accounts and then grant permissions to access the database objects:
  - `CREATE USER`
  - `GRANT`
7. Set the new user name and password to the values of the `[cassandraKeyspace] userName` and `password` options for the Web Engagement Cluster application.

## End

## Authorization

Authorization (which data) is managed by the authorizer in the GWE cluster configuration options. Cassandra offers a familiar relational database **GRANT/REVOKE** paradigm to grant or revoke permissions for accessing data.

## Procedure

### Start

1. Change the `[cassandraEmbedded] authorizer` option for the Web Engagement Cluster application to `CassandraAuthorizer`.  
By default, the **authorizer** option is set to `AllowAllAuthorizer`.
2. Tune your **system\_auth** keyspace replication according to the [DataStax system\\_auth documentation](#).  
Note that the validity period for permissions caching is 2000 ms.  
For more information about permissions see the [DataStax Object permissions documentation](#).

### End

CQL supports these authorization statements:

- `GRANT`
- `LIST PERMISSIONS`
- `REVOKE`

## Securing Network Traffic

The client-to-node and node-to-node traffic in your Cassandra deployment may require protection. They can both be secured by using SSL (Secure Sockets Layer) encryption.

### Client-to-Node Encryption

Client-to-node encryption uses SSL to protect data that is traveling from client machines to a database cluster. It does this by establishing a secure channel between the client and the coordinator node.

#### Prerequisites

- You must install Java Cryptography Extension (to enable 256-bit encryption).
- All nodes must have all of the relevant SSL certificates. See [Preparing server certificates](#).
- To enable client-to-node SSL, you must set the `encryption.client.*` options in the `[cassandraEmbedded]` section.
- All Web Engagement node applications must have correctly configured security certificates. To learn more about how to configure your node application certificates, read the description of the TLS options in [Configuration Options Reference Manual](#).

#### Procedure

##### Start

For each node in the Web Engagement cluster, open the `[cassandraEmbedded]` option section and:

1. Set the `encryption.client.enabled` option to `true`.
2. Set the appropriate paths to your `.keystore` and `.truststore` files in the `encryption.client.keystore` and `encryption.client.truststore` options.
3. Provide the required passwords in `encryption.client.keystorePassword` and `encryption.client.truststorePassword`. The passwords must match the passwords used when generating the keystore and the truststore.
4. To enable client certificate authentication, set the `encryption.client.clientAuth` option to `true`.
5. For security fine tuning, read the description of the TLS options in [Configuration Options Reference Manual](#).

##### End

### Node-to-Node Encryption

Node-to-node encryption uses SSL to protect data being transferred between cluster nodes. This includes node-to-node gossip communication.

#### Prerequisites

---

- You must install Java Cryptography Extension (to enable 256-bit encryption).
- All nodes must have all of the relevant SSL certificates. See [Preparing server certificates](#).
- To enable node-to-node SSL, you must set the `encryption.server.*` options in the `[cassandraEmbedded]` section.

## Procedure

### Start

On each node in the `[cassandraEmbedded]` option section:

1. Set the `encryption.server.internode` option to one of the following:
  - **all**—Cassandra encrypts all internodal traffic
  - **dc**—Cassandra encrypts all traffic between datacenters
  - **rack**—Cassandra encrypts all traffic between racks
2. Set the appropriate paths to your `.keystore` and `.truststore` files in the `encryption.server.keystore` and `encryption.server.truststore` options.
3. Provide the required passwords in `encryption.server.keystorePassword` and `encryption.server.truststorePassword`. The passwords must match the passwords used when generating the keystore and the truststore.
4. To enable client certificate authentication, set the `encryption.server.clientAuth` option to `true`.

### End

#### Using cqlsh

If you are planning to use the `cqlsh` standard utility with encryption, please consult the [Datastax documentation](#).

# Protecting Personally Identifiable Information

Web Engagement allows you to protect personally identifiable information (PII) for all direct requests made to Elasticsearch by way of the HTTP interface, which is normally on port 9200. In particular, you can:

- [Prevent changes to the Elasticsearch Cassandra index](#)
- [Restrict access to sensitive data](#)

**Note:** This protection does not extend to access through the binary interface on port 9300, which is used both by Java clients and for communication between Cassandra nodes.

## Preventing Changes to the Elasticsearch Cassandra Index

You can prevent modifications to the Elasticsearch index by using a read-only filter that prohibits REST calls (POST, PUT, DELETE) on the HTTP interface for all URLs used by Elasticsearch. However, even if you use the filter, Elasticsearch still allows you to make POST or PUT read requests containing JSON content. Here is an example:

```
POST http://server:9200/indexName/someType/_search
{ JSON }
```

You can also make read-only requests to known resources such as `/*.kibana/` and the Web Engagement Reporting Server collector configuration parameter.

The filter uses a regular expression to limit access. The default regex looks like this:

```
/_msearch|/_mget|/_search|/_search_shards|/_suggest|/_count|/_validate|/_explain|\.kibana|
garp\.collector|tenantConfiguration|\.garp|systemConfiguration|/conversion|webtraffic
```

If a request is rejected, the server will respond with 403 "Access Forbidden".

## Application Options

The following options belong to the [\[elasticsearch\]](#) section.

Key	Value	Default	Comment
http-read-only	Boolean	false	Enables read-only mode
read-only-urls-regexp	String	See the default expression mentioned above	Value must be a valid regular expression

## Restricting Access to Sensitive Data

You can also filter all responses from Elasticsearch queries that return data. If one of the top level keys of the JSON source response matches the filter it will be replaced by **\*\*\*\*\***. It is important to note that even a JSON object or a JSON array will be replaced by this string. The length of the string is constant.

Here is the default expression, which is case insensitive:

```
password
```

### Unfiltered Response

```
{
  "took":90,
  "timed_out":false,
  "_shards":{
    "total":5,
    "successful":5,
    "failed":0
  },
  "hits":{
    "total":1,
    "max_score":1.0,
    "hits":[
      {
        "_index":"cointr",
        "_type":"user",
        "_id":"1",
        "_score":1.0,
        "_source":{
          "postDate":"2015-06-24",
          "password":"secret",
          "user":"jh"
        }
      }
    ]
  }
}
```

### Filtered Response

```
{
  "took":1,
  "timed_out":false,
  "_shards":{
    "total":5,
    "successful":5,
    "failed":0
  },
  "hits":{
    "total":1,
    "max_score":1.0,
    "hits":[
      {
        "_index":"cointr",
        "_type":"user",
        "_id":"1",
        "_score":1.0,

```

```

    "_source":{
      "postDate":"2015-06-24",
      "password":"*****",
      "user":"jh"
    }
  ]
}

```

## Application Options

The following options belong to the [\[elasticsearch\]](#) section.

Key	Value	Default	Comment
http-filter	Boolean	false	Enables response filter mode
reponse-filter-regexp	String	password	Value must be a valid regular expression and is not case sensitive
reponse-filter-regexp-type	String	REGEXP	Regular expression type: <b>REGEXP</b> —standard Java regular expression <b>PATH</b> —JSON path expression



# Configuring Specific Features

Genesys Web Engagement includes additional features that you can configure to enable the following functionality:

- **Pacing Algorithm**—Configure this algorithm to help keep a balanced workflow in your contact center by aligning the generated Web Engagement invites with the agents you have available for both proactive and reactive traffic.
- **Chat Channel**—Manually configure the Web Engagement Server and Chat Server to support a chat channel or specify chat as the default channel of engagement.
- **Web Callback Channel**—Manually configure the Web Engagement Server to support a web callback channel or specify web callback as the default channel of engagement.
- **GeoIP Information**—You can configure Web Engagement Server to collect information about a visitor's IP address, as well as converting this information into the visitor's geolocation coordinates.

---

# Pacing Algorithm

Genesys Web Engagement includes an intelligent pacing algorithm that predicts when and how many new Web Engagement invites should be generated to reach an optimization goal that you can set. The pacing algorithm can predict agent availability for both proactive (outbound) and reactive (inbound) traffic and can help keep a balanced workflow in your contact center by avoiding the extremes of either overloading or underloading the system.

## About pacing algorithms

- For a more in-depth look at the pacing algorithm and the usage methodology behind it, click [HERE](#) to download the "Pacing Algorithm Usage Methodology" Word document. See download tips
- If you want to combine proactive (outbound) and reactive (inbound) traffic, find out more about how Web Engagement handles [dual pacing](#).
- You can also customize the way the pacing algorithm is used in the Engagement Logic Strategy. For details, see [Accessing Pacing Information from the Engagement Logic Strategy](#).

To enable the pacing algorithm, complete the procedures below.

## Configuring your optimization model

**Note:** Starting with version 8.5.000.33, Web Engagement allows you to use [Agent Group](#) objects as the primary way to configure pacing settings. You can still configure these settings in the Web Engagement Cluster or Web Engagement Server application objects—as shown in this section and the following section—but settings configured in the [Agent Group](#) objects have a higher priority.

Complete this procedure to enable the pacing algorithm to predict traffic for your Web Engagement solution.

### Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. Edit the Web Engagement Cluster application and click the **Options** tab.
3. In the [\[pacing\]](#) section, set the [algorithm](#) option. The supported values are:
  - SUPER\_PROGRESSIVE — Recommended for small agent groups (1-30 agents); only proactive traffic is predicted.
  - SUPER\_PROGRESSIVE\_DUAL — Recommended for small agent groups (1-30 agents); both proactive and reactive traffic are predicted.
  - PREDICTIVE\_B — Recommended for bigger agent groups (start from 30 agents); only proactive traffic is predicted

- PREDICTIVE\_B\_DUAL — Recommended for bigger agent groups (start from 30 agents); both proactive and reactive traffic are predicted.
4. Set the **optimizationTarget** option. The supported values are:
    - ABANDONMENT\_RATE — Percentage of interactions that will be abandoned.
    - BUSY\_FACTOR — Percentage of time during which an agent plans to be busy with an interaction-related activity.
  5. Set the **optimizationGoal** option. The value you set for this option depends on the value you set for **optimizationTarget**:
    - If your optimization target is ABANDONMENT\_RATE, Genesys recommends that you use small values. For example, from 3 to 5.
    - If your optimization target is BUSY\_FACTOR, Genesys recommends that you use big values. For example, from 70 to 85.
  6. If you chose a dual algorithm in Step 3 (SUPER\_PROGRESSIVE\_DUAL or PREDICTIVE\_B\_DUAL), specify a value for the **proactiveRatio** option.  
 This option basically controls the minimal percentage of agents that will be reserved for processing proactive engagement interactions.
  7. Set the **refreshPeriod** option. This option controls how frequently the pacing algorithm provides predictions. Genesys recommends that you use values from 1 to 5.

Name	Section	Option	Value
Filter	Filter	Filter	Filter
<b>pacing (7 Items)</b>			
pacing/algorithm	pacing	algorithm	SUPER_PROGRESSIVE
pacing/chatGroups	pacing	chatGroups	Web Engagement Chat
pacing/optimizationGoal	pacing	optimizationGoal	3
pacing/optimizationTarget	pacing	optimizationTarget	ABANDONMENT_RATE
pacing/proactiveRatio	pacing	proactiveRatio	0
pacing/refreshPeriod	pacing	refreshPeriod	5
pacing/voiceGroups	pacing	voiceGroups	

Pacing algorithm settings

**End**

**Next Steps**

- [Configuring the Agent Groups](#)

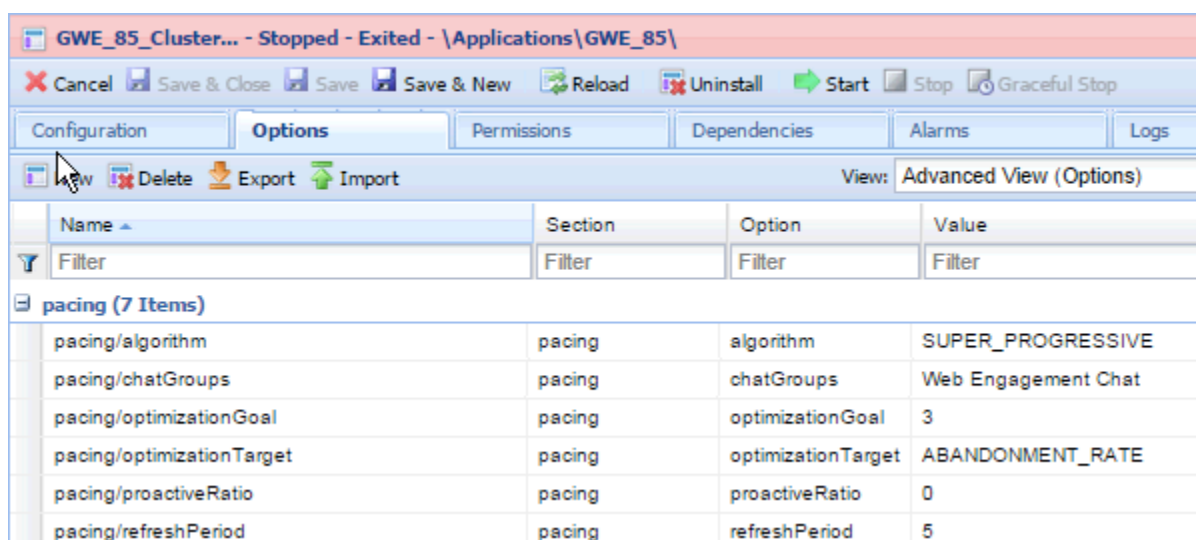
## Configuring the agent groups

In the previous procedure, you set the **algorithm** option according to the size of your agent group and the type of traffic the algorithm should handle. In this procedure, you configure your agent groups for the pacing algorithm to use.

When you install Genesys Web Engagement, the **Provisioning Tool** automatically creates two agent groups:

- Web Engagement Chat
- Web Engagement Voice

In the steps below, you'll confirm that the agent groups were created and then add agents to them.



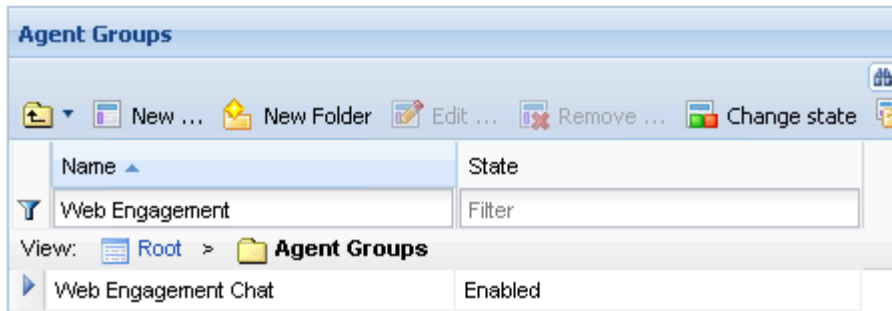
Configure new agent groups

### Important

You can use your own groups instead by changing the values of the **chatGroups** and **voiceGroups** options to the names of other groups you configured.

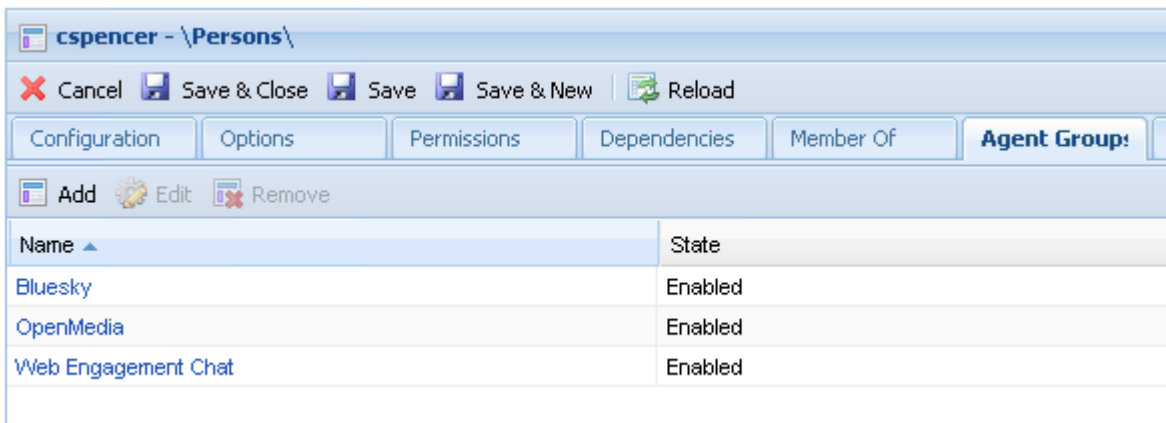
### Start

1. Open Genesys Administrator and navigate to **Provisioning > Accounts > Agent Groups**. Make sure that the Web Engagement Chat and Web Engagement Voice groups are created. You can use the filter to display the groups.



The default agent groups

2. Navigate to **Provisioning > Accounts > Users.**
3. Select an agent that should manage Web Engagement interactions and click **Edit...**
4. Select the Agent Group tab and click **Add.** The Browse dialog opens.
5. Select one of the groups and click **OK.**



The agent named Spencer now belongs to the Web Engagement Chat group.

6. Repeat Steps 3-5 for each agent you want to add to the chat or voice agent groups.

**End**

## Using Agent Groups to configure pacing

Starting with version 8.5.000.33, Web Engagement allows you to use Agent Group objects as the primary way to configure pacing settings. You can still configure these settings in the Web Engagement Cluster or Web Engagement Server application objects, but settings configured in the Agent Group objects have a higher priority.

**Note:** You must **enable** the **Access to Web Engagement Pacing Configuration** privilege for any users who need to control pacing settings.

Select the **Pacing Configuration** menu **(1)** in GAX to configure your other agent groups. **Note:** You

can still configure these options in the Web Engagement Cluster or Web Engagement Server application objects, but options configured in the Agent Group objects have a higher priority.

The checkbox in the **Used for Pacing** column (2) shows whether a group uses pacing:

The screenshot shows the GAX Web Engagement interface. The top navigation bar includes: GAX, Dashboard, Configuration, Routing Parameters, Reports, Administration, and Web Engagement. The main content area is titled "Agents Groups". Below the title is a search bar labeled "Quick Filter" and a refresh button. A table lists several agent groups with checkboxes in the "Used for Pacing" column. A circled '2' points to the "Used for Pacing" column header. A dropdown menu is open on the right, showing "Categories", "Script Generator", and "Pacing Configuration" (highlighted in blue). A circled '1' points to the "Pacing Configuration" menu item.

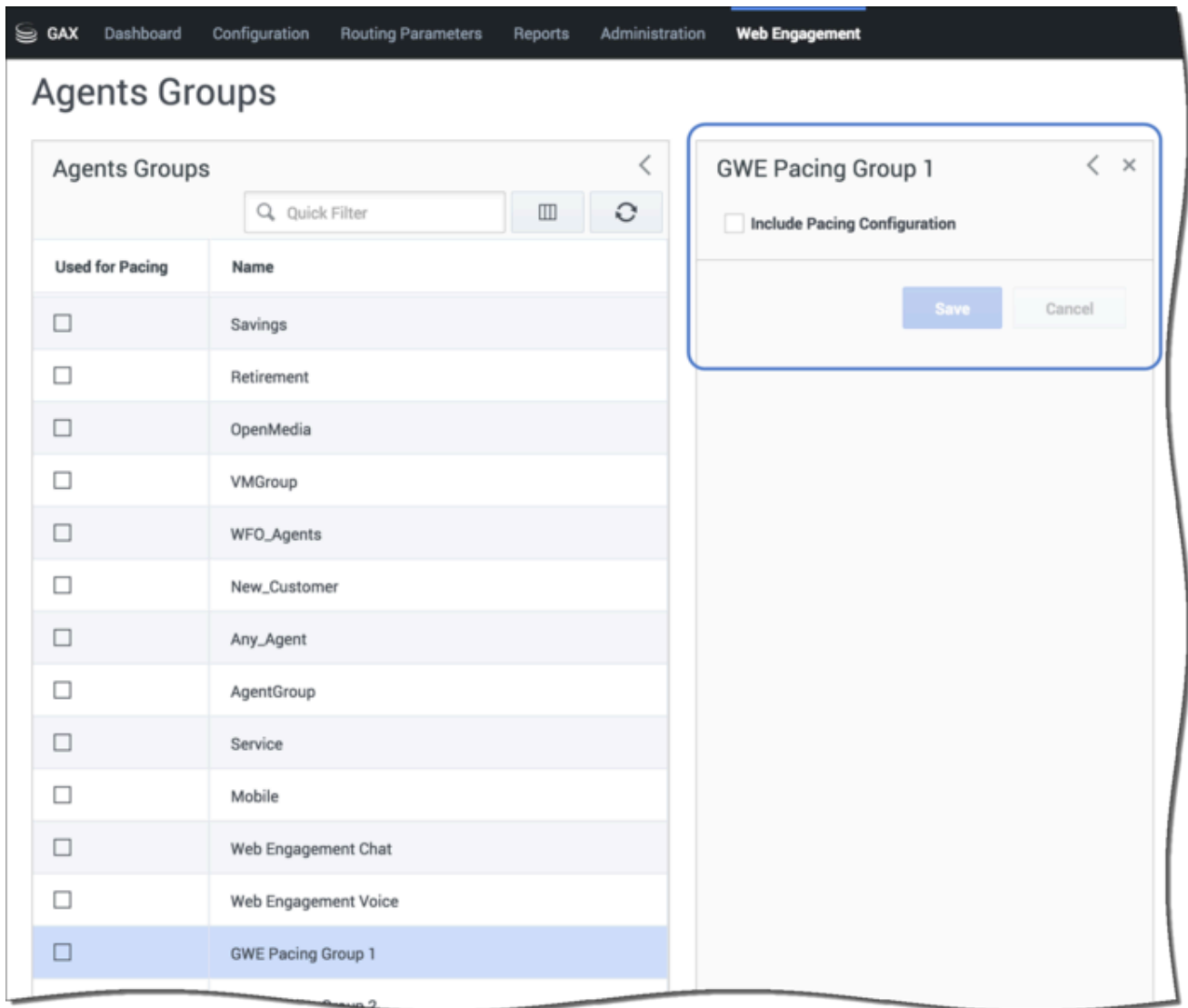
Used for Pacing	Name
<input type="checkbox"/>	Billing
<input type="checkbox"/>	GVPSIP_Ports
<input type="checkbox"/>	E-mail distribution for processing
<input type="checkbox"/>	E-mail QA review group
<input type="checkbox"/>	Supervisors
<input type="checkbox"/>	Chat distribution for processing

You can sort the columns by clicking their headings:

## Agents Groups

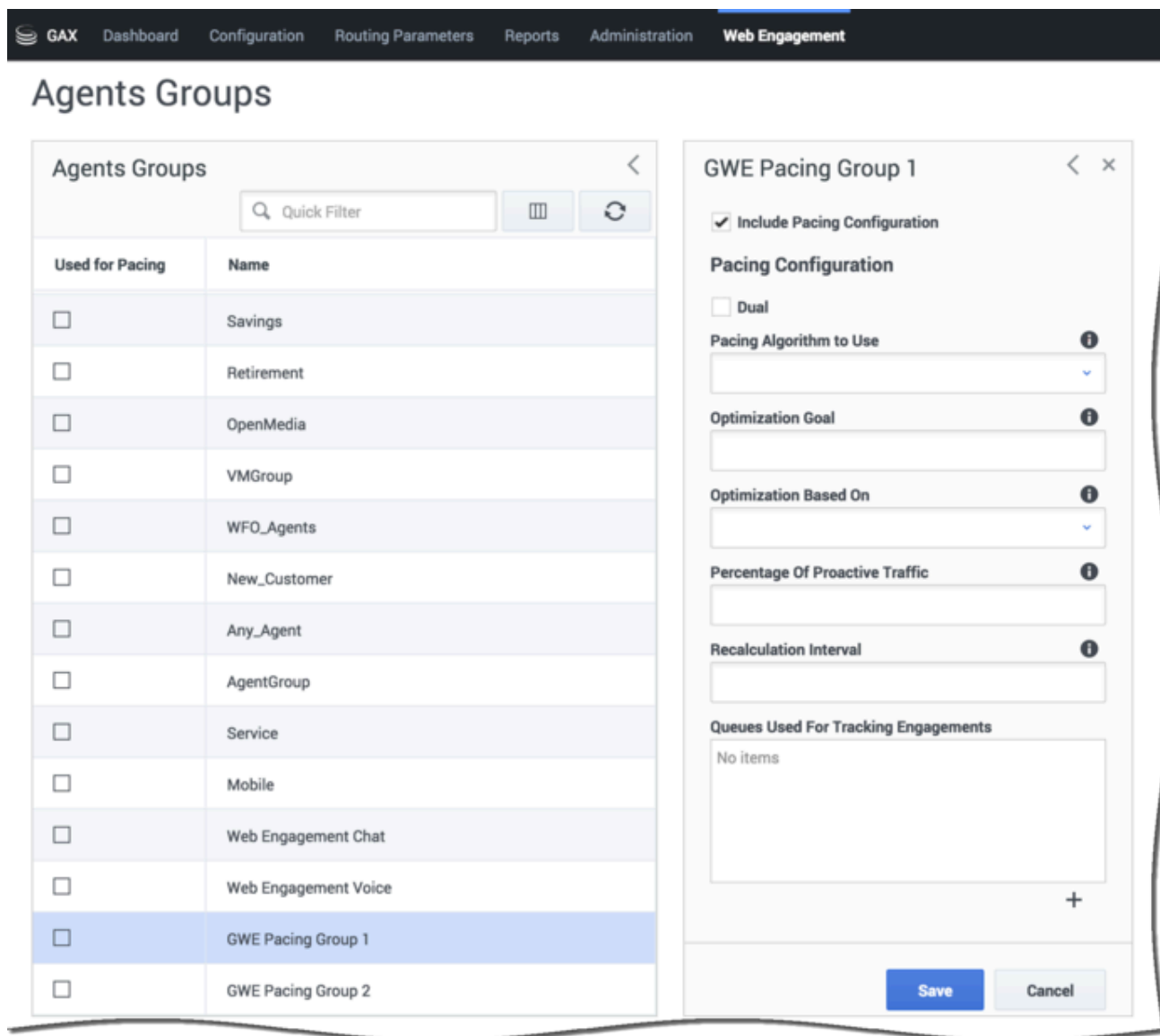
Agents Groups	
<input type="text" value="Quick Filter"/> <span>☰</span> <span>↻</span>	
Used for Pacing	Name
<input type="checkbox"/>	Retirement
<input type="checkbox"/>	OpenMedia
<input type="checkbox"/>	VMGroup
<input type="checkbox"/>	WFO_Agents
<input type="checkbox"/>	New_Customer
<input type="checkbox"/>	Any_Agent
<input type="checkbox"/>	AgentGroup
<input type="checkbox"/>	Service
<input type="checkbox"/>	Mobile
<input type="checkbox"/>	Web Engagement Chat
<input type="checkbox"/>	Web Engagement Voice
<input checked="" type="checkbox"/>	GWE Pacing Group 1
<input checked="" type="checkbox"/>	GWE Pacing Group 2

To turn on pacing for a group, select it from the list. The pane that appears to the right will indicate whether or not pacing is enabled for that group:



You can click the **Include Pacing Configuration** checkbox to include or exclude the group. If you include the group, the **Pacing Configuration** pane appears:





You can use this pane to select the pacing options listed in the next section.

### Pacing Options

Dual	Check this to enable dual pacing. For proactive pacing only, uncheck it.
Pacing Algorithm To Use	Specifies the type of pacing algorithm to be used by the system. Changes take effect after server restart.
Optimization Goal	Specifies the optimal percentage value for your chosen optimization target, which can be either

	ABANDONMENT_RATE or BUSY_FACTOR, as specified in the <b>Optimization Based On</b> field that appears just below this field.
Optimization Based On	Specifies whether to optimize based on ABANDONMENT_RATE or BUSY_FACTOR. Changes take effect after server restart.
Percentage Of Proactive Traffic	Specifies the minimum percentage of agent resources that are reserved for handling proactive interactions. If 0 is specified, no resources are specifically allocated to handle proactive interactions, but proactive traffic is still allowed.  If 100 is specified, all resources are allocated to handle proactive interactions and no reactive interactions are allowed.
Recalculation Interval	Specifies the frequency, in seconds, of predictions produced by the pacing algorithm. Changes take effect after server restart.
Queues Used For Tracking Engagements	A list of the channels and queues used for tracking engagements. <b>Note:</b> <ul style="list-style-type: none"> <li>You must have at least one queue in this list.</li> <li>This list can only contain interaction queues. Virtual queues are not supported.</li> </ul>

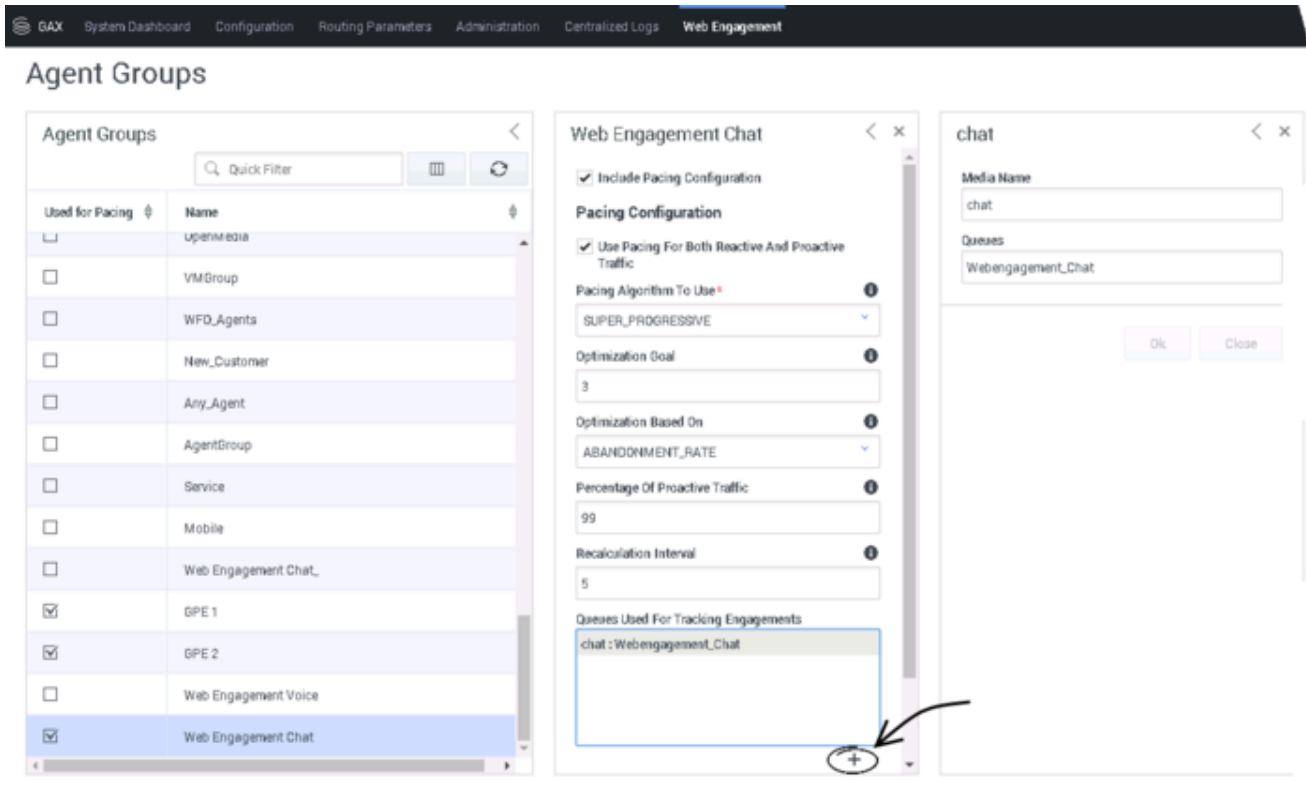
## Associating media types and queues

You can pair each media type with a queue that tracks engagements for that media type.

### Important

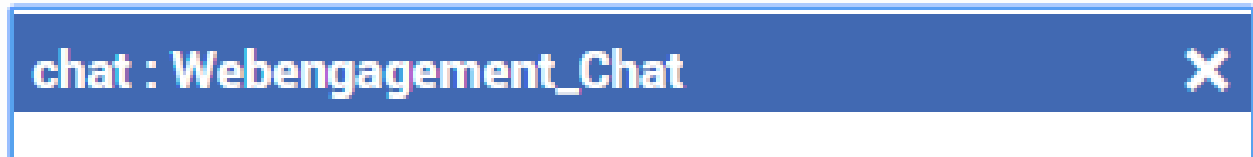
- You must specify at least one media-queue pair.
- A media-queue pair can only specify one media type and one queue.
- Web Engagement 8.5 supports **chat** and **webcallback** medias.
- You can only specify interaction queues. Virtual queues are not supported.

To add a new media-queue pair, click the plus button, then fill out the resulting form, and save it:



To delete a media-queue pair from the list, click the checkbox that appears on the right side of the list item:

## Queues Used For Tracking Engagements



### Next Steps

- Return to the [Genesys Web Engagement Features](#) page.

# Chat Channel

When you install Genesys Web Engagement, the [Provisioning Tool](#) automatically configures the Web Engagement Cluster and Chat Server to support a chat channel for routing chat interactions.

If you need to, you can configure this manually by completing the "Configuring the Web Engagement Cluster and Chat Server to Support a Chat Channel" procedure.

You can also complete the "Configuring Chat as the Default Channel of Engagement" to specify chat as the default channel of engagement. You do this by using the [defaultEngagementChannel](#) option, which is only intended for development purposes and should not be used in a production environment because it turns off the pacing algorithm.

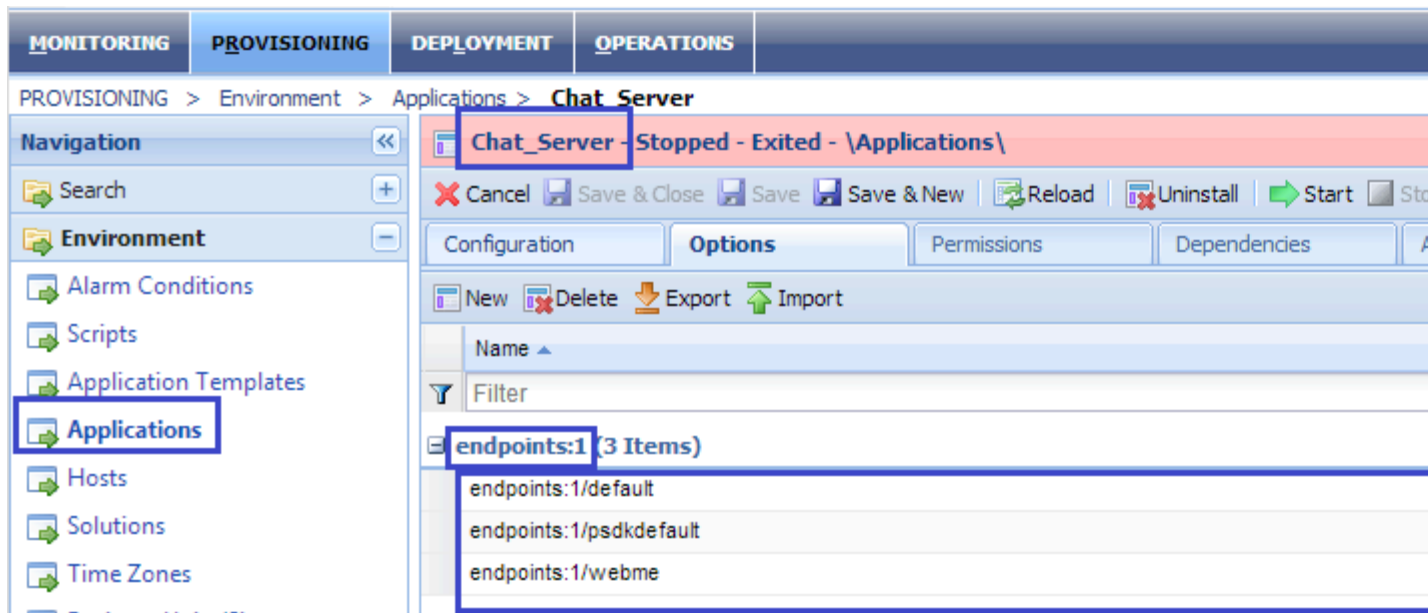
## Configuring the Web Engagement Cluster and Chat Server to Support a Chat Channel

### Prerequisites

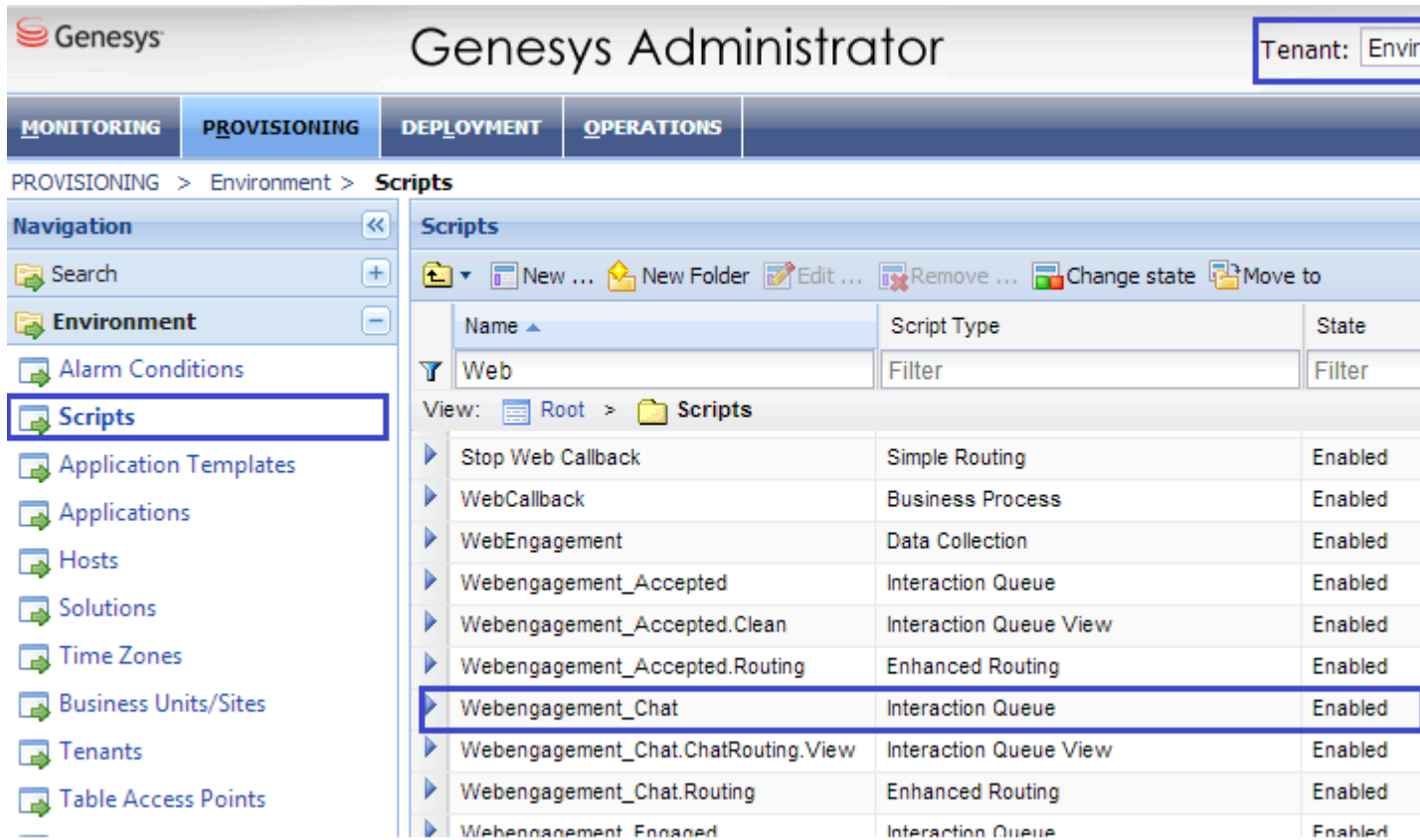
- On your Web Engagement Cluster application, you have a connection to one of following:
  - Chat Server — See [Configuring the Cluster Application](#) for details.
  - A cluster of Chat Servers — See [Configuring a Connection to a Cluster of Chat Servers \(Optional\)](#) for details.

### Start

1. In Genesys Administrator, open your Chat Server application - either the one you connected to directly on the Web Engagement Cluster, or the Chat Server on your Application Cluster (you must complete the following steps for each Chat Server application on your Application Cluster).
2. Select the **Options** tab and find the endpoints section for your tenant: **[endpoints:tenant ID]**. For example, if Chat Server works with the Environment tenant, there should be a section called **[endpoints:1]**.
3. Set the endpoint value for the **endpoints:tenant ID/webme** option to the name of the Interaction Queue where the chat interaction should be placed.  
**Note:** Each Interaction Queue can be related to one routing strategy, either Orchestration Server or Universal Routing Server.



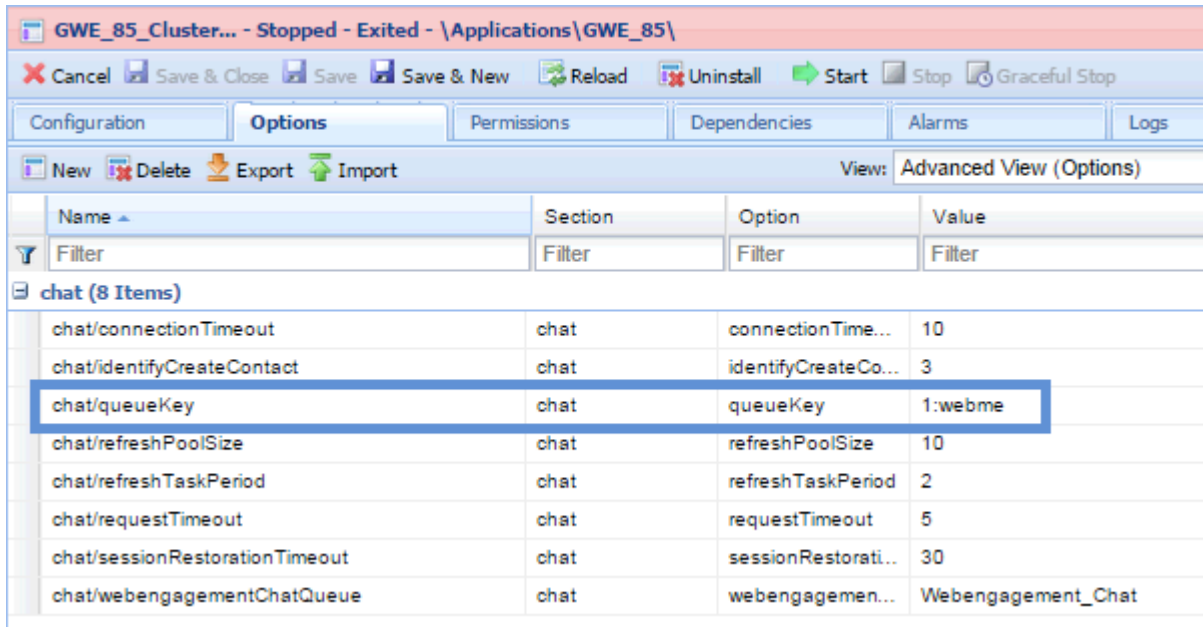
The **webme** option is set to Webengagement\_Chat



The Webengagement\_Chat Interaction Queue

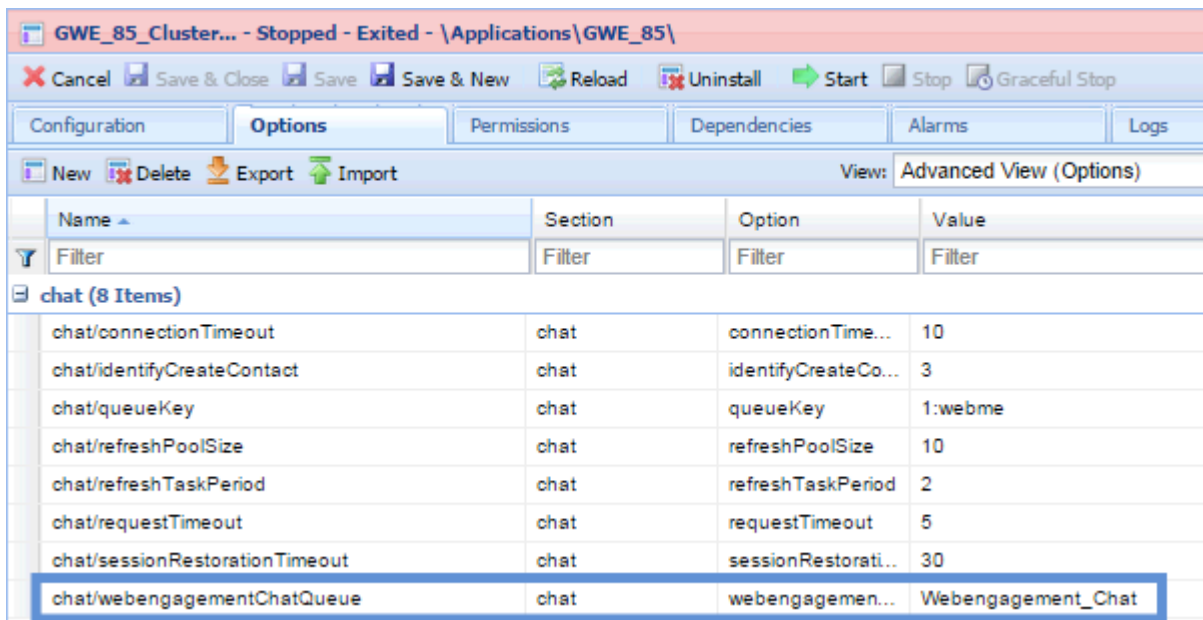
- Configure the Chat Server endpoint for the Web Engagement Web Engagement Cluster application by opening the Web Engagement Cluster application and select the **Options** tab. In the **[chat]** section, set the value of the **queueKey** option to the name of the endpoint you specified in the Chat Server

application option in Step 3. The format is *tenant ID:endpoint name*.



The **queueKey** option is set to 1:webme

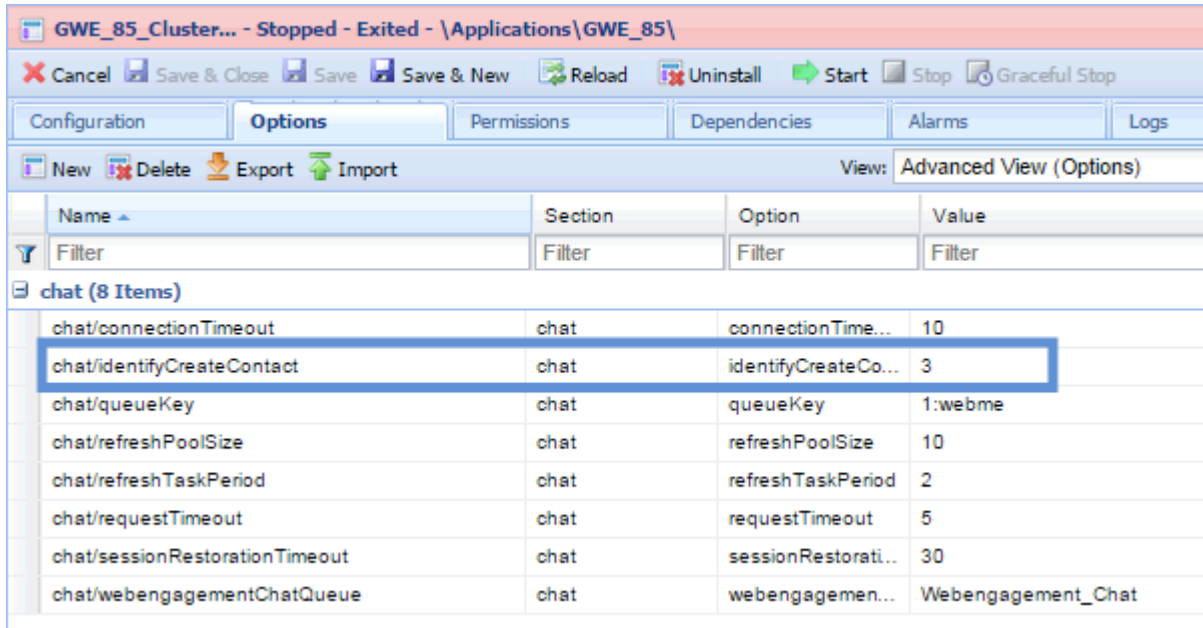
- Specify the Interaction Queue that is used as a starting point to route chat interactions. In the [chat] section, set the value of the **webengagementChatQueue** option to the same queue you specified for the Chat Server endpoint in Step 3.



The **webengagementChatQueue** option is set to Webengagement\_Ch...

- Configure how contact management will behave when a chat session is instantiated. In the [chat] section, set the **identifyCreateContact** option to one of the following values:
  - 1** — Do not identify and do not create a contact

- **2** — Identify, but do not create a contact
- **3** — Identify and create a contact (if absent).

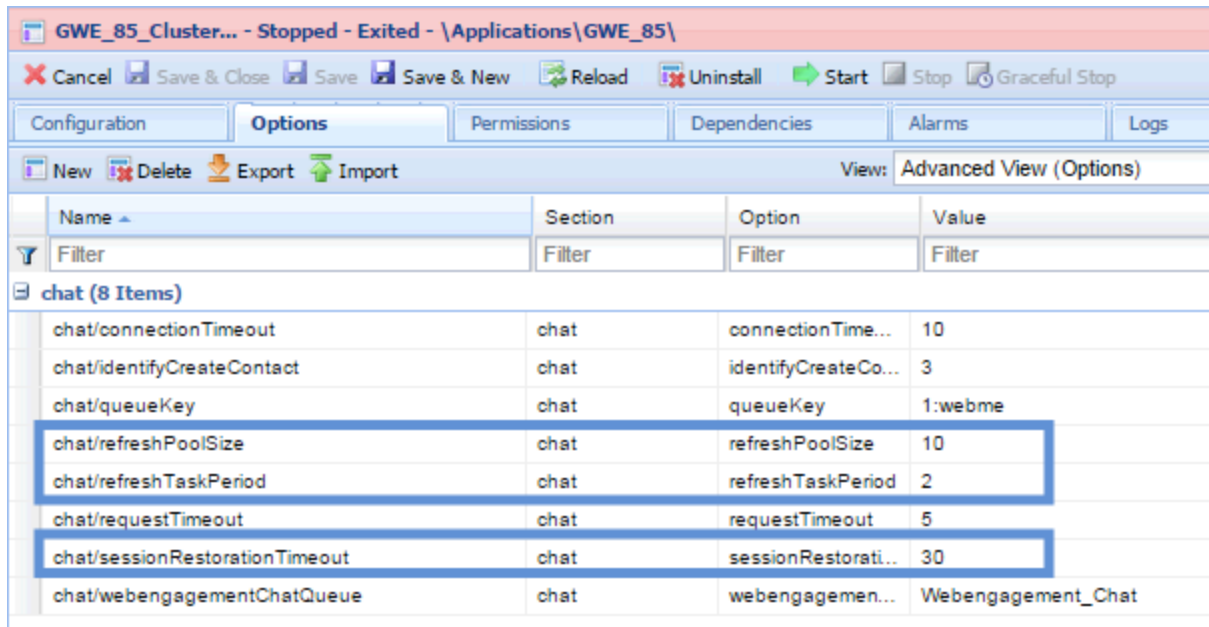


The **identifyCreateContact** option is set to 3

The default value (3) is applied if the option is absent or specified incorrectly.

**Note:** Your Chat Server must have a connection to Universal Contact Server in order to control contact management through the chat session.

7. Configure the chat session behavior by setting the following three options in the **[chat]** section:
  - **refreshTaskPeriod** — Specifies the frequency (in seconds) of chat session updates in the chat widget. The allowed range is from 1 to 5 seconds.
  - **refreshPoolSize** — Specifies the count of threads that serve the communication between the chat widgets and Chat Server(s)
  - **sessionRestorationTimeout** — Specifies the timeout (in seconds) during which Genesys Web Engagement tries to restore a broken chat session if the Chat Server becomes unavailable.



Chat-related options in the chat section

**End**

## Configuring Chat as the Default Channel of Engagement

Specifying chat as the default channel tells the default SCXML strategy to ignore results provided by the pacing algorithm. As a result, the engagement attempt is always activated on the chat channel and the count of ready agents is ignored. If the `defaultEngagementChannel` option is not specified or specified with empty value, the pacing algorithm is used.

**Start**

1. In Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. Open the application for the Web Engagement Cluster.
3. Set the chat channel as the default channel of engagement. In the `[engagement]` section, set the value of the `defaultEngagementChannel` option to `proactiveChat`.

**End**

**Next Steps**

- Return to the [Genesys Web Engagement Features](#) page.



# Web Callback Channel

When you install Genesys Web Engagement, the [Provisioning Tool](#) automatically configures the Web Engagement Server to support working with a voice (web callback) channel.

If you need to, you can configure this manually by completing the "Configuring the Web Engagement Server to Support a Voice (web callback) Channel".

You can also complete the "Configuring Web Callback as the Default Channel of Engagement" to specify web callback as the default channel of engagement. You do this by using the [defaultEngagementChannel](#) option, which is only intended for development purposes and should not be used in a production environment because it turns off the pacing algorithm.

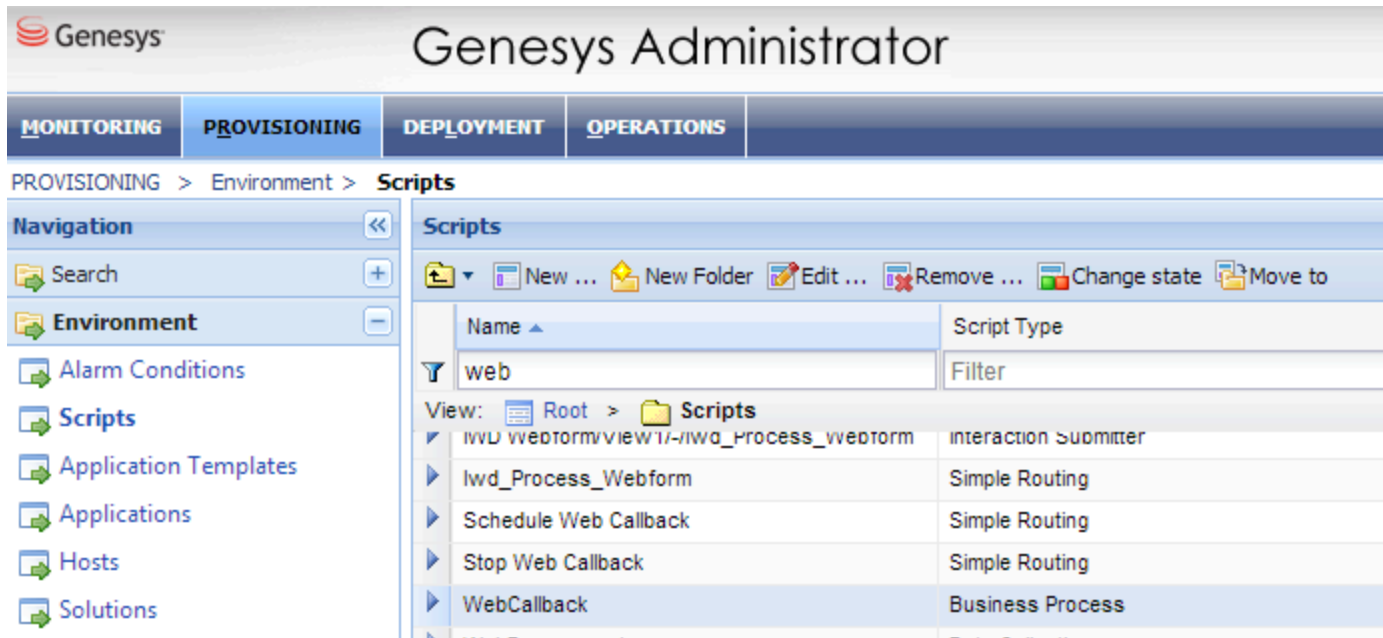
## Configuring the Web Engagement Server to Support a Voice (web callback) Channel

### Prerequisites

- You installed the eServices Web API Samples in your environment. During this installation, you created the web callback routing process and an incoming Interaction Queue named **New**. For details, see the [eServices Deployment Guide](#) and the [Web API Client Developer's Guide](#).

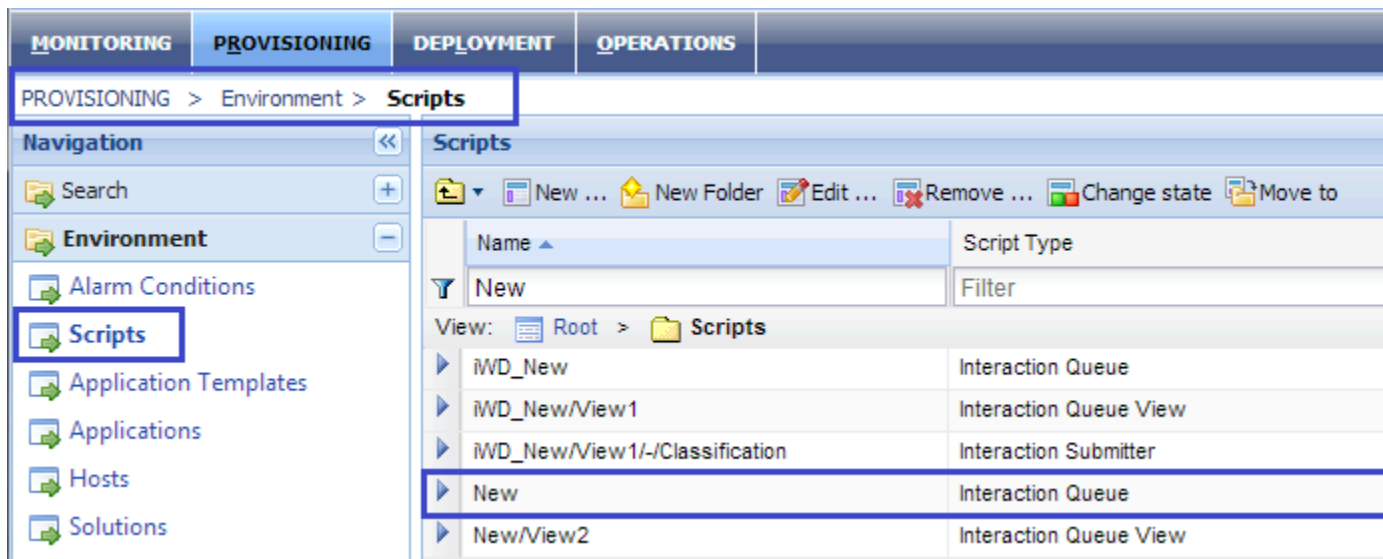
### Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Scripts** and confirm the following:
  - The Web Callback business process is installed:



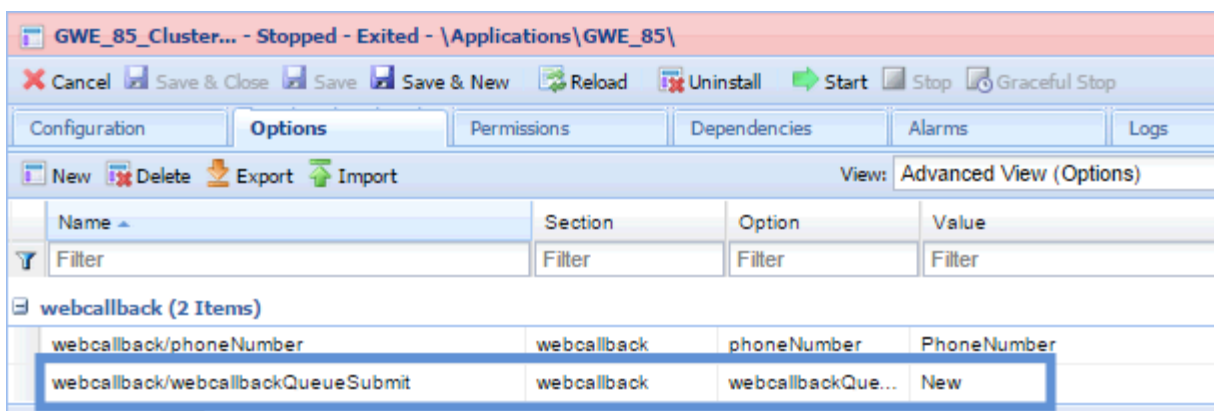
The Web Callback business process

- An Interaction Queue named New exists:



An Interaction Queue named New

2. Navigate to **Provisioning > Environment > Applications** and open the Web Engagement Server application to configure the web callback endpoint.
3. Select the **Options** tab.
4. In the `[webcallback]` section, set the value of the `webcallbackQueueSubmit` option to New.



Set the **webcallbackQueueSubmit** option to New

5. Make sure that agents who participate in web callback processing belong to the following groups:
  - *WebCallback distribution for processing* — This group is used in the Web Callback business process as the delivery target for web callback interactions.
  - *Web Engagement Voice* — This group (or groups) is specified in the **voiceGroups** option to support pacing algorithm functionality.

**End**

## Configuring Web Callback as the Default Channel of Engagement

Specifying web callback as the default channel tells the default SCXML strategy to ignore results provided by the pacing algorithm. As a result, the engagement attempt is always activated on the web callback channel and the count of ready agents is ignored. If the **defaultEngagementChannel** option is not specified or specified with empty value, the pacing algorithm is used.

**Start**

1. In Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. Open the application for the Web Engagement Server.
3. Set the web callback channel as the default channel of engagement. In the **[engagement]** section, set the value of the **defaultEngagementChannel** option to proactiveVoice.

**End**

**Next Steps**

- Return to the [Genesys Web Engagement Features](#) page.

## Limitations in webcallback channel support

GWE 8.5 does not support simultaneous webcallback engagements for the same visitID. To address

this, do either of the following things:

- Modify the invitation widget to stop it from triggering simultaneous webcallback requests.
- Use the `strictEngagementMode` option to prevent simultaneous engagement attempts without differentiating between different channels.

# GeoIP Information

## Turning on Geolocation

Web Engagement Server 8.5 can collect information about a visitor's IP address, as well as converting this information into the visitor's geolocation coordinates. This capability is turned off by default.

To convert IP addresses into geolocation coordinates, Web Engagement Server uses a library and a GeoDB file (**GeoLite2-City.mmdb**) provided by [MaxMind](#).

Web Engagement installs **GeoLite2-City.mmdb** automatically into **Web Engagement installation dir\apps\your application\resources\geo**. However, you may need to upgrade it to the most recent version. To do that, follow these steps:

1. Download the **GeoLite2-City.mmdb** file from [MaxMind](#).
2. Place the downloaded file into **Web Engagement installation dir\apps\your application\resources\geo**.
3. Deploy your application.

Before you start your Web Engagement Server, make sure that the following options from the [\[privacy\]](#) section have the appropriate values:

- **collectActualIPs** = true
- **collectForwardedIPs** = true
- **geoMode** = COUNTRY, CITY, or LOCATION

## Verifying Geolocation Coordinates

There are two ways to determine whether your geolocation coordinates are being calculated properly:

- Use the Advanced Reporting Dashboard and view the results in the map widget
- Inspect the Cassandra database and find one or more **VisitStarted** events

If the transformation is working, the **data** object of your **VisitStarted** events will contain one of the following fields, depending on the value of the **geoMode** option: **location**, **city**, or **countryCode**.

---

# Monitoring Web Engagement Server

Web Engagement provides access to **metrics** and other key performance indicators (KPIs).

It also gives you the ability to configure Message Server **alarms** when a KPI passes its threshold value.

## Web Engagement Metrics

Starting with release 8.5.000.13, Web Engagement integrates with the third-party **Metrics Java library** to keep track of several Web Engagement metrics. The Metrics toolkit includes counters, timers, histograms, and gauges.

You will probably want to use Java Management Extensions (JMX) as your main way of reporting on these metrics. We show how to do that [here](#). Or you may want to check out some of the **other tools** that are available.

You can also use REST—which is helpful for performance testing—or write your metrics to a log file or to the console.

## Available Metrics

Web Engagement Server generates the following kinds of metrics:

- Cache usage statistics
- Event processing time statistics
- A flag for incorrect load balancing
- A flag for interactions that are running too long

Metric name	Description
<CacheName>.CacheSize	Returns the approximate number of entries in this cache
<CacheName>.EvictionCount	Returns the number of times an entry has been evicted
<CacheName>.HitCount	Returns the number of times that cache lookup methods have returned a cached value
<CacheName>.LoadExceptionCount	Returns the number of times that cache lookup methods have thrown an exception while loading a new value
<CacheName>.LoadSuccessCount	Returns the number of times that cache lookup

Metric name	Description
	methods have successfully loaded a new value
<CacheName>.MissCount	Returns the number of times that cache lookup methods have returned an uncached (newly loaded) value
<CacheName>.TotalLoadTime	Returns the total number of nanoseconds the cache has spent loading new values
<CacheName>.isFull	Returns true if the cache is full; otherwise returns false
monitoring.event.timer	Provide event-processing statistics
lb.routing.correct	Returns true if a sticky cookie has a valid application name; otherwise returns false
interaction.duration.exceeded	Returns true if a media interaction has been processed for too long; otherwise returns false

The following cache names are available:

- DroolsSessionCache
- EventsCache
- IxnProfileCache
- VisitProfileCache

For example, the cache size for the events cache will be available as a metric called **EventsCache.CacheSize**.

## Web Engagement Alarms

Web Engagement lets you use tools from the [Genesys Management Layer](#) for monitoring and controlling your applications. These tools can be an important factor in improving performance—especially **alarms**, which let you set performance thresholds for these key metrics:

- Event duration
- Incorrect load balancer routing
- Interaction duration
- **VisitProfile** cache size
- **IxnProfile** cache size
- **Events** cache size
- **DroolsSession** cache size
- Garbage collection latency
- Heap memory usage

## Alarm Configuration



Alarm name	Alarm description	Alarm Condition object					Related configuration option
Threshold type	Selection mode	Application type	Detect Event ID	Cancel Event ID			
Event Duration	Indicates that the time spent processing the event was greater than the specified limit.	predefined	Select by Application Type	Web Engagement Backend Server	100601	100602	EventDuration.threshold (metrics section)
Incorrect load balancer routing	Indicates that a request from a specific client has come to a node that is different from the nodes to which previously served requests from that client were routed.				100401	100402	N/A
Interaction duration	Indicates that a "short-lived" media interaction was detected. This may mean there is malicious traffic that can corrupt the results of the pacing algorithms.				100501	100502	interactionMinProcessingTime (pacing section)
VisitProfile cache size	Indicates that the <b>VisitProfile</b>				100305	100306	N/A

Alarm name	Alarm description	Alarm Condition object				Related configuration option	
	object cache is full. This usually means that the server needs more memory						
IxnProfile cache size	Indicates that the <b>IxnProfile</b> object cache is full. This usually means that the server needs more memory				100303	100304	N/A
Events cache size	Indicates that the <b>Event</b> object cache is full. This usually means that the server needs more memory				100301	100302	N/A
DroolsSession cache size	Indicates that the <b>Drools</b> session object cache is full. This usually means that the server needs more memory				100201	100202	N/A
Heap Memory Usage	Defines the heap memory usage threshold value. This is the ratio of used heap memory to				10001	10002	<b>HeapMemoryUsage.threshold</b> ( <b>metrics</b> section)

Alarm name	Alarm description	Alarm Condition object					Related configuration option
	maximum heap memory.						
GC Latency	Defines the garbage collection latency threshold value, in milliseconds, in relation to the last time the garbage was collected within the configured time interval.				10005	10006	GcLatency.threshold ( <b>metrics</b> section)

## Alarm Actions

Alarm name	Detect alarm message example	Problem description	Solution	Cancel alarm message
Event Duration	[ERROR] Average event duration is too long %s	The average event processing time is too long.	<ul style="list-style-type: none"> <li>Resolve performance issues</li> <li>Correct the solution configuration</li> </ul>	[INFO] Average event duration is back to normal
Incorrect load balancer routing	[ERROR] Incorrect routing for request %s	Invalid sticky cookie in request	<ul style="list-style-type: none"> <li>Correct invalid StickyCookieFilter configuration</li> <li>Correct invalid Load Balancer configuration</li> </ul>	[INFO] Routing is back to normal
Interaction duration	[ERROR] Interaction duration %s is too short	The media interaction processing time is too short. Potential DDoS attack, due to an excessively large number of chat requests.	Resolve network security issues	[INFO] Interaction duration is back to normal
VisitProfile cache size	[ERROR] VisitProfile Cache is full	The cache is not large enough to effectively process the actual number of visit profiles.	<ul style="list-style-type: none"> <li>Increase cache size</li> <li>Resolve performance issues</li> </ul>	[INFO] Cache size is back to normal
IxnProfile cache size	[ERROR] IxnProfile Cache is full	The cache is not large enough to effectively process the actual number of engagement profiles.	<ul style="list-style-type: none"> <li>Increase cache size</li> <li>Resolve performance issues</li> </ul>	[INFO] Cache size is back to normal
Events cache size	[ERROR] Events Cache is full	The cache is not large enough to effectively process the actual number of events.	<ul style="list-style-type: none"> <li>Increase cache size</li> <li>Resolve performance issues</li> </ul>	[INFO] Cache size is back to normal

---

<b>Alarm name</b>	<b>Detect alarm message example</b>	<b>Problem description</b>	<b>Solution</b>	<b>Cancel alarm message</b>
DroolsSession cache size	[ERROR] DroolsSession Cache is full	The cache is not large enough to effectively process the actual number of drools sessions.	<ul style="list-style-type: none"><li>• Increase cache size</li><li>• Resolve performance issues</li></ul>	[INFO] Cache size is back to normal

## View Metrics with JMX

You can use [JConsole](#) to view metrics provided by your Web Engagement Server. To do this, you can start Web Engagement Server as a:

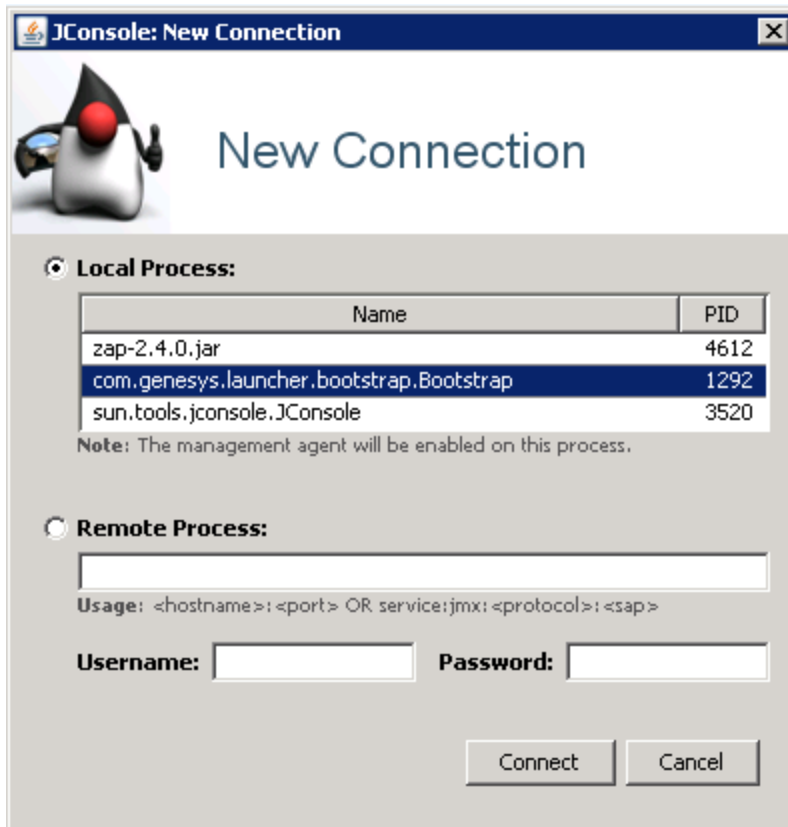
- [Local java process](#)
- [Server on a remote host](#)
- [Windows service](#)

Once you have connected, you can view your metrics in a JConsole [JMX panel](#).

You may also want to look into some of the [other tools](#) that are available for viewing your Web Engagement metrics.

---

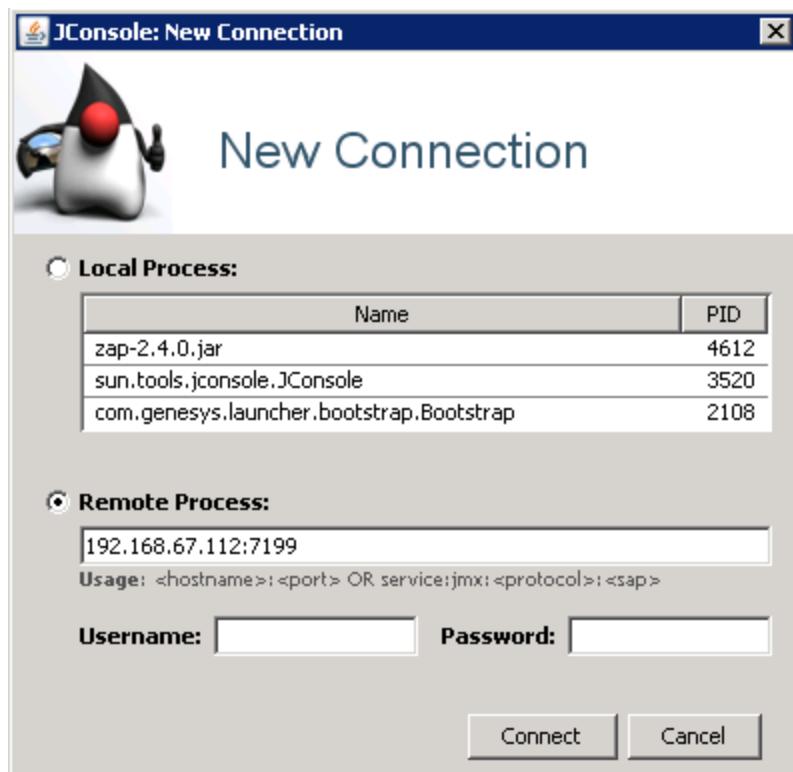
Connect to Web Engagement started as a **local java process**.



1. Run **jconsole.exe** from the **jdk/bin** directory.
2. In the **New Connection** dialog, specify the Web Engagement launcher java process.

If the Web Engagement Server was started via a BAT file in the same host where the JMX console is opened, this launcher process is the **com.genesys.launcher.bootstrap.Bootstrap** process from the **Local Process** list.

Connect to Web Engagement Server started on a **remote host**.



If the Web Engagement Server was started remotely as a server, follow these steps:

1. Run **jconsole.exe** from the **jdk/bin** directory.
2. Open **setenv.bat** and uncomment all of the lines under the line that begins:  

```
:: Uncomment for enabling JMX
```
3. Save your changes.
4. Restart the Web Engagement Server application.
5. Specify **host:JMX port** in the **Remote Process** section, as shown in the screenshot on the left.

Connect to Web Engagement started as a **Windows service**.

If Web Engagement Server is started as a Windows service, you should first stop the service, reinstall it, and restart it, as shown in these steps:

1. Stop the service.
2. Open **setenv.bat** and find the service name in the line that says **SVC\_NAME=**.



3. Run this command to remove the service:

```
server.bat -service <service name> remove
```

4. Open **setenv.bat** and uncomment all of the lines under this one:

```
:: Uncomment for enabling JMX Remote. Memorize JMX port.
```

5. Save your changes.

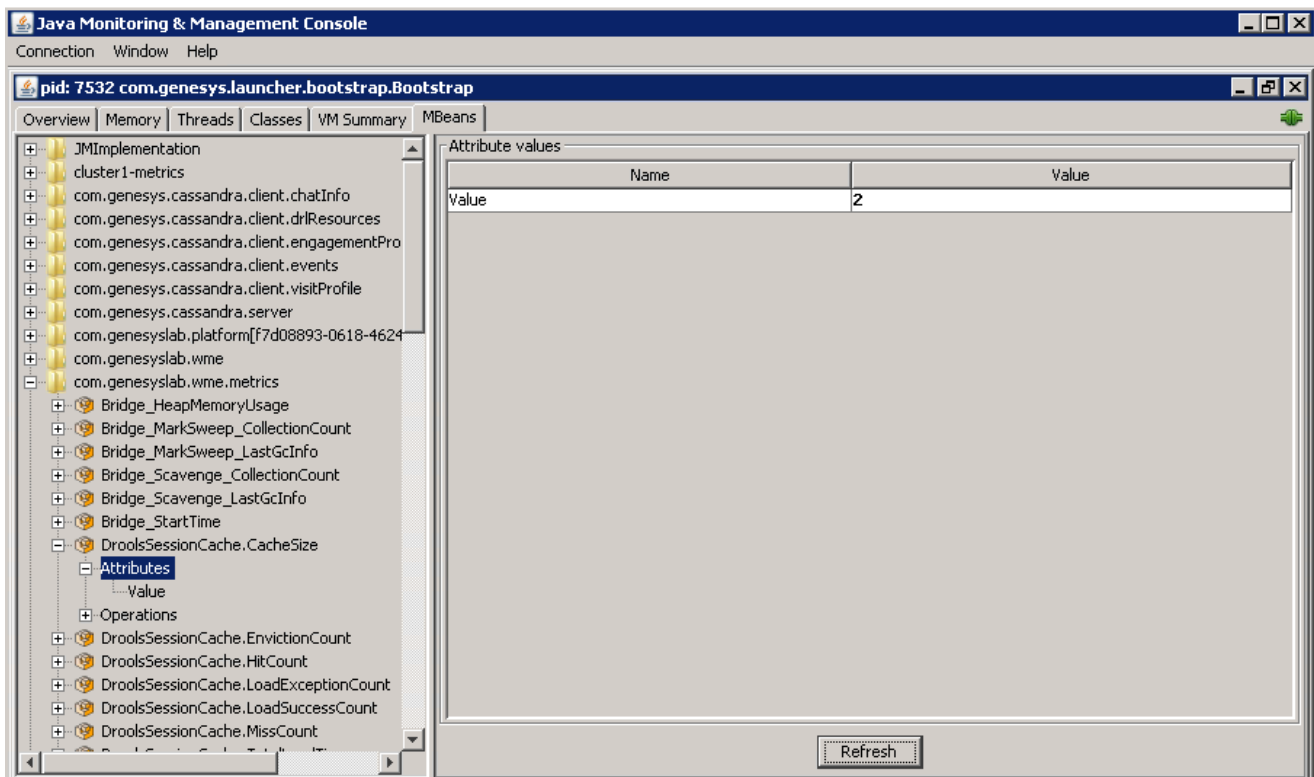
6. Run this command to install the service:

```
server.bat -service <service name> install
```

7. Start the service.

8. Specify *host:JMX port* in the **Remote Process** section, as shown in the above [screenshot](#).

Open the JMX panel to view the metrics.



1. Click **Connect** in the **New Connection** dialog. The JMX panel opens.
2. Open the **MBeans** tab and expand **com.genesyslab.wme.metrics**. All of the Web Engagement metrics are there.
3. To refresh the metrics, click **Refresh**.

## Other Tools

We have just explained how to use the JConsole tool bundled with Oracle Java (TM) to view your metrics, but there are several other tools you can use to do this:

- The EJTools JMX Browser
- Panoptes
- jManage
- MC4J
- Zabbix

# Load Balancing

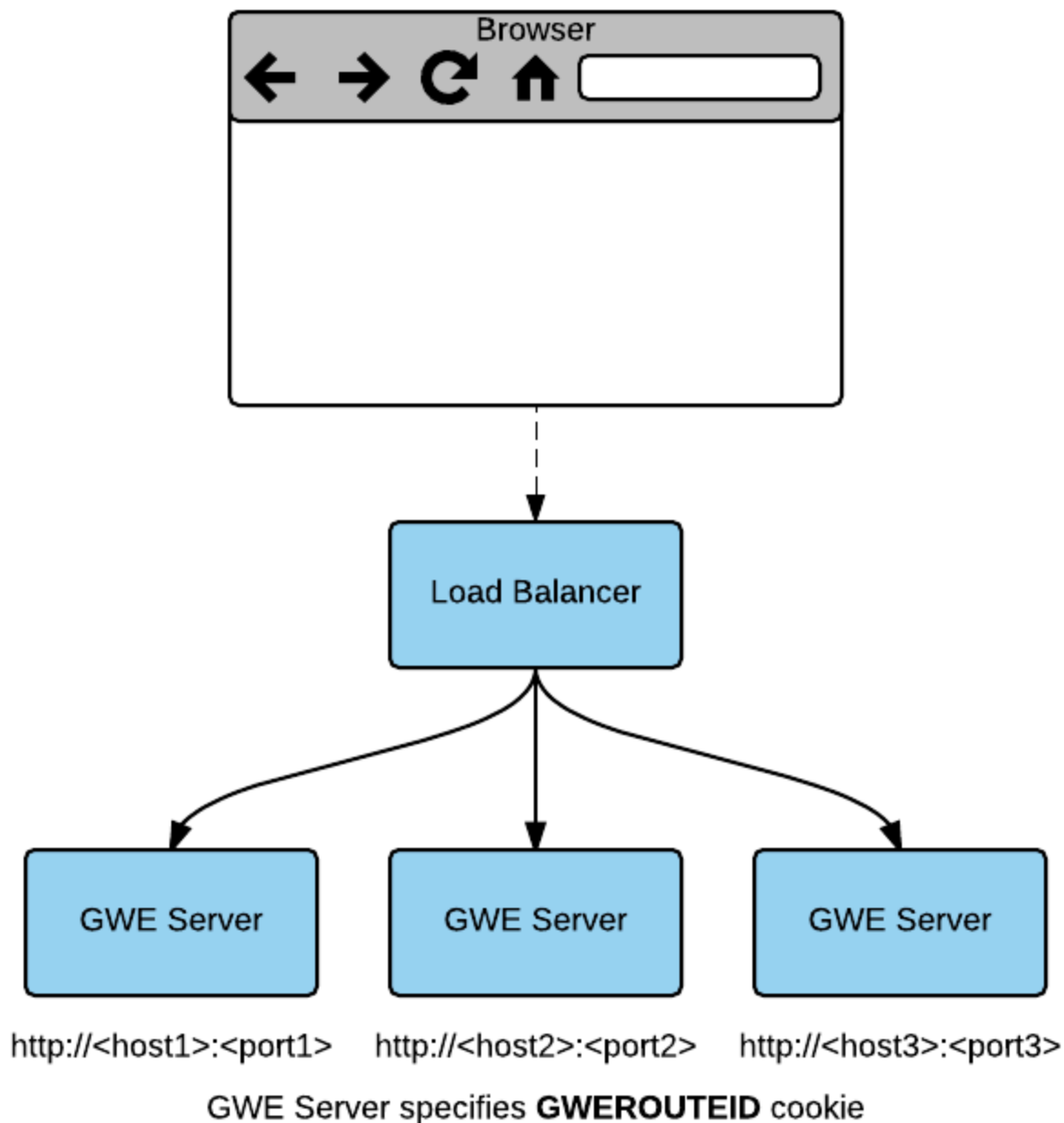
Genesys Web Engagement supports any third-party load balancer as long as the load balancing features include cookie support.

## The following points are important for you to consider when setting up load balancing:

- Due to Safari's strict cookie policy, Genesys recommends that your load balancer is hosted under the same domain as the website (or its subdomain). Otherwise, chat "stickiness" cookies might be rejected as "third-party", and the solution will not work (users won't be able to start chat).
- Apache does not support WebSockets load balancing by default. If you want to enable this option, you must use the [mod\\_proxy\\_wstunnel](#) module. **Note:** This module requires Apache version 2.4+ and is only available for Linux.
- If your load balancer does not support WebSockets, make sure that you disable WebSockets on the client side. See [Chat Application - disableWebSockets](#) and [Tracker Application - disableWebSockets](#) for details. You can also control the usage of WebSockets in CometD on the server side. See the [transports](#) option for details.

## Architecture

The following diagram shows how you can implement a load balancing configuration for your Web Engagement servers.



#### Sample of Deployment for Load Balancing

In the above example, the load balancer implements sticky sessions to GWE Servers based on the **GWROUTEID** cookie. The GWE Server is responsible for specifying this cookie.

In the Web Engagement 8.5 architecture, the load balancer can be associated with the Web Engagement cluster application. Usually, this means that the host and the default port specified in the cluster application should correspond to the host and port of the load balancer. For more information about the cluster application see the section on [Creating the Cluster Application in Installing the Genesys Web Engagement Server](#).

## Sticky Sessions

The load balancer implements sticky sessions to GWE Servers based on the **GWROUTID** cookie specified by the GWE Server instance. Therefore, it must support the following feature:

- Cookie-based stickiness to enable engagement.

### Important

The GWROUTEID cookies are created by the Genesys Web Engagement servers.

GWE requires sticky sessions not only for performance reasons, but also to switch over ongoing transactions if a node fails.

Cookie Name	Cookie Value
GWROUTEID	"." + <GWE Server application name in Genesys Administrator>

## Sample Configurations

Genesys provides sample load balancing configurations for two common load balancers: Apache and Nginx. For details, select a tab below:

### Apache

The following procedures provide a sample load balancing configuration for Apache. Before you begin, make sure you have completed the following prerequisites:

- You already deployed your Web Engagement application into a production (or production-like) environment and have at least two nodes configured to work in the cluster (see [Installing the Genesys Web Engagement Server](#) for details).
- For this configuration example, you installed and configured an instance of Apache, version 2.2.

### Configuring the Apache Load Balancer

#### Start

1. Confirm that the following modules are present in your Apache load balancer:
  - mod\_proxy.so
  - mod\_proxy\_balancer.so

- mod\_proxy\_connect.so
- mod\_proxy\_http.so
- mod\_headers.so

2. Edit the **./conf/httpd.conf** file and confirm that the modules are loaded:

- LoadModule proxy\_module modules/mod\_proxy.so
- LoadModule proxy\_module modules/mod\_proxy\_balancer.so
- LoadModule proxy\_module modules/mod\_proxy\_connect.so
- LoadModule proxy\_module modules/mod\_proxy\_http.so
- LoadModule proxy\_module modules/mod\_headers.so

3. Add the following configuration script to the end of the **httpd.conf** file:

```
<VirtualHost *:80>
ProxyRequests Off
<Proxy *>
order allow,deny
Allow from All
</Proxy>
ProxyPass /server http://<GWE_cluster_app_host_IP_or_FQDN>:<GWE_cluster_app_port>/server
ProxyPassReverse /server
http://<GWE_cluster_app_host_IP_or_FQDN>:<GWE_cluster_app_port>/server
</VirtualHost>
Listen <GWE_cluster_app_port>
<VirtualHost *:<GWE_cluster_app_port>>
<Proxy balancer://cluster>
    #BalancerMember route parameter is the same as name of GWE Server node application in
    the Genesys Configuration Layer, for example GWE_Node_1
    BalancerMember http://<GWE_node1_app_host_IP_or_FQDN>:<GWE_node1_app_port>/server
    route=GWE_Node_1
    BalancerMember http://<GWE_nodeN_app_host_IP_or_FQDN>:<GWE_nodeM_app_port>/server
    route=GWE_Node_N
ProxySet stickysession=GWEROUTEID
</Proxy>
ProxyPass /server balancer://cluster
<Location /balancer-manager>
SetHandler balancer-manager
Order Deny,Allow
Allow from all
</Location>
</VirtualHost>
```

4. Save your changes. The load balancer is now configured.

5. Your cluster is healthy if the load balancer receives a successful response on requests to

- `http(s)://Web Engagement Server Host:Web Engagement Server Port (secured port for https)/server/about`

**or**

- `http(s)://Web Engagement Server Host:Web Engagement Server Port (secured port for https)/server/isAlive`

**End**

## Nginx

The following procedures provide a sample load balancing configuration for Nginx. Before you begin, make sure you have completed the following prerequisites:

- You already deployed your Web Engagement application into a production (or production-like) environment and have at least two nodes configured to work in the cluster (see [Installing the Genesys Web Engagement Server](#) for details).
- For this configuration sample, you installed and configured an instance of Nginx.

To configure your Nginx load balancer, edit the **./conf/nginx.conf** file and modify the configuration according to the samples provided below. For details about the configuration, consult the [Nginx documentation](#).

### Configuration Sample: Nginx Load Balancer

```
events {
    worker_connections 1024;
}
http {
    include mime.types;
    default_type application/octet-stream;
    map_hash_bucket_size 64;

    log_format main 'balancing_cookie: $cookie_GWEROUTEID --> $remote_addr - $remote_user
[$time_local] "$request" ';

    access_log logs/nginx_access.log main;
    error_log logs/nginx_error.log debug;

# Select node on top of existing cookie GWEROUTEID
map $cookie_GWEROUTEID $http_sticky {
    .GWE_Node_1 192.168.1.1:9081; # GWE_Node_1 is running on 192.168.1.1:9081
    .GWE_Node_2 192.168.1.2:9081; # GWE_Node_2 is running on 192.168.1.2:9081
}

# Select node (round-robin) if cookie GQWEROUTEID is absent
upstream http_cluster {
    server 192.168.1.1:9081 fail_timeout=30s; # GWE_Node_1 is running on 192.168.1.1:9081
    server 192.168.1.2:9081 fail_timeout=30s; # GWE_Node_2 is running on 192.168.1.2:9081
}

map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}
server {
    listen <GWE_cluster_app_port>;

    location @fallback {
        proxy_pass http://http_cluster;
    }

    location /server {
        # Allow websockets, see http://nginx.org/en/docs/http/websocket.html
    }
}
```

---

```
proxy_http_version 1.1;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $connection_upgrade;
# Increase buffer sizes to find room for DOM and CSS messages
proxy_buffers 8 2m;
proxy_buffer_size 10m;
proxy_busy_buffers_size 10m;
proxy_connect_timeout 5s;
# Fall back if server responds incorrectly
error_page 502 = @fallback;
# or if doesn't respond at all.
error_page 504 = @fallback;
# Create a map of choices
# see https://gist.github.com/jrom/1760790
if ($scheme = 'http') {
    set $ftest HTTP;
}
if ($http_sticky) {
    #echo 'HTTP-STICKY scheme';
    set $ftest "${ftest}-STICKY";
}
if ($ftest = HTTP-STICKY) {
    #echo 'Pass to stickyness ';
    proxy_pass http://$http_sticky$uri?$args;
    break;
}
if ($ftest = HTTP) {
    proxy_pass http://http_cluster;
    break;
}
return 500 "Misconfiguration";
}
}
}
```



# Configuration Options

You can configure the behavior of Genesys Web Engagement by using the configuration options listed below.

**Note:** The configuration options for Web Engagement Server 8.5 fall into three categories:

- Those options which should be specified in the Web Engagement Cluster application only (cluster-only options). It is critical to guarantee that these options, such as the one that turns embedded Cassandra on and off, are the same for all cluster nodes.
- Those options which should only be specified in Web Engagement Server applications (node-only options). These options are tightly coupled with a particular instance of Web Engagement Server. For example, the listening hosts for embedded Cassandra, or the name of the log file that will be produced by a particular node.
- Those options which can be specified either in the Web Engagement Cluster or in Web Engagement Server applications. If one of these options is specified in both cluster and server applications, then option for the particular server will be used. The verbosity level is a good example of this kind of option, as it is specified in the Cluster application, but can be redefined in a particular node for troubleshooting purposes.

- [cassandraEmbedded Section](#)
- [cassandraKeyspace Section](#)
- [elasticsearch Section](#)
- [privacy Section](#)
- [cep Section](#)
- [chat Section](#)
- [cometd Section](#)
- [engagement Section](#)
- [kibana Section](#)
- [metrics Section](#)
- [pacing Section](#)
- [queues Section](#)
- [userData Section](#)
- [webcallback Section](#)
- [log Section](#)
- [security Section](#)
- [web Section](#)

---

# cassandraEmbedded Section

**Note:** The options in this section are applicable only when `enabled = true`.

## enabled (Cluster Only)

**Description:** Indicates whether the Embedded Cassandra service is enabled.

**Mandatory:** No

**Default Value:** `true`

**Valid Values:** `true`, `false`

**Changes Take Effect:** After start/restart

## clusterName (Cluster Only)

**Description:** The name of the cluster. This setting prevents nodes in one logical cluster from joining another. All nodes in a cluster must have the same value.

**Default Value:** `Cluster`

**Valid Values:** Valid string

**Mandatory:** No

**Changes Take Effect:** After start/restart

## endpointSnitch (Cluster Only)

**Description:** A snitch determines which data centers and racks nodes belong to. They inform Cassandra about the network topology so that requests are routed efficiently. They also allow Cassandra to distribute replicas by grouping machines into data centers and racks. Specifically, the replication strategy places the replicas based on the information provided by the new snitch as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/architecture/architectureSnitchesAbout\\_c.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/architecture/architectureSnitchesAbout_c.html).

**Default Value:** `GossipingPropertyFileSnitch`

**Valid Values:** `SimpleSnitch`, `GossipingPropertyFileSnitch`, `PropertyFileSnitch`, `Ec2Snitch`, `Ec2MultiRegionSnitch`, `RackInferringSnitch`

**Mandatory:** No

**Changes Take Effect:** After start/restart

## partitioner (Cluster Only)

**Description:** A partitioner determines how data is distributed across the nodes in the cluster (including replicas). Basically, a partitioner is a function for deriving a token representing a row from its partition key, typically by hashing. Each row of data is then distributed across the cluster by the value of the token, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/architecture/architecturePartitionerAbout\\_c.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/architecture/architecturePartitionerAbout_c.html).

**Default Value:** `org.apache.cassandra.dht.Murmur3Partitioner`

**Valid Values:** `org.apache.cassandra.dht.RandomPartitioner`,

org.apache.cassandra.dht.RandomPartitioner,  
org.apache.cassandra.dht.Murmur3Partitioner

**Mandatory:** No

**Changes Take Effect:** After start/restart

## dataDirectory

**Description:** The directory location where table data (SSTables) is stored. Cassandra distributes data evenly across the location, subject to the granularity of the configured compaction strategy.

**Default Value:** ./storage/data

**Valid Values:** Valid folder path

**Mandatory:** No

**Changes Take Effect:** After start/restart

## commitLogDirectory

**Description:** The directory where the commit log is stored.

**Default Value:** ./storage/commitlog

**Valid Values:** Valid folder path

**Mandatory:** No

**Changes Take Effect:** After start/restart

## savedCachesDirectory

**Description:** The directory location where table key and row caches are stored.

**Default Value:** ./storage/saved\_caches

**Valid Values:** Valid folder path

**Mandatory:** No

**Changes Take Effect:** After start/restart

## seedNodes (Cluster Only)

**Description:** A comma-delimited list of IP addresses used by gossip for bootstrapping new nodes joining a cluster. In multiple data-center clusters, the seed list should include at least one node from each data center (replication group). More than a single seed node per data center is recommended for fault tolerance. Otherwise, gossip has to communicate with another data center when bootstrapping a node. Making every node a seed node is **not** recommended because of increased maintenance and reduced gossip performance. Gossip optimization is not critical, but Genesys recommends that you use a small seed list. For more information, refer to [https://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureGossipAbout\\_c.html](https://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureGossipAbout_c.html).

**Valid Values:** A single or comma-delimited list of IP addresses

**Mandatory:** Yes

**Changes Take Effect:** After start/restart

### rpcAddress (Node Only)

**Description:** The listen address for thrift client connections.

**Valid Values:** Valid IP address or hostname of the host where the Web Engagement Server is running

**Mandatory:** Yes

**Changes Take Effect:** After start/restart

### listenAddress (Node Only)

**Description:** The IP address or hostname that Cassandra binds to for connecting to other Cassandra nodes.

**Valid Values:** Valid IP address or hostname of the host where the Web Engagement Server is running

**Mandatory:** Yes

**Changes Take Effect:** After start/restart

### rpcPort

**Description:** Thrift port for client connections.

**Default Value:** 9160

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

### nativeTransportPort

**Description:** Port on which the CQL native transport listens for clients.

**Default Value:** 9042

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

### storagePort

**Description:** The port for inter-node communication.

**Default Value:** 7000

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

### sslStoragePort

**Description:** The SSL port for encrypted communication. Not used unless enabled in the **encryption.server.internode** option.

**Default Value:** 7001

**Valid Values:** Valid port number

**Mandatory:** No

**Changes Take Effect:** After start/restart

## commitLogSync

**Description:** The method that Cassandra uses to acknowledge writes, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml\\_durability\\_c.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_durability_c.html).

**Default Value:** periodic

**Valid Values:** periodic, batch

**Mandatory:** No

**Changes Take Effect:** After start/restart

## commitLogSyncPeriod

**Description:** The period that Cassandra uses to acknowledge writes (in milliseconds).

**Default Value:** 10000

**Valid Values:** Valid integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

## numTokens

**Description:** Defines the number of tokens randomly assigned to this node on the ring when using virtual nodes (vnodes). The more tokens, relative to other nodes, the larger the proportion of data that the node stores.

**Default Value:** 256

**Valid Values:** Valid integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

## authenticator

**Description:** The authentication backend, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secure\\_about\\_native\\_authenticate\\_c.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secure_about_native_authenticate_c.html).

**Default Value:** AllowAllAuthenticator

**Valid Values:** AllowAllAuthenticator, PasswordAuthenticator

**Mandatory:** No

**Changes Take Effect:** After start/restart

## authorizer

**Description:** The authorization backend, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secure\\_about\\_native\\_authorize\\_c.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secure_about_native_authorize_c.html).

**Default Value:** AllowAllAuthorizer

**Valid Values:** AllowAllAuthorizer, CassandraAuthorizer

**Mandatory:** No

**Changes Take Effect:** After start/restart

## writeTimeout

**Description:** The time (in milliseconds) that the coordinator waits for write operations to complete.

**Default Value:** 2000

**Valid Values:** Valid long

**Mandatory:** No

**Changes Take Effect:** After start/restart

## readTimeout

**Description:** The time (in milliseconds) that the coordinator waits for read operations to complete.

**Default Value:** 5000

**Valid Values:** Valid long

**Mandatory:** No

**Changes Take Effect:** After start/restart

## configFile

**Description:** Embedded Cassandra external configuration YAML file path. Overrides all Cassandra settings.

**Valid Values:** Valid YAML file path

**Mandatory:** No

**Changes Take Effect:** After start/restart

## encryption.server.internode

**Description:** Enables or disables inter-node encryption. You must also generate keys and provide the appropriate key and trust store locations and passwords. No custom encryption options are currently enabled, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secureSSLNodeToNode\\_t.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secureSSLNodeToNode_t.html).

**Default Value:** none

**Valid Values:** none, all, dc, rack

**Mandatory:** No

**Changes Take Effect:** After start/restart

## encryption.server.keystore

**Description:** The location of a server-side Java keystore (JKS) suitable for use with Java Secure Socket Extension (JSSE), which is the Java version of the Secure Sockets Layer (SSL), and Transport Layer Security (TLS) protocols. The keystore contains the private key used to encrypt outgoing messages.

**Default Value:** conf/.keystore

**Valid Values:** Valid path

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.server.keystorePassword

**Description:** Password for the server-side keystore.

**Default Value:** cassandra

**Valid Values:** Valid string

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.server.truststore

**Description:** Location of the truststore containing the trusted certificate for authenticating remote servers.

**Default Value:** conf/.truststore

**Valid Values:** Valid path

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.server.truststorePassword

**Description:** Password for the truststore.

**Default Value:** cassandra

**Valid Values:** Valid string

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.server.clientAuth

**Description:** Enables or disables certificate authentication.

**Default Value:** false

**Valid Values:** true, false

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.client.enabled

**Description:** Enables or disables client-to-node encryption. You must also generate keys and provide the appropriate key and trust store locations and passwords. No custom encryption options are currently enabled, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secureSSLClientToNode\\_t.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secureSSLClientToNode_t.html).

**Default Value:** false

**Valid Values:** true, false

**Mandatory:** No

---

**Changes Take Effect:** After start/restart

### encryption.client.keystore

**Description:** The location of a client-side Java keystore (JKS) suitable for use with Java Secure Socket Extension (JSSE), which is the Java version of the Secure Sockets Layer (SSL), and Transport Layer Security (TLS) protocols. The keystore contains the private key used to encrypt outgoing messages.

**Default Value:** conf/.keystore

**Valid Values:** Valid path

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.client.keystorePassword

**Description:** Password for the client-side keystore. This must match the password used when generating the keystore and truststore.

**Default Value:** cassandra

**Valid Values:**

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.client.truststore

**Description:** Set if **encryption.client.clientAuth** is true.

**Default Value:** conf/.truststore

**Valid Values:** Valid path

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.client.truststorePassword

**Description:** Set if **encryption.client.clientAuth** is true.

**Default Value:** *truststore\_password*

**Valid Values:** Valid string

**Mandatory:** No

**Changes Take Effect:** After start/restart

### encryption.client.clientAuth

**Description:** Enables or disables certificate authentication.

**Default Value:** false

**Valid Values:** true, false

**Mandatory:** No

**Changes Take Effect:** After start/restart



---

# cassandraKeyspace Section

## name

**Description:** Cassandra keyspace name.

**Default Value:** none

**Valid Values:** A lowercase string that begins with an alphanumeric character and continues with a sequence of alphanumeric characters and underscores, with a maximum length of 48 characters.

**Mandatory:** Yes

**Changes Take Effect:** After start/restart

## replicationStrategy

**Description:** Replication strategy name, as described at [http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureDataDistributeReplication\\_c.html](http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureDataDistributeReplication_c.html).

**Default Value:** NetworkTopologyStrategy

**Valid Values:** SimpleStrategy, NetworkTopologyStrategy

**Mandatory:** No

**Changes Take Effect:** After start/restart

### Warning

Genesys strongly recommends **not** using SimpleStrategy. If you use SimpleStrategy, you cannot use more than one data center and KeySpaces you create will be difficult to migrate to NetworkTopologyStrategy once they contain a lot of data. Also see [https://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureDataDistributeReplication\\_c.html](https://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureDataDistributeReplication_c.html)

If you set your replication strategy to SimpleStrategy you must also:

- Configure a replication factor in the [replicationStrategyParams](#) option.
- For embedded Cassandra, set the [endpointSnitch](#) option of the [\[cassandraEmbedded\]](#) section to SimpleSnitch.
- For external Cassandra, set `endpoint_snitch: SimpleSnitch` in your `cassandra.yaml` file.

## replicationStrategyParams

**Description:** A comma-separated list of replication strategy parameters, where every list item is a colon-separated pair whose first item is the data center name enclosed in single quotes and whose second item is the number of replicas for that data center. For example, `'OperationalDC1': 3, 'OperationalDC2': 2` indicates that the OperationalDC1 data center includes 3 data replicas, while

OperationDC2 includes 2 data replicas. For more information, consult [http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureDataDistributeReplication\\_c.html](http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureDataDistributeReplication_c.html).

**Default Value:** 'OperationalDC':1

**Valid Values:** See the Cassandra replication configuration

**Mandatory:** No

**Changes Take Effect:** After start/restart

## dataCompression

**Description:** Stored data compression method.

**Default Value:** LZ4

**Valid Values:** NONE, SNAPPY, LZ4, DEFLATE

**Mandatory:** No

**Changes Take Effect:** After start/restart

## readConsistencyLevel

**Description:** Consistency level for read operations, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml\\_config\\_consistency\\_c.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_config_consistency_c.html).

**Default Value:** LOCAL\_QUORUM

**Valid Values:** EACH\_QUORUM, QUORUM, LOCAL\_QUORUM

**Mandatory:** No

**Changes Take Effect:** After start/restart

## writeConsistencyLevel

**Description:** Consistency level for write operations, as described at [http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml\\_config\\_consistency\\_c.html](http://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_config_consistency_c.html).

**Default Value:** LOCAL\_QUORUM

**Valid Values:** EACH\_QUORUM, QUORUM, LOCAL\_QUORUM

**Mandatory:** No

**Changes Take Effect:** After start/restart

## retention.time-unit

**Description:** Defines the time units for the expiration period set in the **retention.entity.all** and **retention.entity.\*** options.

**Default Value:** day

**Valid Values:** sec, min, hour, day, month

**Mandatory:** No

**Changes Take Effect:** After start/restart

## retention.entity.all

**Description:** Specifies the time to live (TTL) for all entities in the Cassandra database, in the time units set in the **retention.time-unit** options. This option affects all business entities in the database.

---

**Default Value:** 1

**Valid Values:** Valid integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

retention.entity.<object>

**Description:** Specifies the time to live (TTL) for the selected entity, in the time units set in the retention.time-unit options. <object> must be replaced with a business entity name.

**Default Value:** none

**Valid Values:** Valid integer

**Mandatory:** No

**Changes Take Effect:** After start/restart

userName

**Description:** The user's name

**Default Value:**

**Valid Values:** Valid string

**Mandatory:** No

**Changes Take Effect:** After start/restart

password

**Description:** The user's password

**Default Value:**

**Valid Values:** Valid string

**Mandatory:** No

**Changes Take Effect:** After start/restart

transportCompression

**Description:** The transport Compression method.

**Default Value:** NONE

**Valid Values:** NONE, SNAPPY, LZ4

**Mandatory:** No

**Changes Take Effect:** After start/restart

## elasticsearch Section

**Most of this section is for internal use only. Therefore, you should avoid changing the options in this section, which should be modified only by authorized Genesys representatives, except for the option listed below.**

If you need to configure other aspects of Genesys Web Engagement's indexing capabilities, please contact your Genesys representative.

### http-read-only

**Description:** Indicates whether Elasticsearch is read only. If you need to import Pulse templates, you must set this option to `false` to make Elasticsearch available for writing.

**Mandatory:** No

**Default Value:** `true`

**Valid Values:** `true`, `false`

**Changes Take Effect:** After start/restart

## privacy Section

### collectActualIPs

**Description:** If true, the IP addresses of visitors will be collected and stored in the database.

**Default Value:** false

**Valid Values:** true, false

**Mandatory:** No

**Changes Take Effect:** After server restart

### collectForwardedIPs

**Description:** If true, the IP addresses that the visitor arrived from will be collected and stored in the database.

**Default Value:** false

**Valid Values:** true, false

**Mandatory:** No

**Changes Take Effect:** After server restart

### geoMode

**Description:** The precision of the geolocation service. **Note:** This option will only work if `collectActualIPs` is enabled.

**Default Value:** LOCATION

**Valid Values:** OFF, LOCATION

**Mandatory:** No

**Changes Take Effect:** After server restart

### pathToGeoDB

**Description:** The path to the `MaxMind` GeoDB file.

**Default Value:** `./gwe/resources/geo/GeoLite2-City.mmdb`

**Valid Values:** Any valid path to a GeoDB file

**Mandatory:** No

**Changes Take Effect:** After server restart

## cep Section

Settings for Complex Event Processing (CEP).

### domainSeparation

**Description:** Specifies whether the DROOLS knowledge base should be separated by domains and sub-domains.

**Default Value:** false

**Valid Values:** true, false

**Mandatory:** No

**Changes Take Effect:** After server restart

### cepSessionCacheSize

**Description:** Defines the size of the cache that is used for storing DRL sessions. This value should be not less than expected maximum count of simultaneous visits on a particular server.

**Default Value:** 10000

**Valid Values:** positive integer greater than 100

**Mandatory:** No

**Changes Take Effect:** After server restart

### cepSessionCacheTtl

**Description:** Inactivity timeout (in seconds) on the DRL session. After the timeout value is reached, the session is treated as invalid and removed from cache.

**Default Value:** 600

**Valid Values:** positive integer greater than 300

**Mandatory:** No

**Changes Take Effect:** After server restart

## chat Section

### connectionTimeout

**Description:** Specifies the timeout, in seconds, for the connection to Chat Server. After the timeout value is reached, all sessions associated with the lost Chat Server are marked for restoration.

**Default Value:** 10

**Valid Values:** A positive integer between 1 and 10

**Mandatory:** No

**Changes Take Effect:** After server restart

### identifyCreateContact

**Description:** Specifies how a contact should be processed in the Contact Server.

**Note:** This option is applicable for proactive chat sessions only. For reactive chat sessions, you can specify the related parameter in the chat widget. See the [createContact API docs](#) for details.

**Note:** Web Engagement does not work with Contact Server directly, it only passes a parameter value, which has one of the following meanings:

- 1— Do not try to identify this contact and do not try to create it.
- 2—Try to identify this contact, but do not create it if absent.
- 3—Try to identify this contact and create it if absent.

**Default Value:** 3

**Valid Values:** 1, 2, or 3

**Mandatory:** No

**Changes Take Effect:** Immediately

### queueKey

**Description:** The key in the connected Chat Server that specifies the entry point for the Chat Routing strategy (Interaction Queue). For instance, in the Environment tenant this would be a key defined in the **endpoints:1** section of the Chat Server application.

**Valid Values:** Key defined in the **endpoints:<tenantID>** section of the Chat Server application.

**Mandatory:** Yes

**Changes Take Effect:** After server restart

### refreshPoolSize

**Description:** Specifies the number of threads used for processing requests to Chat Server.

**Default Value:** 10

**Valid Values:** Any positive integer between 1 and 99.

---

**Use the following formula to calculate the recommended value:**
$$\text{peak\_chat\_sessions\_count} / (\text{refreshPeriod} * 5)$$
**Where:**

- refreshPeriod is the value of the refreshTaskPeriod option
- peak\_chat\_sessions\_count is the maximum number of simultaneous chat sessions planned for one GWE Server

**Mandatory:** No**Changes Take Effect:** After server restart

### refreshTaskPeriod

**Description:** Specifies how often, in seconds, to refresh the chat content in the chat session.**Default Value:** 2**Valid Values:** A positive integer between 1 and 5**Mandatory:** No**Changes Take Effect:** After server restart

### requestTimeout

**Description:** Specifies the timeout, in seconds, for requests to Chat Server.**Default Value:** 5**Valid Values:** A positive integer between 1 and 10**Mandatory:** No**Changes Take Effect:** After server restart

### sessionRestorationTimeout

**Description:** Specifies the time, in seconds, during which the Web Engagement Server tries to restore a broken chat session. A session can be broken, for example, if Chat Server crashes.**Default Value:** 30**Valid Values:** A positive integer between 5 and 300**Mandatory:** No**Changes Take Effect:** After session restart

### webengagementChatQueue

**Description:** The name of the Interaction Queue that is used as the entry point to the Chat Routing strategy. This queue must correspond to the queue specified by the key set in the **queueKey** option.**Valid Values:** The name of a valid Interaction Queue**Mandatory:** Yes**Changes Take Effect:** After server restart

---



## cometd Section

This section includes options which correspond to the native CometD options described at [https://docs.cometd.org/current/reference/#\\_java\\_server\\_configuration\\_bayeux](https://docs.cometd.org/current/reference/#_java_server_configuration_bayeux).

### maxInterval

**Description:** The maximum number of seconds that the server will wait for a new long poll from a client before that client is considered invalid and is removed.

**Default Value:** 15

**Valid Values:** positive integer

**Mandatory:** No

**Changes Take Effect:** After server restart

### timeout

**Description:** The maximum number of seconds that the server will wait for a message before responding to a long poll with an empty response.

**Default Value:** 30

**Valid Values:** positive integer

**Mandatory:** No

**Changes Take Effect:** After server restart

### interval

**Description:** The maximum number of seconds that the client must wait between the end of one long poll request and the start of the next.

**Default Value:** 0

**Valid Values:** positive integer

**Mandatory:** No

**Changes Take Effect:** After server restart

### transports

**Description:** Comma-separated list of allowed CometD transports. By default, all transports are supported. This option is useful if, for example, you need to turn off the websocket transport on the server side.

**Default Value:** `org.cometd.websocket.server.JettyWebSocketTransport, org.cometd.server.transport.JSONTransport, org.cometd.server.transport.JSONPTransport`

**Valid Values:** comma-separated names of classes which implement CometD transport

**Mandatory:** No

**Changes Take Effect:** After server restart

## maxSessionsPerBrowser

**Description:** Maximum count of CometD connections opened per browser connection.

**Default Value:** 1

**Valid Values:**

**Mandatory:** No

**Changes Take Effect:** After server restart

## engagement Section

### defaultEngagementChannel

**Description:** Specifies the name of the default engagement channel. If specified and not empty, the related channel is selected for engagements by the out-of-the-box Engagement Logic strategy and the results of the pacing algorithm are ignored. Genesys recommends that you only set this option during application development.

**Valid Values:** The name of an engagement channel that is known to the Engagement Logic strategy. The valid out-of-the-box values are `proactiveChat` and `proactiveCallback`.

**Mandatory:** No

**Changes Take Effect:** Immediately

### engagementExpirationTime

**Description:** Specifies the number of seconds during which the **webengagement** interaction is considered to be valid. This time is applicable for the **webengagement** interaction from the time the interaction is created until the moment the engagement invitation is sent to the visitor.

**Default Value:** 30

**Valid Values:** positive integer from 10 to 1800

**Mandatory:** No

### registrationFormExpirationTime

**Description:** Specifies the number of seconds during which an engagement attempt is treated as alive in the state of "waiting for the form to be completed by the visitor". During this period, new engagement invitations will be blocked if the server is running in **strictEngagementMode**.

**Note:** Even after this period, the visitor is able to trigger a media (chat or webcallback) interaction. That is, the registration form by itself is not invalidated.

**Default Value:** 120

**Valid Values:** positive integer from 10 to 1800

**Mandatory:** No

**Changes Take Effect:** Immediately

### strategiesControlWebengagementIxn

**Description:** If true, the Web Engagement Server will not apply additional actions to a **webengagement** Open Media interaction, other than submitting it to Interaction Server and updating its **UserData**. In this case, the strategies are responsible for the interaction lifecycle. If false, the Web Engagement Server will be responsible for moving the interaction through the Interaction Queues and, eventually, for stopping the interaction. A value of false provides behavior compatible with 8.1.2 strategies.

**Default Value:** true

**Valid Values:** true, false

**Mandatory:** N

**Changes Take Effect:** Immediately

### strictEngagementMode

**Description:** If true, instructs the system to allow only one engagement at a time for a particular visit. If false, multiple engagement attempts will be allowed. A value of true corresponds to the behavior of GWE 8.1.2.

**Default Value:** true

**Valid Values:** true, false

**Mandatory:** No

**Changes Take Effect:** Immediately

### userIdentifier

**Description:** Specifies the name of the **UserData** key that is used by Contact Server as the key for the customer identification process.

**Valid Values:** A valid **UserData** key

**Default Value:** EmailAddress

**Mandatory:** No

**Changes Take Effect:** Immediately

### visitExpirationTime

**Description:** Specifies the number of seconds during which a visit is considered valid. The visit expiration time is checked during each request for categories. A value of 0 means that the visit never expires.

**Default Value:** 86400 (24 hours)

**Valid Values:** nonnegative integer

**Mandatory:** No

**Changes Take Effect:** Immediately

## kibana Section

### elasticsearch\_url

**Description:** Specifies the location of the Elasticsearch HTTP port. **Note:** The Elasticsearch port is not specified in the default settings for external Cassandra. Because of this, you should use the default value of 9200. If you need to use a different port for Elasticsearch, use the `-Des.http.port` option to set that value. For more information, refer to [Deploy a Cassandra Cluster Node](#).

**Mandatory:** No

**Default Value:** `http://localhost:9200`

**Valid Values:** A valid URL that points to the Elasticsearch port

**Changes Take Effect:** After start/restart

### enabled

**Description:** Indicates whether the Kibana server is started during GWE Server startup. The Kibana server is used in the Advanced Reporting Dashboard.

**Mandatory:** No

**Default Value:** `true`

**Valid Values:** `true`, `false`

**Changes Take Effect:** After start/restart

### port

**Description:** Specifies the port on which the Kibana server is listening. Used only if the Kibana server is started.

**Mandatory:** No

**Default Value:** `5601`

**Valid Values:** Any positive integer valid for specifying a port ID

**Changes Take Effect:** After start/restart

## metrics Section

### reporter.jmx.enabled

**Description:** Enables or disables the JMX reporter.

**Default Value:** true

**Valid Values:** true, false

**Changes Take Effect:** Immediately

### reporter.log.enabled

**Description:** Enables or disables metrics reporting to a file.

**Default Value:** false

**Valid Values:** true, false

**Changes Take Effect:** Immediately

### reporter.log.logFrequency

**Description:** Defines the reporting frequency for logging to a file.

**Default Value:** 30min

**Valid Values:** An expression containing a positive integer and the units being measured, such as ms, s, min, h, d. For example: 30min, 50s.

**Changes Take Effect:** Immediately

### reporter.messageServer.enabled

**Description:** Enables or disables the Message Server reporter.

**Default Value:** true

**Valid Values:** true, false

**Changes Take Effect:** Immediately

### reporter.messageServer.logFrequency

**Description:** Defines the reporting frequency for the Message Server reporter.

**Default Value:** 30min

**Valid Values:** An expression containing a positive integer and the units being measured, such as ms, s, min, h, d. For example: 30min, 50s.

**Changes Take Effect:** Immediately

### reporter.console.enabled

**Description:** Enables or disables metrics reporting to the **stdout** console.

**Default Value:** false

**Valid Values:** true, false

**Changes Take Effect:** Immediately

### reporter.console.logFrequency

**Description:** Defines the reporting frequency for logging to the **stdout** console.

**Default Value:** 30min

**Valid Values:** An expression containing a positive integer and the units being measured, such as ms, s, min, h, d. For example: 30min, 50s.

**Changes Take Effect:** Immediately

### HeapMemoryUsage.threshold

**Description:** Defines the heap memory usage threshold value. This is the ratio of used heap memory to the maximum heap memory.

**Default Value:** 0.8

**Valid Values:** A decimal fraction between 0 and 1

**Changes Take Effect:** Immediately

### GcFrequency.threshold

**Description:** Defines how many times garbage collection can occur within a given hour.

**Default Value:** 24

**Valid Values:** A positive numeric value

**Changes Take Effect:** Immediately

### GcLatency.threshold

**Description:** Defines the garbage collection latency threshold value, in milliseconds, in relation to the last time the garbage was collected within the configured time interval.

**Default Value:** 1000

**Valid Values:** The number of milliseconds

**Changes Take Effect:** Immediately

### EventDuration.threshold

**Description:** Defines the average event processing time threshold value in milliseconds.

**Default Value:** 1000

**Valid Values:** The number of milliseconds

**Changes Take Effect:** Immediately

## monitoring.event.timer.slidingWindowSize

**Description:** Defines how many of the most recent measurements will be applied when calculating a metric.

**Default Value:** 10

**Valid Values:** Any positive integer

**Changes Take Effect:** After server restart



# pacing Section

## algorithm

**Description:** Specifies which type of pacing algorithm will be used by the system.

**Valid Values:** SUPER\_PROGRESSIVE, SUPER\_PROGRESSIVE\_DUAL, PREDICTIVE\_B, PREDICTIVE\_B\_DUAL

**Mandatory:** Yes

**Changes Take Effect:** After server restart

## chatGroups

**Description:** Specifies a list of groups used by the pacing algorithm for predictions on the chat channel.

**Valid Values:** Comma-separated list of group names. Each item should be the valid name of an Agent Group in the Genesys Configuration Layer. For example: Chat Group 1,Chat Group 2.

**Mandatory:** Yes

**Changes Take Effect:** After server restart

## interactionMinProcessingTime

**Description:** Specifies the minimum number of seconds during which an interaction should be alive in order to participate in the calculation of inbound reactive traffic.

This option allows the Web Engagement server to prevent the corruption of results provided by dual pacing algorithms in the presence of low-volume malicious traffic (where visitors continually start chat interactions and then immediately close them without delivering them to an agent).

**Default Value:** 20

**Valid Values:** integer from 5 to 60

**Mandatory:** No

**Changes Take Effect:** After server restart

## optimizationGoal

**Description:** Specifies the percentage goal for the optimization target; the value you set for this option depends on the value you set for the **optimizationTarget** option.

**For example:** No more than 3% of Abandoned interactions or no less than a 75% Busy factor for the agents.

**Default Value:** 3

**Valid Values:** integer from 0 to 100

**Mandatory:** No

**Changes Take Effect:** After server restart

## optimizationTarget

**Description:** Specifies the optimization target for the pacing algorithm.

**Valid Values:** ABANDONMENT\_RATE or BUSY\_FACTOR

**Mandatory:** Yes

**Changes Take Effect:** After server restart

## proactiveRatio

**Description:** Specifies the minimum percentage of agent resources that are reserved to handle proactive interactions. If 0 is specified, no resources are specifically allocated to handle proactive interactions (note that proactive traffic is still allowed). If 100 is specified, all resources are allocated to handle proactive interactions and no reactive interactions are allowed.

**Default Value:** 100

**Valid Values:** integer from 0 to 100

**Mandatory:** No

**Changes Take Effect:** After server restart

## refreshPeriod

**Description:** Specifies, in seconds, the frequency of predictions produced by the pacing algorithm.

**Default Value:** 2

**Valid Values:** integer from 1 to 5

**Mandatory:** No

**Changes Take Effect:** After server restart

## voiceGroups

**Description:** Specifies a comma-separated list of groups used by the pacing algorithm for predictions on the **webcallback** channel.

**Note:** By default, Web Engagement 8.5 (as opposed to GWE 8.1.2) does not provision the value of the **voiceGroups** option. You must fill in this value manually if you want to use the **webcallback** channel.

**For example:** Webcallback Group 1,Webcallback Group 2.

**Valid Values:** Comma-separated list of group names. Each item should be the valid name of an Agent Group in the Genesys Configuration Layer.

**Mandatory:** No

**Changes Take Effect:** After server restart

## queues Section

### queueAccepted

**Description:** Specifies the name of the Interaction Queue where the **webengagement** interaction is placed after a visitor accepts an engagement invitation.

**Default Value:** Webengagement\_Accepted

**Valid Values:** The name of an existing Interaction Queue

**Mandatory:** Yes

**Changes Take Effect:** Immediately

### queueEngaged

**Description:** Specifies the name of the Interaction Queue where the **webengagement** interaction is placed after a positive engagement decision is made in the Engagement Logic SCXML strategy.

**Default Value:** Webengagement\_Engaged

**Valid Values:** The name of an existing Interaction Queue

**Mandatory:** Yes

**Changes Take Effect:** Immediately

### queueFailed

**Description:** Specifies the name of the Interaction Queue where the **webengagement** interaction is placed after an error in the strategy or in the Web Engagement Server.

**Default Value:** Webengagement\_Failed

**Valid Values:** The name of an existing Interaction Queue

**Mandatory:** Yes

**Changes Take Effect:** Immediately

### queueMissed

**Description:** Specifies the name of the Interaction Queue where the **webengagement** interaction is placed as the result of a negative decision in the Engagement Logic strategy.

**Default Value:** Webengagement\_Missed

**Valid Values:** The name of an existing Interaction Queue

**Mandatory:** Yes

**Changes Take Effect:** Immediately

### queueQualified

**Description:** Specifies the name of the Interaction Queue where the **webengagement** interaction is placed after a HotLead event is triggered by Complex Event Processing on the Web Engagement

Server side. This queue is the entry point to the execution of the Engagement Logic strategy.

**Default Value:** Webengagement\_Qualified

**Valid Values:** The name of an existing Interaction Queue

**Mandatory:** Yes

**Changes Take Effect:** Immediately

## queueRejected

**Description:** Specifies the name of the Interaction Queue where the **webengagement** interaction is placed after a visitor rejects an engagement invitation.

**Default Value:** Webengagement\_Rejected

**Valid Values:** The name of an existing Interaction Queue

**Mandatory:** Yes

**Changes Take Effect:** Immediately

## queueTimeout

**Description:** Specifies the name of the Interaction Queue where the **webengagement** interaction is placed after a visitor's browser sends the Timeout disposition code for the engagement invitation or after the Engagement Logic strategy is finalized by a timeout while waiting for a disposition code.

**Default Value:** Webengagement\_Timeout

**Valid Values:** The name of an existing Interaction Queue

**Mandatory:** Yes

**Changes Take Effect:** Immediately

## userData Section

### attach812StyleUserData

**Description:** If true, the Web Engagement Server will attach 8.1.2-style user data to the **webengagement** OM interaction. The following keys will be affected: **jsonEvent**, **ixnType**, **engagements\_in\_progress**, **engagement\_type** and **engagement\_attempts**. If false, only Web Engagement 8.5-style data will be added.

**Default Value:** true

**Valid Values:** true, false

**Mandatory:** No

**Changes Take Effect:** Immediately

### eventName.SignIn

**Description:** Comma-separated list of fields from the data object of a **SignIn** event that belongs to the current session. The value of these fields will be added to the User Data of the **webengagement** Open Media interaction prior to submitting it into the Interaction Queue

**Valid Values:** comma-separated list of fields expected in data object of **SignIn** event.

**Mandatory:** No

**Changes Take Effect:** Immediately

### eventName.UserInfo

**Description:** Comma-separated list of fields from the data object of a **UserInfo** event that belongs to the current session. The value of these fields will be added to the User Data of the **webengagement** Open Media interaction prior to submitting it into the Interaction Queue

**Valid Values:** comma-separated list of fields expected in data object of **UserInfo** event.

**Mandatory:** No

**Changes Take Effect:** Immediately

### eventName.VisitStarted

**Description:** Comma-separated list of fields from the data object of a **VisitStarted** event. The value of these fields will be added to the User Data of the **webengagement** Open Media interaction prior to submitting it into the Interaction Queue.

**Default Value:** timezoneOffset

**Valid Values:** comma-separated list of fields expected in data object of **VisitStarted** event.

**Mandatory:** No

**Changes Take Effect:** Immediately

## eventType.ACTIONABLE

**Description:** Comma-separated list of fields from the data object of an **ACTIONABLE** event. The value of these fields will be added to the User Data of the **webengagement** Open Media interaction prior to submitting it into the Interaction Queue.

**Default Value:** timezoneOffset

**Valid Values:** comma-separated list of fields expected from the data object of an **ACTIONABLE** event.

**Mandatory:** No

**Changes Take Effect:** Immediately

## keysToPropagate

**Description:** Comma-separated list of keys. The specified keys will be copied from the **UserData** of the **webengagement** interaction into the **UserData** of the media (**chat** or **webcallback**) interaction.

**Valid Values:** list of fields expected in **UserData** of **webengagement** interaction (either passed as part of Web Engagement events or gathered in Engagement Logic strategy)

**Mandatory:** No

**Changes Take Effect:** Immediately

## webcallback Section

### phoneNumber

**Description:** Specifies the name of the key that is used by Workspace Desktop Edition to obtain the phone number from the interaction's User Data.

**Default Value:** PhoneNumber

**Valid Values:** A valid **UserData** key

**Mandatory:** No

**Changes Take Effect:** Immediately

### webcallbackQueueSubmit

**Description:** Specifies the Interaction Queue that is used as the entry point to the **webcallback** routing strategy.

**Default Value:** New

**Valid Values:** The name of an existing Interaction Queue object

**Mandatory:** No

**Changes Take Effect:** Immediately

# log Section

## all

**Description:** Specifies the outputs to which an application sends all log events. The log output types must be separated by a comma when more than one output is configured. For example: `all = stdout, logfile`

**Default Value:** `stdout`

**Valid Values** (log output types):

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

**Changes Take Effect:** After start/restart

## standard

**Description:** Specifies the outputs to which an application sends the log events of the Standard level. The log output types must be separated by a comma when more than one output is configured. For example: `standard = stderr, network`

**Default Value:** `stdout`

**Valid Values:**

<code>stdout</code>	Log events are sent to the Standard output (stdout).
<code>stderr</code>	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 the log events in the Log Database.
memory	Log events are sent to the memory output on the local disk. This is the safest output in terms of the application performance.
[filename]	Log events are stored in a file with the specified name. If a path is not specified, the file is created in the application's working directory.

**Changes Take Effect:** Immediately

### trace

**Description:** Specifies the outputs to which an application sends the log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels). The log outputs must be separated by a comma when more than one output is configured. For example: `trace = stderr, network`

**Default Value:** stdout

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

**Changes Take Effect:** Immediately

### verbose

**Description:** Determines whether a log output is created. If it is, specifies the minimum level of log events generated. The log events levels, starting with the highest priority level, are Standard, Interaction, Trace, and Debug.

**Default Value:** standard

**Valid Values:**

all	All log events (that is, log events of the Standard, Trace, Interaction, and Debug levels) are generated.
debug	The same as all.
trace	Log events of the Trace level and higher (that is, log events of the Standard, Interaction, and Trace levels) are generated, but log events of the Debug level are not generated.
interaction	Log events of the Interaction level and higher (that is, log events of the Standard and Interaction levels) are generated, but log events of the Trace and Debug levels are not generated.
standard	Log events of the Standard level are generated, but log events of the Interaction, Trace, and Debug levels are not generated.
none	No output is produced.

**Changes Take Effect:** Immediately

## segment

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

**Default Value:** 1000

**Valid Values:**

false	No segmentation is 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 the segment to stay open. The minimum number is 1 hour.

**Changes Take Effect:** After restart

## expire

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

**Default Value:** 3

**Valid Values:**

false	No expiration; all generated segments are stored.
<number> file or <number>	Sets the maximum number of log files to store.

	Specify a number from 1—1000.
<code>&lt;number&gt; day</code>	Sets the maximum number of days before log files are deleted. Specify a number from 1—100.

**Changes Take Effect:** After restart

### Warning

If an option's value is set incorrectly — out of the range of valid values — it will be automatically reset to 10.

## affectedLoggers

**Description:** Verbosity settings are explicitly applied for the following loggers:

- Loggers that are not declared explicitly in the `log4j2.xml` configuration file.
- Loggers that are specified explicitly in the `log4j2.xml` and are specified in the value for this `affectedLoggers` option.

For other loggers specified in `log4j2.xml`, but not mentioned in the value for this option, the verbosity level is not re-applied.

Here is a use case for when you might need to set this option:

- Cassandra needs to write error messages to a log file, and at the same time, Genesys components also need to write debug messages to the log file.

To resolve this use case, you would:

1. Specify the following logger in `log4j2.xml`: `<logger name="org.apache.cassandra" level="error" additivity="false">`
2. **Do not** include `org.apache.cassandra` in the value for the `affectedLoggers` option.
3. The default `log4j2.xml` file contains the following logger: `<logger name="com.genesyslab.platform" level="info" additivity="false">`
4. Include `com.genesyslab.platform` in the value for the `affectedLoggers` option.
5. Set the `verbose` option to debug.

In the sample above, the value of `affectedLoggers` should be `com.genesyslab.platform`. Error (but no debug or info) messages from Cassandra will be available in logs, and debug messages from `com.genesyslab.platform` will be available in logs.

**Default Value:** The default value is an empty string, which means that there aren't any affected loggers.

**Valid Values:** Comma-separated list of logger names, specified in the `LOG4J2.xml`. For example: `com.genesyslab.webme.common,PROTOCOL,org.apache.cassandra`

**Changes Take Effect:** Immediately

## time\_format

**Description:** Specifies how to represent, in a log file, the time when an application generates log records. A log record's time field in the ISO 8601 format looks like this:

```
2001-07-24T04:58:10.123
```

**Default Value:** time

**Valid Values:**

time	The time string is formatted according to the HH:MM:SS.sss (hours, minutes, seconds, and milliseconds) format.
locale	The time string is formatted according to the system's locale.
ISO8601	The date in the time string is formatted according to the ISO 8601 format. Fractional seconds are given in milliseconds.

**Changes Take Effect:** Immediately

## time\_convert

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

**Default Value:** local

**Valid Values:**

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

**Changes Take Effect:** Immediately

# security Section

## auth-scheme

**Description:** Specifies the HTTP authentication scheme used to secure REST requests to the Web Engagement Server.

**Default Value:** none

**Valid Values:** none, basic

**Mandatory:** No

**Changes Take Effect:** After server restart

## password

**Description:** The password used in the authentication process for REST requests to the Web Engagement Server.

**Valid Values:** Any string

**Mandatory:** No

**Changes Take Effect:** After server restart

## user-id

**Description:** The User ID used in the authentication process for REST requests to the Web Engagement Server.

**Valid Values:** Any string

**Mandatory:** No

**Changes Take Effect:** After server restart

## provider

**Description:** Type of trusted storage. The default provider uses a trust store shipped with the current JDK distribution. It is located at **`$JAVA_HOME/jre/lib/security/cacerts`**

**Default Value:** DEFAULT

**Valid Values:** DEFAULT, JKS, MSCAPI, PKCS11, PEM

**Mandatory:** No

**Changes Take Effect:** After server restart

## trusted-ca

**Description:** Specifies the location of an X.509 certificate to be used by the application to validate remote party certificates.

**Valid Values:** A path, which can use both forward and backward slash characters.

**Mandatory:** No

---

**Changes Take Effect:** After server restart

### truststore-password

**Description:** Password for the JKS trusted storage.

**Default Value:** none

**Valid Values:** String

**Mandatory:** No

**Changes Take Effect:** After server restart

### certificate

**Description:** Specifies the location of an X.509 certificate to be used by application.

**Valid Values:** A path, which can use both forward and backward slash characters.

**Mandatory:** No

**Changes Take Effect:** After server restart

### certificate-key

**Description:** Specifies the location of a PKCS#8 private key to be used by the application in conjunction with the certificate.

**Valid Values:** A path, which can use both forward and backward slash characters.

**Mandatory:** No

**Changes Take Effect:** After server restart

### keystore-password

**Description:** Password for the JKS key storage.

**Default Value:** none

**Valid Values:** String

**Mandatory:** No

**Changes Take Effect:** After server restart

### key-entry-password

**Description:** Password for the specific key inside of key storage.

**Default Value:** none

**Valid Values:** String

**Mandatory:** No

**Changes Take Effect:** After server restart

# web Section

## staticResourcesCacheControl

**Description:** Define configuration of cache-control header for static web resources.

**Default Value:** public, max-age=1800

**Valid Values:** any values that follow the cache-control header specification, such as:

1. Deny request caching: private, no-cache, no-store, max-age=0
2. Allow request caching for 1800 seconds: public, max-age=1800

For more information, refer to: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html> and <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>.

**Mandatory:** No

**Changes Take Effect:** After server restart

## staticResourcesCacheControlPattern

**Description:** This regular expression defines the url pattern of static resources that should be affected by the behavior prescribed by the **staticResourcesCacheControl** option.

**Default pattern:** ^(http)://.\*\$

- The default pattern means "all static resources accessed through the http schema."
- The wildcard .\* means "all static resources."

**Valid Values:** any valid regular expression that describes the path to the static resource(s).

**Mandatory:** No

**Changes Take Effect:** After server restart

Option introduced in 8.5.000.43

## cors.allowedOrigins

**Description:** A comma-separated list of origins that are allowed to access the resources. The default value is \*, meaning all origins. If an allowed origin contains one or more \* characters (for example [http://\\*.domain.com](http://*.domain.com)), then \* characters are converted to .\*, . characters are escaped to \., and the resulting allowed origin is interpreted as a regular expression. Allowed origins can therefore be more complex expressions such as `https?://*.domain.[a-z]{3}`, which matches either http or https, multiple subdomains, and any 3 letter top-level domain, such as .com, .net, or .org.

**Default Value:** \*

**Valid Values:** See the explanation in the description.

**Mandatory:** No

**Changes Take Effect:** After server restart

---

## cors.allowedHeaders

**Description:** A comma-separated list of HTTP headers that are allowed to be specified when accessing the resources. If the value is a single \*, this means that any headers will be accepted.

**Default Value:** x-requested-with, content-type, accept, origin, authorization, cookie

**Valid Values:** comma-separated list of HTTP headers

**Mandatory:** No

**Changes Take Effect:** After server restart

## cors.allowedMethods

**Description:** A comma separated list of HTTP methods that are allowed to be specified when accessing the resources.

**Default Value:** GET, POST, OPTIONS, HEAD, DELETE

**Valid Values:** Comma-separated list of valid HTTP methods

**Mandatory:** No

**Changes Take Effect:** After server restart

Option introduced in 8.5.000.29

## cors.urlMapping

**Description:** Defines a filter mapping with the given URL patterns and dispatcher types for the CORS Filter.

**Default Value:** \*

**Valid Values:** Any string that contains a URL mapping parameter that follows the Servlet 3.0 mapping specification (chapter 12.2 Specification of Mappings), as documented at [http://download.oracle.com/otn-pub/jcp/servlet-3.0-fr-eval-oth-JSpec/servlet-3\\_0-final-spec.pdf?AuthParam=1442871239\\_ec11f8b00a7ffbc532fdc10df90e5c10](http://download.oracle.com/otn-pub/jcp/servlet-3.0-fr-eval-oth-JSpec/servlet-3_0-final-spec.pdf?AuthParam=1442871239_ec11f8b00a7ffbc532fdc10df90e5c10)

**Mandatory:** No

**Changes Take Effect:** After server restart

## jsonp.whiteList

**Description:** Defines a white list of regular expressions that specify the object and callback names allowed for JSONP requests.

**Default Value:** ^\_gt\.setCategory\$, ^\_gt\.setDSL\$ , ^jQuery\d\*\_d\*\$

**Valid Values:** A comma-delimited list of regular expressions, each of which specifies the name of one of the allowed callback functions. These names normally have a format of **obj.callback**.

**Mandatory:** No

**Changes Take Effect:** After server restart

Option introduced in 8.5.000.42