Deployment Guide

Genesys Web Engagement 8.1.2

12/31/2021

Table of Contents

Genesys Web Engagement Deployment Guide

Welcome to the Genesys Web Engagement 8.1.2 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.

Installing GWE in a Lab

Find procedures to install and configure Genesys Web Engagement.

Installing the GWE Plug-ins

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

Deploying GWE in Production

Find procedures to deploy and configure a cluster of Genesys Web Engagement servers.

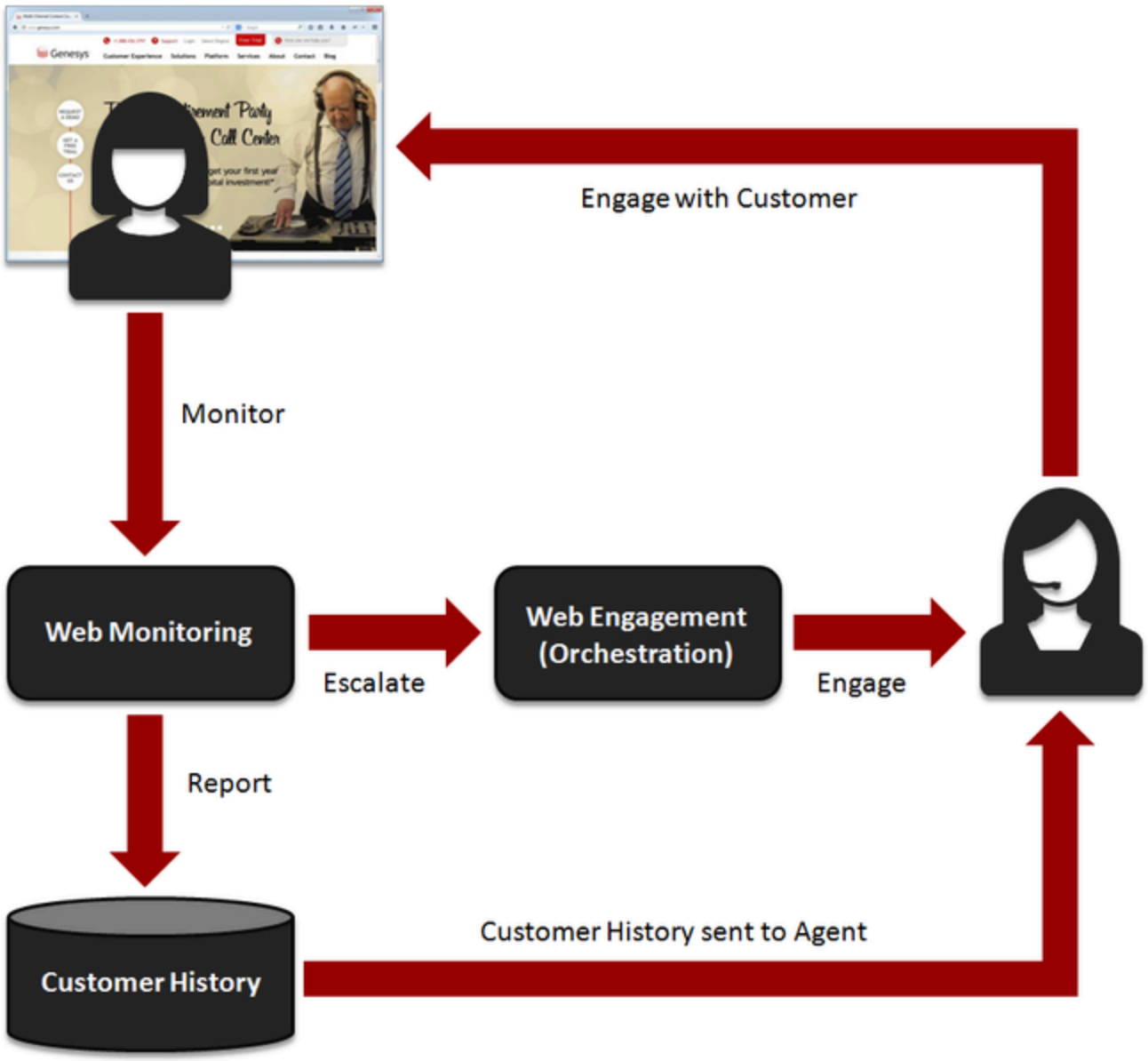
What is Genesys Web Engagement?

Overview

Genesys Web Engagement provides the ability to monitor, identify, and proactively engage web visitors in conversations that match business objectives. Customers are identified using robust business rules that provide a simple and comprehensive means for identifying key customers based on their behavior on your website and their value to your business. Key customers are then evaluated, leveraging the full power of Genesys Orchestration, and the best candidates are matched with the best agents, allowing you to better achieve your business objectives, including new customer acquisition, product sales, or customer support.

Genesys Web Engagement integrates the browsing activity of your web visitors into the overall Genesys customer service process. It records the customer web-browsing history, gathers accurate information, and converts it into Genesys interactions.

In addition to its monitoring features, Genesys Web Engagement enables you to engage the online customer by chat or web callback. In fact, you can mix and match engagement invites for available medias the way you want; you can customize the look and feel of the invites; you can support second- and third-party media; and you can even support advertisements.



From the customer standpoint, there is no visible change in the web experience.

Essentially, you can use Web Engagement to change the way you engage visitors to your website and, with the pacing algorithm, you can eliminate the need for a wait time.

Get to Know your Web Visitors

Genesys Web Engagement helps you to better understand the nature of your customers' interactions with your website. It can identify customers and their interaction history through data stored in Universal Contact Server and translate raw web activity into a form that can enable better customer service.

- **Basic Usage Information.** Get to know if a customer has ever used the web channel, and if so, how

recently (and/or how frequently).

- **Browsing History.** Each time that the customer browses your website, a session is created to store the visited pages in the customer history. This provides information on what the customer may have been looking for or may be interested in.
- **Activities and Outcomes.** Genesys Web Engagement allows you to tag web pages and define associations between URIs and outcomes to build a higher-level model of the customer browsing activity. This model is then usable to drive further interactions on other channels (chat and web callback, for now).

In addition, Genesys Web Engagement includes several scenarios for identified and unknown web visitors, detailed in Visitor Identification.

- When dealing with a user who has not been authorized, Genesys Web Engagement can ask for the user's registration. This can be customized in the Engagement Logic SCXML strategy.

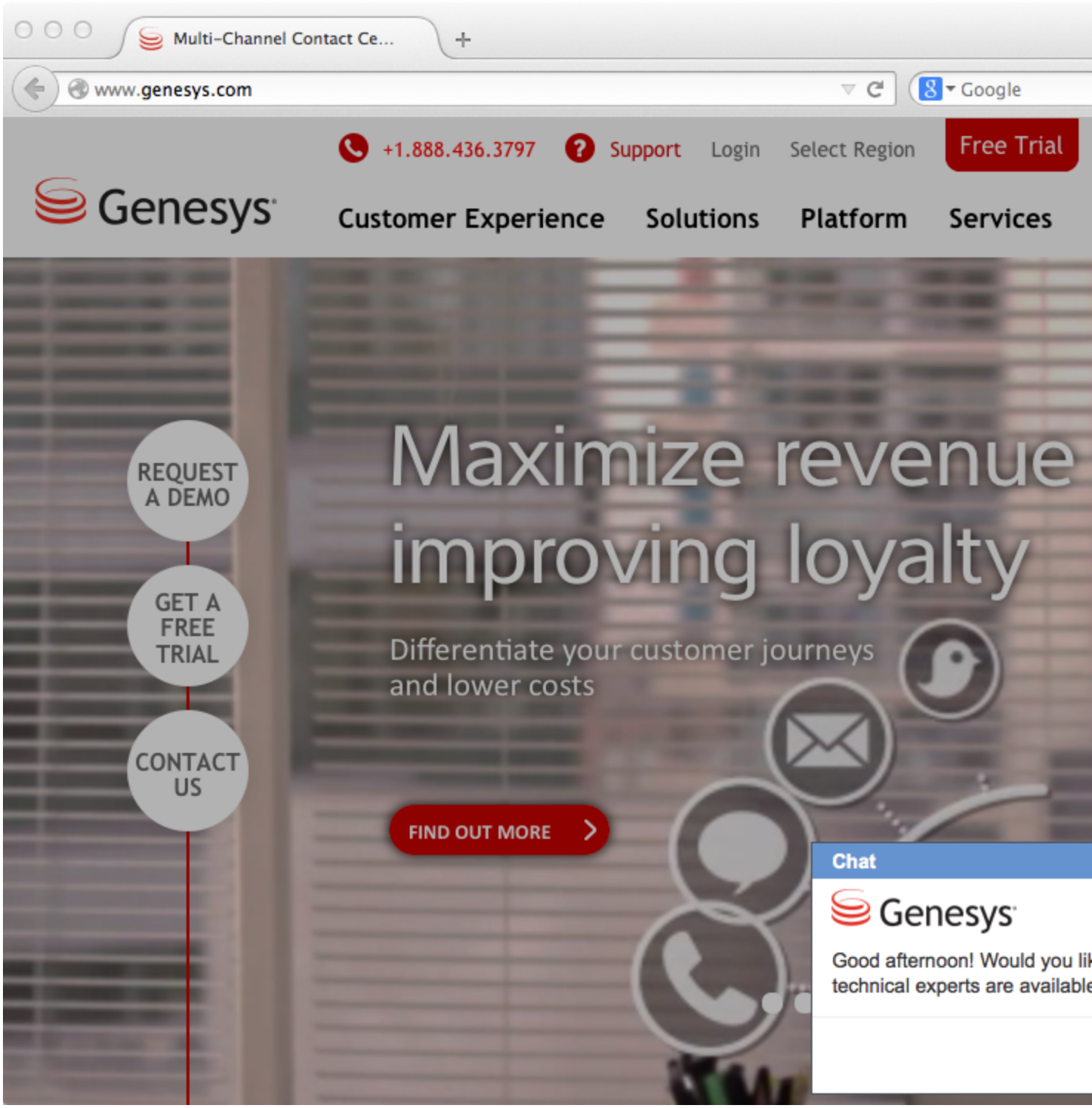
Web Follow-up

Genesys Web Engagement enables real-time or offline post processing of customers' web-browsing activity to identify potential for proactive follow-up. You can define service assistance to notify agents when some specific use cases should lead to a proactive follow-up. In addition, the flexibility of Genesys Web Engagement allows you to submit this follow-up for validation to agents, in order to make sure that the follow-up is appropriate.

For example, some use cases could be:

- When a shopping cart is abandoned, it can be caused by a lack of information. A proactive follow-up by an agent - via any number of channels - can help to close the sale.
- If a customer bought a product several weeks ago but abandoned a transaction recently, an agent could call to ask about satisfaction with the earlier purchase and afterwards follow-up with a question about the abandonment.
- If a customer submits a bad rating or comments about one of your products, an agent could follow-up by e-mail with a survey about his dissatisfaction.

In addition, if the contact center decides to make a proactive offer, the Browser-Tier Component checks that the visitor is still present, then pops up a widget in the browser window - for example, a chat invitation. If the visitor accepts the offer, the chat connection is made in the standard way using existing components.



A chat invitation.

Components

Genesys Web Engagement interfaces with the standard Genesys contact center solution and requires minimal changes to your website: to interface your website with this product, you simply add a JavaScript tracking code to your web pages. In addition, standard Genesys interfaces, such as Composer, Genesys Rules Development Tool, and Rules Authoring Tool, enable you to develop and deploy custom rules and business attributes to fine-tune your Web Engagement scenarios.

Genesys Web Engagement is composed of three components, detailed in the High-Level Architecture page:

- **Web Engagement Browser-Tier Agents** are loaded in the JavaScript tracking code, which submits system and custom events based on the customer's browsing activity.
- **Web Engagement Frontend Server** manages the event flow submitted by the browser-tier agents and is responsible for the complex event processing and submitting actionable events.
- **Web Engagement Backend Server** processes actionable events that come from the Web Engagement Frontend Server and pushes them for processing in the SCXML strategy. It also serves as the entry point for notifications from the SCXML strategy and sends notifications to the Web Engagement Frontend Server.

In addition to rules templates, Genesys Web Engagement include the following plug-ins:

- **Web Engagement Plug-in for Administrator Extension**, implementing:
  - **Script Generator**, to generate the standard JavaScript tracking code that you must add to your web pages.
  - **Categories**, to create custom business data.
- **Web Engagement Plug-in for Interaction Workspace**, to get all the web-based contexts routed to agents. This plug-in is mandatory to enable chat and web callback engagement features in Interaction Workspace.
- **Web Engagement Plug-in for Workspace Desktop Edition**, to get all the web-based contexts routed to agents. This plug-in is mandatory to enable chat and web callback engagement features in Workspace Desktop Edition 8.5.

Related Genesys Components

Genesys Web Engagement interacts with the following Genesys products:

- Interaction Workspace — Use the Genesys Web Engagement Plug-in for Interaction Workspace to interface Genesys Interaction Workspace with Web Engagement. This plug-in enables you to get all the web-based contexts routed to agents. This plug-in is mandatory to enable chat and voice engagement.
- Orchestration Server — Provides routing for chat and voice interactions.
- Stat Server — Provides statistics used by the reporting templates.
- Interaction Server — Processes interactions.
- Chat Server — Enables chat functionality in GWE.
- Composer — Use this application to publish the Genesys Rules System rules templates for GWE and to update or deploy routing and engagement strategies.

- Genesys Rules Development Tool — Deploy this as a Composer plug-in and use it to publish rules templates for GWE.
- Genesys Rules Authoring Tool — Use this web application to create rules based on the GWE rules templates. These rules are used to determine whether or not to generate an actionable event.
- Genesys Administrator Extension — Create categories that are used by the GWE rules to determine whether or not to generate an actionable event.

Features

Genesys Web Engagement includes the following features:

- Can be integrated with Genesys Co-browse and Chat
- Can be integrated with second- and third-party media, such as Genesys Mobile Services.
- Integrated Reactive Chat Application
  - Support for Chat Application customization through API
  - Embedded or pop-up mode
- Browser support for Firefox, Chrome, Safari, and Internet Explorer.
- Mobile browser support for iOS Safari and Android Chrome.
- Integrated Web Chat and Web Callback
  - Optimization / Pacing of Web Engagement Invitations
  - Included Web Engagement applications for proactive Genesys Chat and Genesys Web Callback
  - Embedded or pop-up mode for proactive chat
- Behavior Rules Authoring for simplified tooling of Web Pages
  - Categorization - Key word and regular expressions for out-of-the box web page identification
  - Out-of-the-box business events for capturing searches and timeout as part of behavior rules
  - Out-of-the-box rule templates and business rules interface for defining engagement rules based on customer behavior
  - Support for advanced business events to capture events not covered by categories
  - Business User friendly UIs for creating both Categories and Business rules
- Monitoring and data storage of customer web activity
  - Storage of web history for authenticated and anonymous customers
  - RESTful API for full access to web history, including history of current session for anonymous customers and history of current and previous sessions for authorized customers.
- Integrated Agent Interface for Interaction Workspace
  - Out-of-the box Agent Desktop support
  - Live monitoring of customer during engagement
  - Integrated view of Web History

- Reporting
  Templates for out-of-the-box real-time interaction reporting of Web Engagement
- Integrated with core Genesys product suite
- Web Engagement sample applications:
  "playground" - demonstrates the full set of Genesys Web Engagement features
- Developer tools:
  Basic and advanced proxy tools which enable you to inject the instrumentation script "on the fly" and develop your Web Engagement application without touching your website directly.

  Instrumentation tool (InTools), a Chromium-based tool that makes it easier to create DSL for your site.

Multi-Tenancy

To implement Genesys Web Engagement, you should create one Genesys Web Engagement application per tenant. This application supports multiple domains and all their associated subdomains. For instance, the **genesys** application supports the **genesys.com** domain and its associated subdomain, **docs.genesys.com**.

> **Important**
>
> When a visitor crosses domain boundaries, Web Engagement creates a new visit and closes the previous visit.

As a result, your deployment must respect the following constraints:

- One tenant can contain multiple DSL files, but instrumentation will upload only one of them at a given time; a single DSL file is active for a given page during the customer's visit on your website.
- The load balancer for the Frontend Server cluster (or the Frontend Server) is the single entry point.

Deployment Scenarios

Genesys Web Engagement has two flavors of deployment: Standalone is appropriate for a lab environment, while Clustering is geared towards a production environment. 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 follow the Standalone deployment scenario to install Genesys Web Engagement in a lab environment where you can test your application.

Standalone

Overview

This deployment is appropriate for a lab environment and consists of a single Frontend Server and a single Backend Server — 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 Backend and Frontend servers.
3. Configure Genesys Rules System. You need to configure Genesys Rules Authoring Tool and Genesys Rules Development Tool to work with GWE.
4. Configure the generic Cassandra settings. Set the Java heap size and configure the embedded Cassandra instance in the Backend Server.

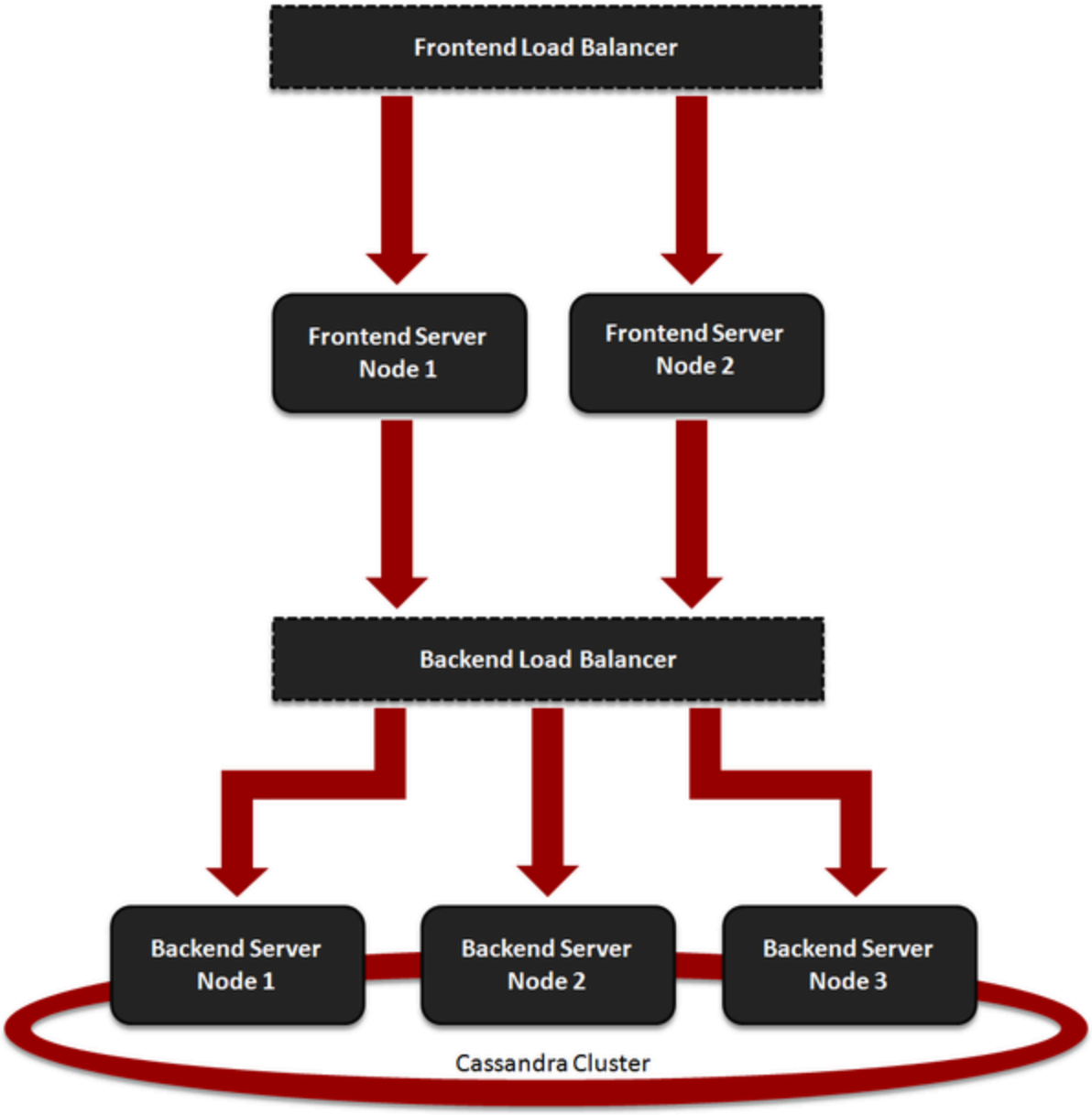
5. Install the Plug-in for Interaction Workspace. You will need this plug-in to enable chat and web callback engagement features in Interaction Workspace.
6. Install the Plug-in for Genesys Administrator Extension. You will need this plug-in to create categories for your application and enable GWE on your website.
7. Configure your required security. You can enable SSL, configure TLS for Genesys and GWE servers, and enable authentication for the Backend Server, Interaction Workspace, and the Engagement Strategy.
8. Configure your required features. You can set up the pacing algorithm, enable authentication and SSL, configure UTF-8, and more.
9. Develop an application. You're ready to move on to the Developer's Guide where you will learn how to develop and customize an application, not to mention create categories, business events, and rules.

Clustering

Overview

This deployment scenario is appropriate for a production environment and consists of multiple Frontend and Backend servers. 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

You should plan to host your Backend Servers in a secure zone, along with your Chat Server(s), in order to protect the data. The Backend 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

The Deploying and Configuring the Genesys Web Engagement Cluster page includes all the procedures required to configure and deploy your GWE cluster:

1. Review the Prerequisites. Make sure you review these before you get started.
2. Update the Backend Server Application. In this procedure you update the Backend Server Application object in Genesys Administrator and run the Provisioning Tool.
3. Configure the Frontend Server Nodes. Configure each of your Frontend Server nodes to work with load balancing and copy the server directory to the node host.
4. Configure the Backend Server nodes. Configure each of your Backend Server nodes to work with load balancing and copy the server directory to the node host.
5. Configure Load Balancing. This guide includes sample configurations for Apache, although your settings may be different depending on the architecture of your cluster.
6. Configure Cassandra for the Cluster. Make sure your Cassandra nodes (Backend Server nodes) are configured for a cluster.
7. Enable SSL for the Cluster. You need to use a certificate issued by a third-party Certificate Authority for a production environment.
8. Configure Rules Deployment for the Cluster. Set up one of your Backend Servers to be in charge of rules deployment over the cluster.
9. Start the Server Clusters. Start your load balancers, Backend Server cluster, and Frontend Server cluster.

Prerequisites

Hardware Requirements

See Sizing for details.

OS Requirements

Genesys Supported Operating Environment Reference Guide

Browser Support

Web Browsers

- Google Chrome
- Mozilla Firefox 9.0+
- Microsoft Internet Explorer 8+
- Apple Safari 5.1+

Mobile Browsers

- iOS Safari
- Android Chrome

Known Limitations

The Playground application included in the Genesys Web Engagement solution only supports Internet Explorer 9.0+.

Java Requirements

Java version 1.6.0_38 or higher.

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	8.1.300.39
Stat Server	8.1.000.23
Interaction Server	8.5.100.18
Chat Server	8.1.001.41
Genesys Rules Authoring Server	8.5.000.11
Configuration Server (for UTF-8 support)	8.1.200.04
Configuration Server DB (for UTF-8 support)	8.1.200.04

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	8.5.000.11	You must create a user with Roles Privileges that enable the creation of Rules package in Genesys Rules Authoring. See Role-Based Access Control in the Genesys Rules System Deployment Guide.
Genesys Rules Development Tool	8.1.400.02	This component must be deployed as Composer plug-in or independently in Eclipse; make sure that settings are correct for Configuration Server and Repository Server, as detailed in Installing the GRDT Component in the Genesys Rules System Deployment Guide.
Composer	8.1.300.89	Mandatory to publish the Web Engagement rules template. This component can also be used to update or deploy routing and engagement strategies.
Genesys Administrator Extension	8.1.400.58	Mandatory for the Simple Engagement model.
Genesys Administrator	8.1.300.02	Mandatory.

Components for Interaction Management

Component Name	Minimum Compliant Version
Interaction Workspace (If you use the GWE plug-in for Interaction Workspace)	8.1.401.36
Workspace Desktop Edition (If you use the GWE plug-in for Workspace Desktop Edition)	8.5.000.55

Additional Components (Optional)

Component Name	Minimum Compliant Version
CCPulse+	8.0.000.36

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 spreadsheet 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

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 the Backend 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 (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)

You 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. So, to estimate the possible peak load, you can 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 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

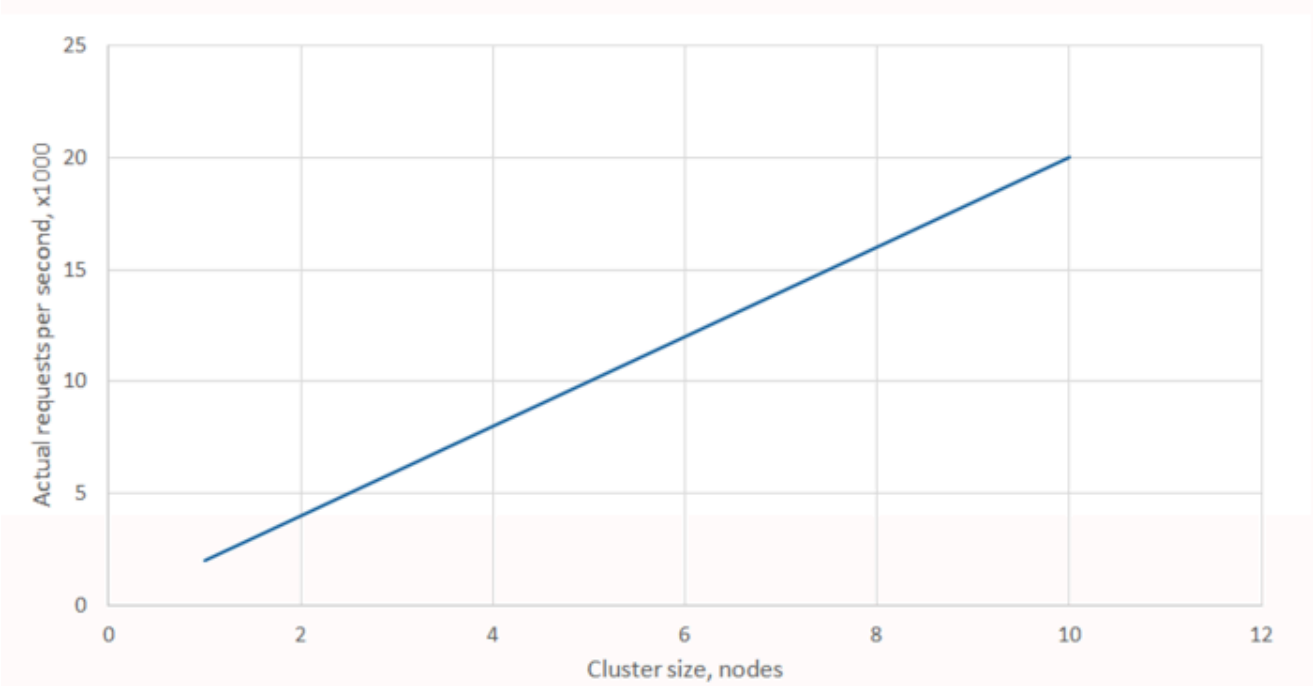
The solution deployed should handle all user input, and have some N+1 redundancy, so the minimum solution size is:

- Web Engagement Server – two nodes (to allow redundancy and load balancing)
- Cassandra – three nodes (keep data consistent and allow one node to fail).

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

Recommended and Peak Solution Size

Given the estimated average and maximum requests rates, you can find the possible solution size from the Web Engagement Server scaling charts (used in the **GWE Sizing Calculator**).



Disk Space Usage

Disk space usage directly depends on the event rate in the solution.

Using the previously defined solution configuration:

- Events for each visit: VisitStarted
- Events for each page: PageEntered + PageExited + 2xBusinessEvent
- A single category is applying for approximately 50% of pages views

And assuming that the Backend cluster has 4 nodes and replication factor = 3:

Average page views per hour, x 1000	Events per hour, x 1000	Total database size, GB				
		Hour	Day	Week	Month	Year
5	20.5	0.15	3.7	25.9	112.6	1 351.1
10	41	0.31	7.4	51.82	225.2	2 702.3
500	2 060	15.5	371	2 595	11 276	135 309
1 000	4 000	30.9	742	5 190	22 550	270 620

Notes:

- Database sizes shown without effect of Commit Log size (defaults to 4Gb for each node).
- Disk space required for each node: (Total Size)/4 nodes.

Glossary

Visit – series of page views from the same uniquely identified individual (a visitor).

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 service, including WebEngagement requests:

- Loading of monitoring script
- Loading of categorization information
- Sending the event information 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 a Cassandra cluster:

- For help understanding the Cassandra architecture, see http://www.datastax.com/documentation/cassandra/1.2/cassandra/architecture/architectureTOC.html
- For information about hardware considerations for Cassandra nodes, see http://www.datastax.com/documentation/cassandra/1.2/cassandra/architecture/architecturePlanningHardware_c.html
- For details about Cassandra cluster configuration, refer to http://www.datastax.com/documentation/cassandra/1.2/cassandra/initialize/initializeSingleDS.html
- For more about Cassandra clusters and memory, see http://www.datastax.com/documentation/

cassandra/1.2/cassandra/operations/ops_tune_jvm_c.html

- For more help calculating the number of Cassandra nodes (Backend servers) you need to support data consistency in the cluster, see http://www.ecyrd.com/cassandracalculator/.

Installing the Genesys Web Engagement Servers

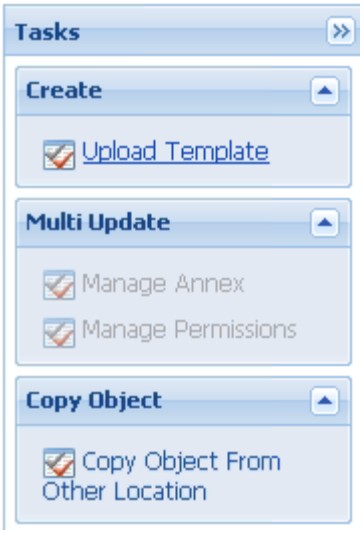
Complete the procedures on this page if you are following the Genesys Web Engagement Standalone deployment scenario, which is appropriate for a lab environment.

1. Importing the Backend Server Application Template
2. Creating the Backend Server Application
3. Configuring the Backend Server Application
4. Configuring a Connection to a Cluster of Chat Servers (Optional)
5. Installing the Backend Server
6. Importing the Frontend Server Application Template
7. Creating the Frontend Server Application
8. Configuring the Frontend Server Application
9. Installing the Frontend Server

Importing the Backend Server Application Template

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_Backend.apd** file or, if your Configuration Server does not support Web Engagement specific types, select **Web_Engagement_Backend_Generic.apd**, available in the **templates** directory of your installation CD. The **New Application Template** panel opens.
5. Click **Save & Close**.

End

Back to top

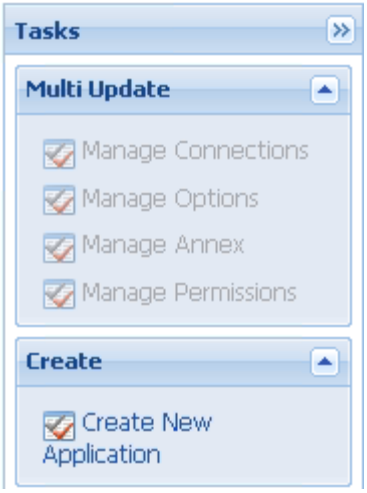
Creating the Backend Server Application

Prerequisites

- You completed Importing the Backend Server Application Template.

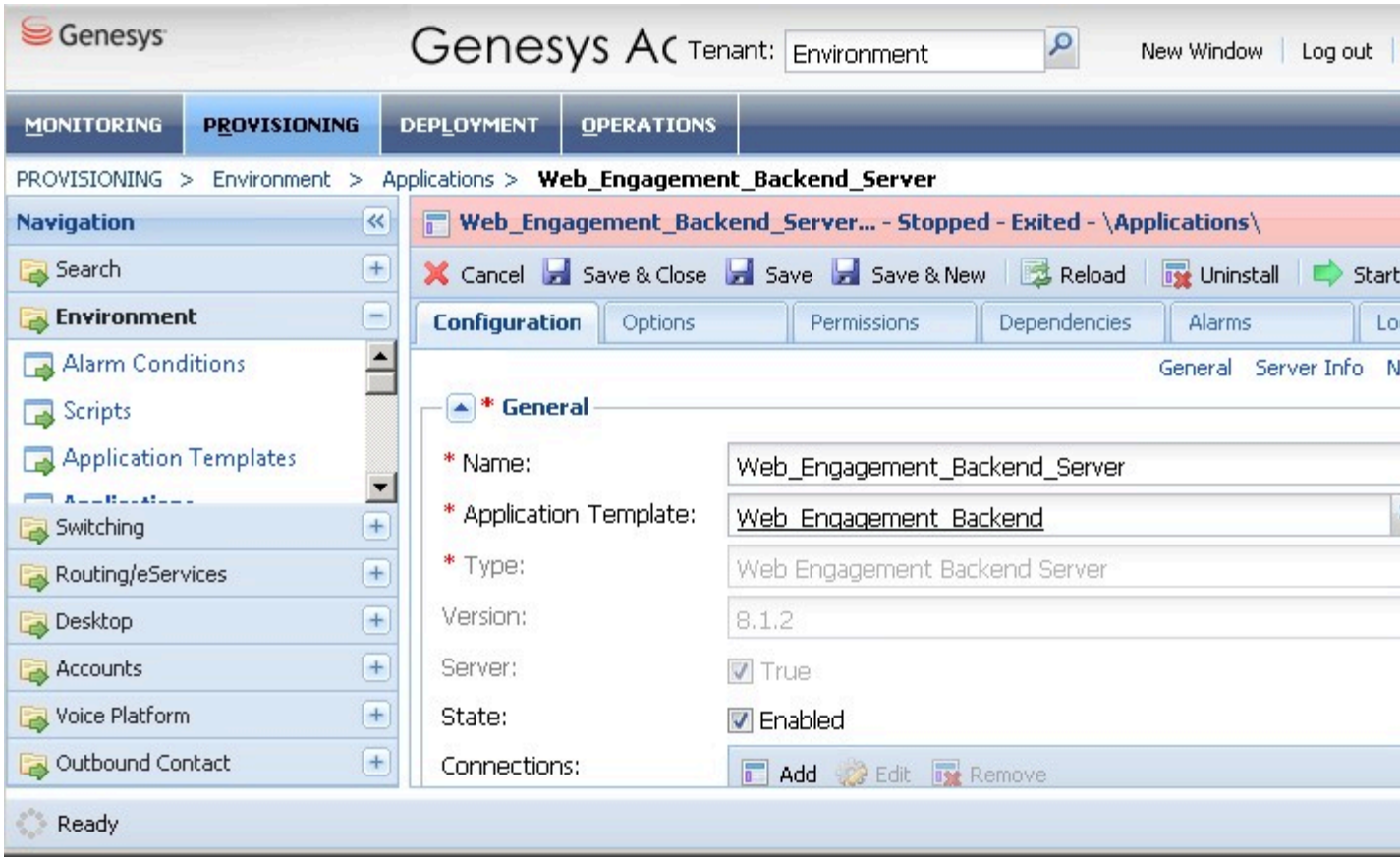
Start

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 Backend Server template that you imported in Importing the Backend Server Application 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 **Web_Engagement_Backend.xml** file or select the **Web_Engagement_Backend_Generic.xml** file if you chose **Web_Engagement_Backend_Generic.apd** in Step 4 of Importing the Backend Server Application 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, `Web_Engagement_Backend_Server`.
   - Enable the **State**.
   - Select the **Host** on which the Backend Server will reside.
   - Click **Create**.
8. The **Results** panel opens.
9. Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The Backend Server application form opens and you can start configuring the Backend Server application.



Backend Server application opened in Genesys Administrator.

End

Back to top

Configuring the Backend Server Application

Prerequisites

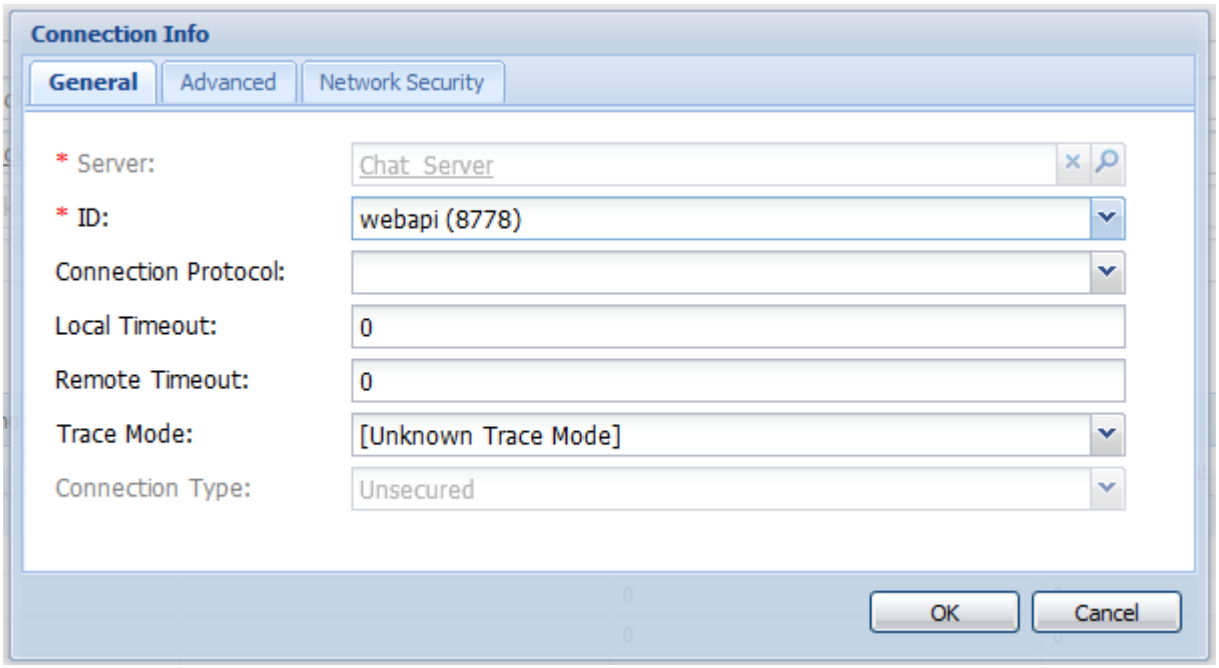
- You completed Creating the Backend Server Application.

Start

1. If your Backend Server application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the Web Engagement Backend Server and click **Edit...**.
2. 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**.
3. Repeat the previous step for Stat Server and Chat Server. Optionally, you can also add a connection to Orchestration Server or Message Server (to apply the **network** logging option).
   Note: You must add a connection to the Frontend Server after you complete Configuring the

Frontend Server Application.

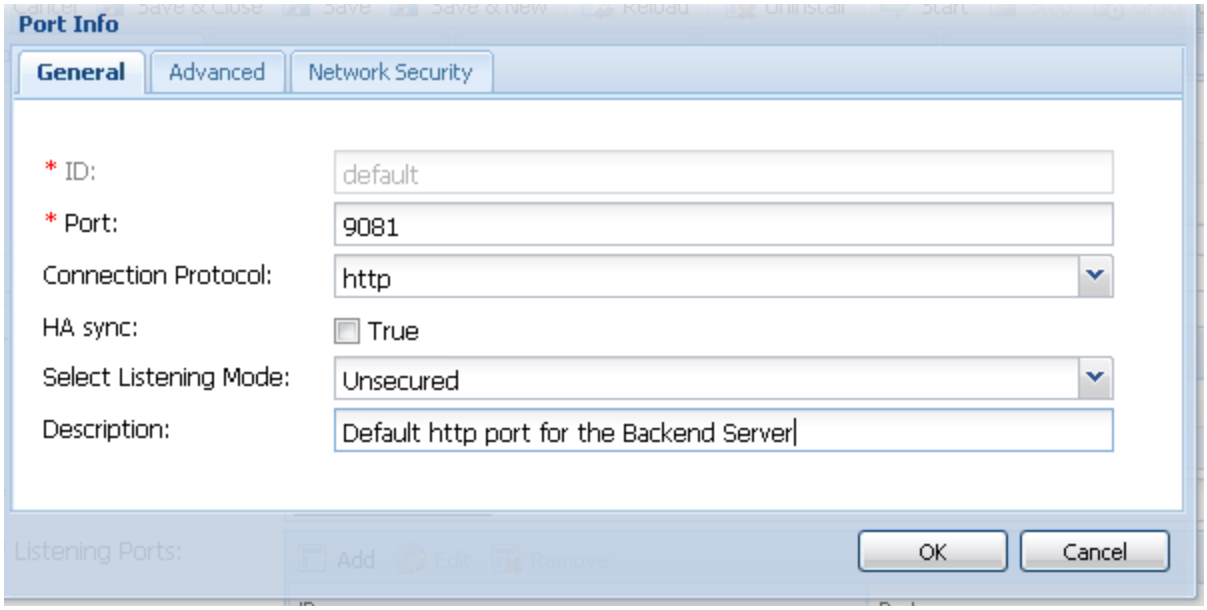
4. Select the Chat Server row in the list of connections and click `Edit`. In the **Connection Info** window, select the `webapi` port for the **ID**. Click OK.



You must set the **ID** to the `webapi` port.

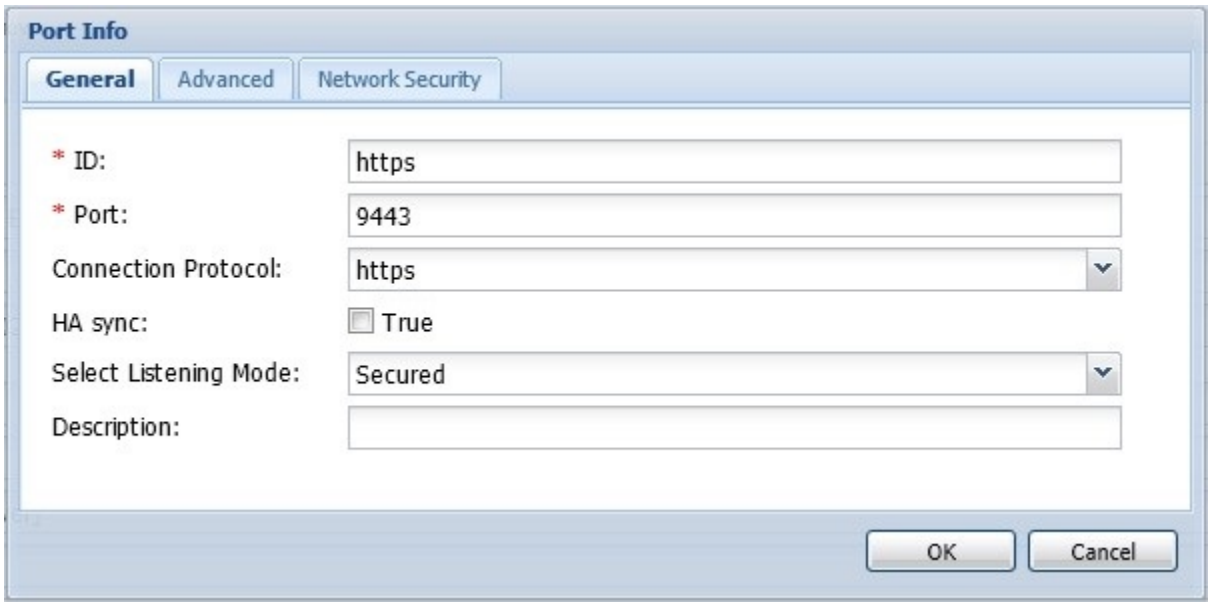
Note: Genesys Web Engagement also supports Chat Server clusters. To configure the Web Engagement Backend Server to work with a cluster of Chat Server objects, see Configuring a Connection to a Cluster of Chat Servers (Optional).

5. Expand the **Server Info** pane.
6. In the Tenant section, click **Add** and select your tenant. For instance, `Environment`. Click **OK**.
   Note: The Frontend Server and Backend Server applications should belong to the same Tenant. See Multi-Tenancy for details.
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.
   - 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**.



Default HTP port

9. 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 8.1.2.
   - Enter the **Port**. For instance, 9443.
   - Enter `https` for the **Connection Protocol**.
   - Choose `Secured` for the **Listening Mode**.
   - Click **OK**.



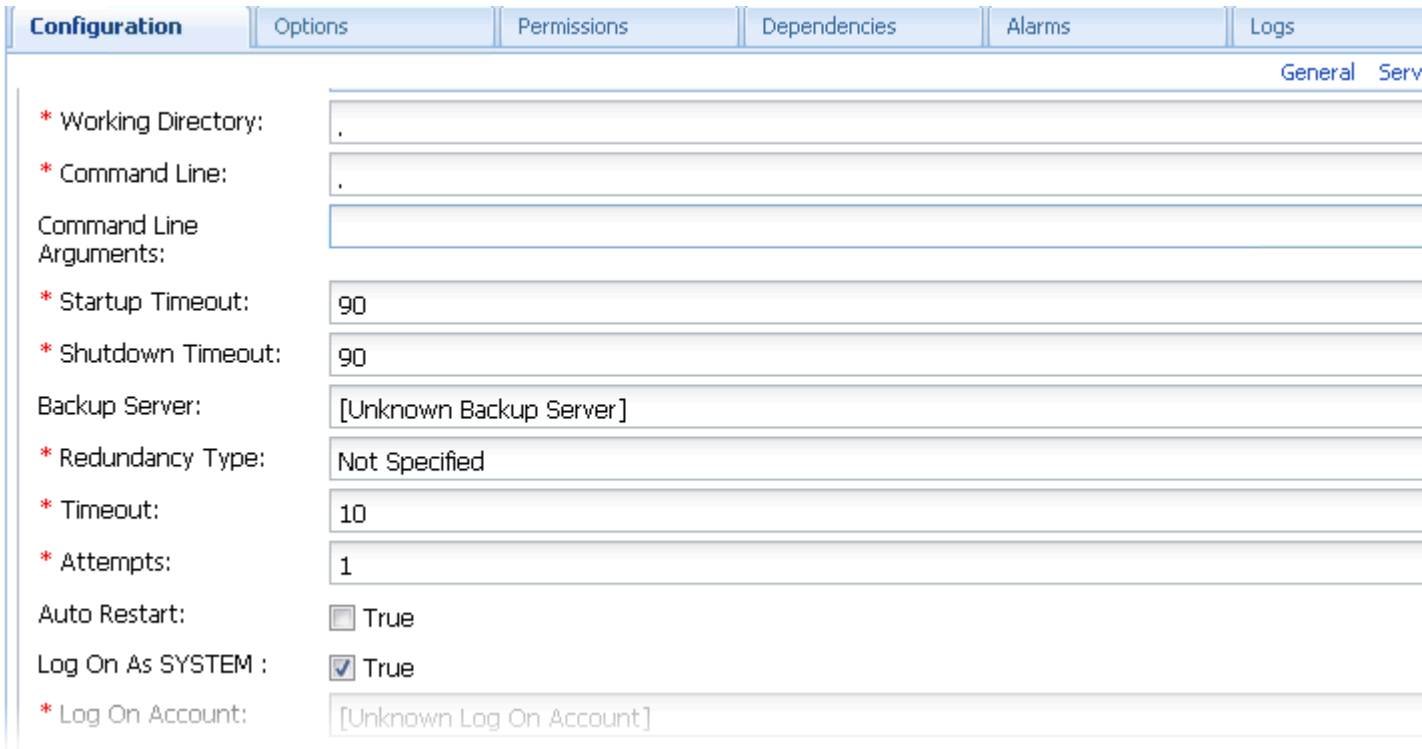
Secure listening port

10. Create the port with the `backend/data/rules/deploy` identifier by clicking **Add**.
    - Enter `backend/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 `backend/data/rules/deploy` ID appears in the list of Listening ports.
11. Optionally, you can explicitly add a Jetty stop port for the Backend Server. If you do not define a stop port, it is auto-generated when you start the server. To specify the stop port, click the `Add` button. The `Port Info` dialog opens.
    - Enter `stop` for the **ID** field. This specific ID is required.
    - Enter the **Port**. For instance, `18081`.
    - Click **OK**. The `stop` port appears in the list of Listening ports.

ID	Port
backend/data/rules/deploy	9081
default	9081
https	9443
stop	19081

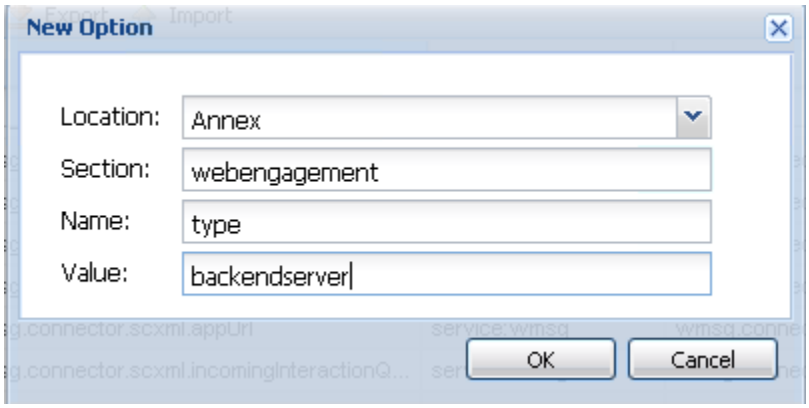
Listening ports

12. Ensure the **Working Directory** and **Command Line** fields contain "." (period). They are automatically populated when the Backend Server is installed.



Commands

13. Click **Save**.
14. The **Confirmation** dialog for changing the application's port opens. Click **Yes**.
15. (Optional) Select the **Options** tab. 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\Backend_Server` to force the system to write logs in the console and in a file.
16. (Optional) You need to create a **webengagement** annex only if your Web Engagement Backend Server application's type is Genesys Generic Server. Click **New** in the **Options** tab. The **New Options** dialog opens.
    - Select `Annex` for **Location**.
    - Enter `webengagement` for **Section**.
    - Enter `type` for **Name**.
    - Enter `backendserver` for **Value**.

    Backend type.
    - Click **OK** to create the new annex. You can see it by selecting `Advanced View (Annex)` in the **View** selector.
17. Click **Save & Close**. If the **Confirmation** dialog opens, click **Yes**.

End

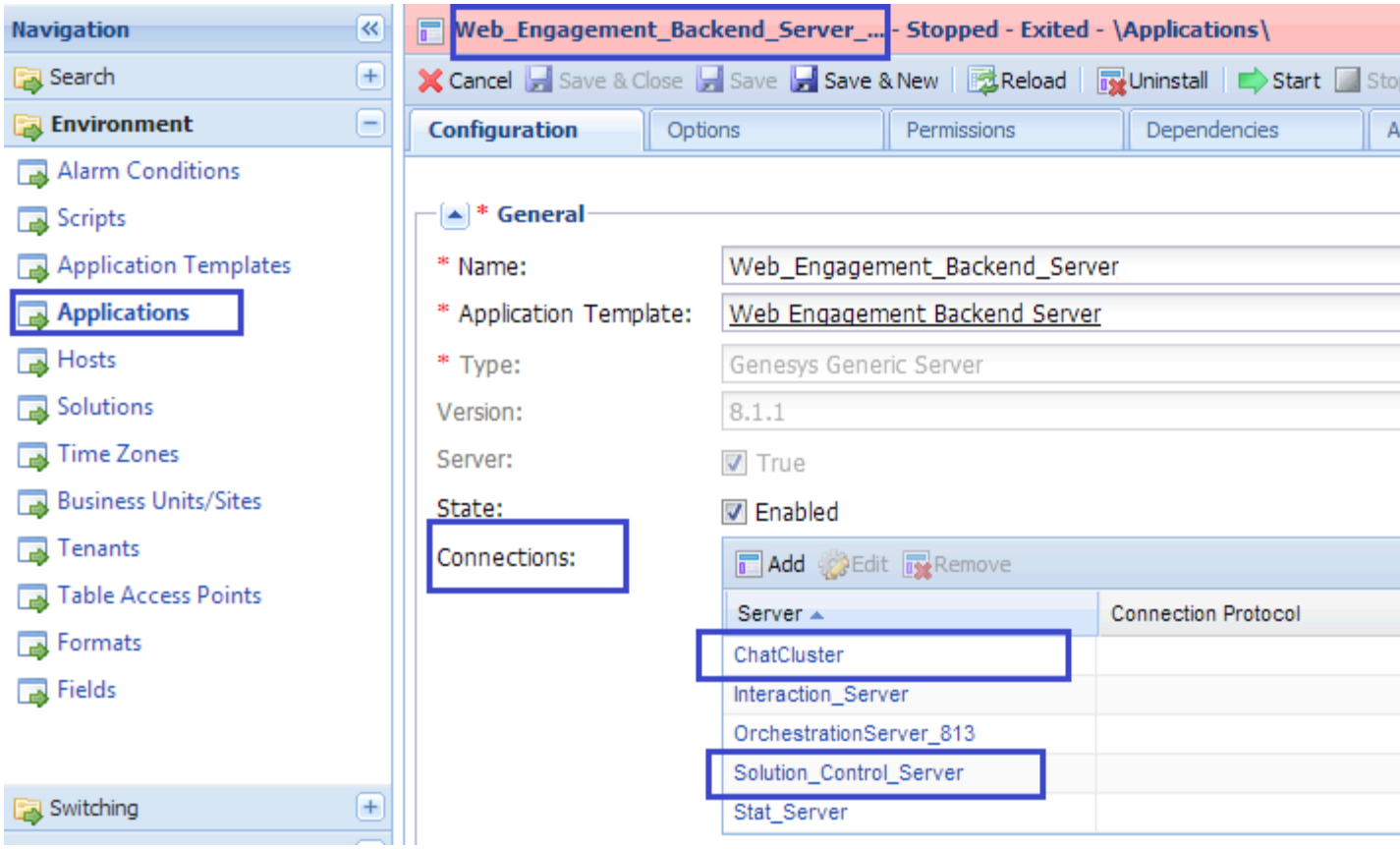
Back to top

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

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

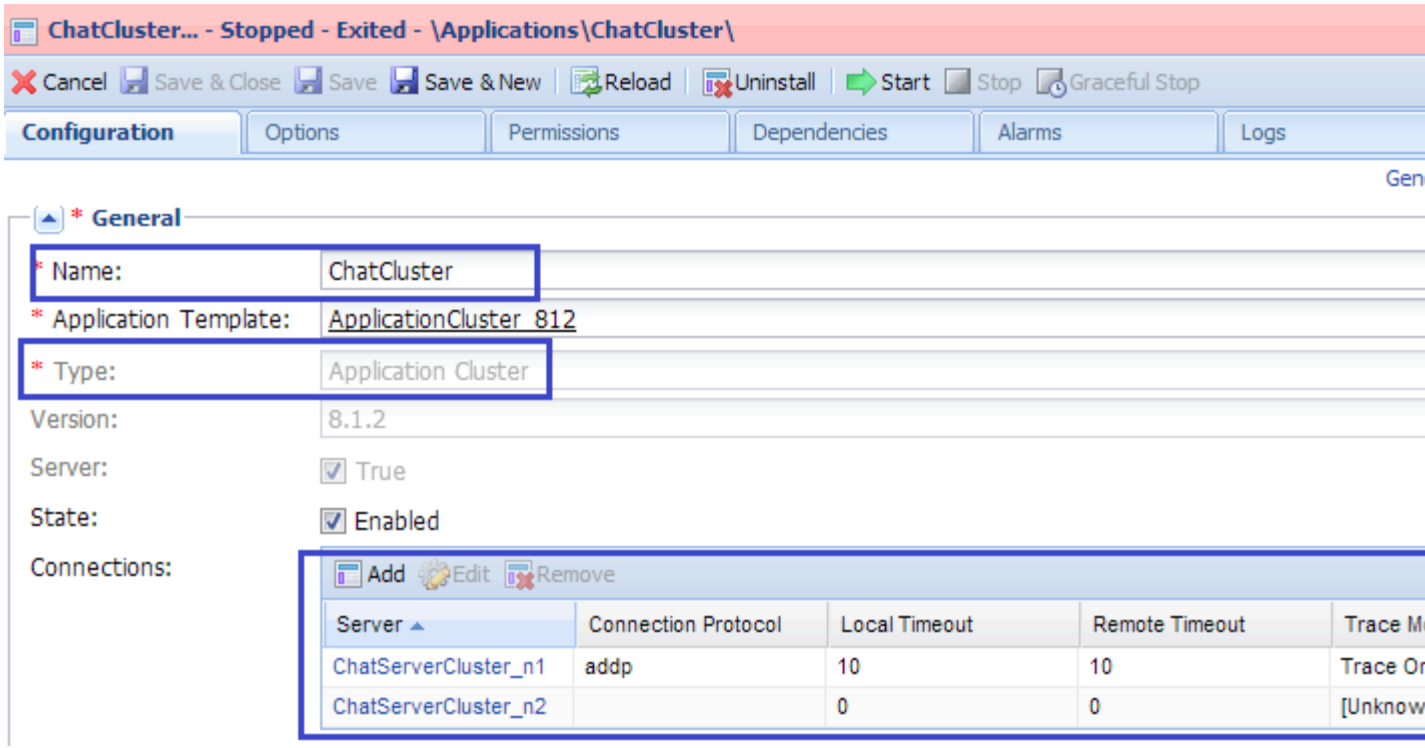
Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select your Backend Server application, and click **Edit**.
2. 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 Backend Server has a connection to the **ChatCluster** Application Cluster application.

3. Click **Save & Close**
4. Open your Application Cluster application.
5. In the Connections section, add connections to one or more Chat Servers.



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

6. Click **Save & Close**.

End

Back to top

Installing the Backend Server

Install the Backend Server on Windows or Linux.

Windows

Prerequisites

- Configuring the Backend Server 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 Backend Server Application that you are installing. The **Application Properties** area shows the **Type**, **Host**, **Working Directory**, **Command Line executable**, and **Command Line Argument**s 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 Backend Server. When through, the **Installation Complete** screen appears.
13. Click **Finish** to complete your installation of the Backend Server.
14. Inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.

End

Linux

Prerequisites

- Configuring the Backend Server 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, `demosrv.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 Backend Server application 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 Backend, version 8.1.x has completed
   successfully.
   ```
8. Inspect the directory tree of your system to make sure that the files have been installed in the location that you intended.

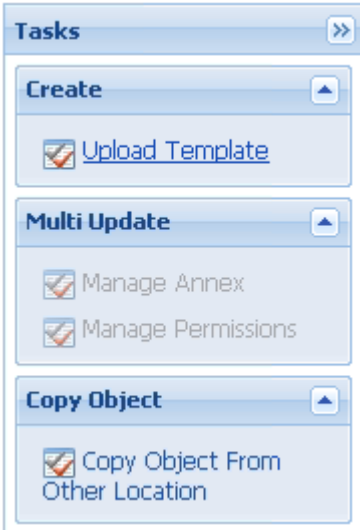
End

Back to top

Importing the Frontend Server Application Template

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_Frontend.apd** file or, if your Configuration Server does not support Web Engagement specific types, select **Web_Engagement_Frontend_Generic.apd**, available in the **templates** directory of your installation CD. The **New Application Template** panel opens.
5. Click **Save & Close**.

End

Back to top

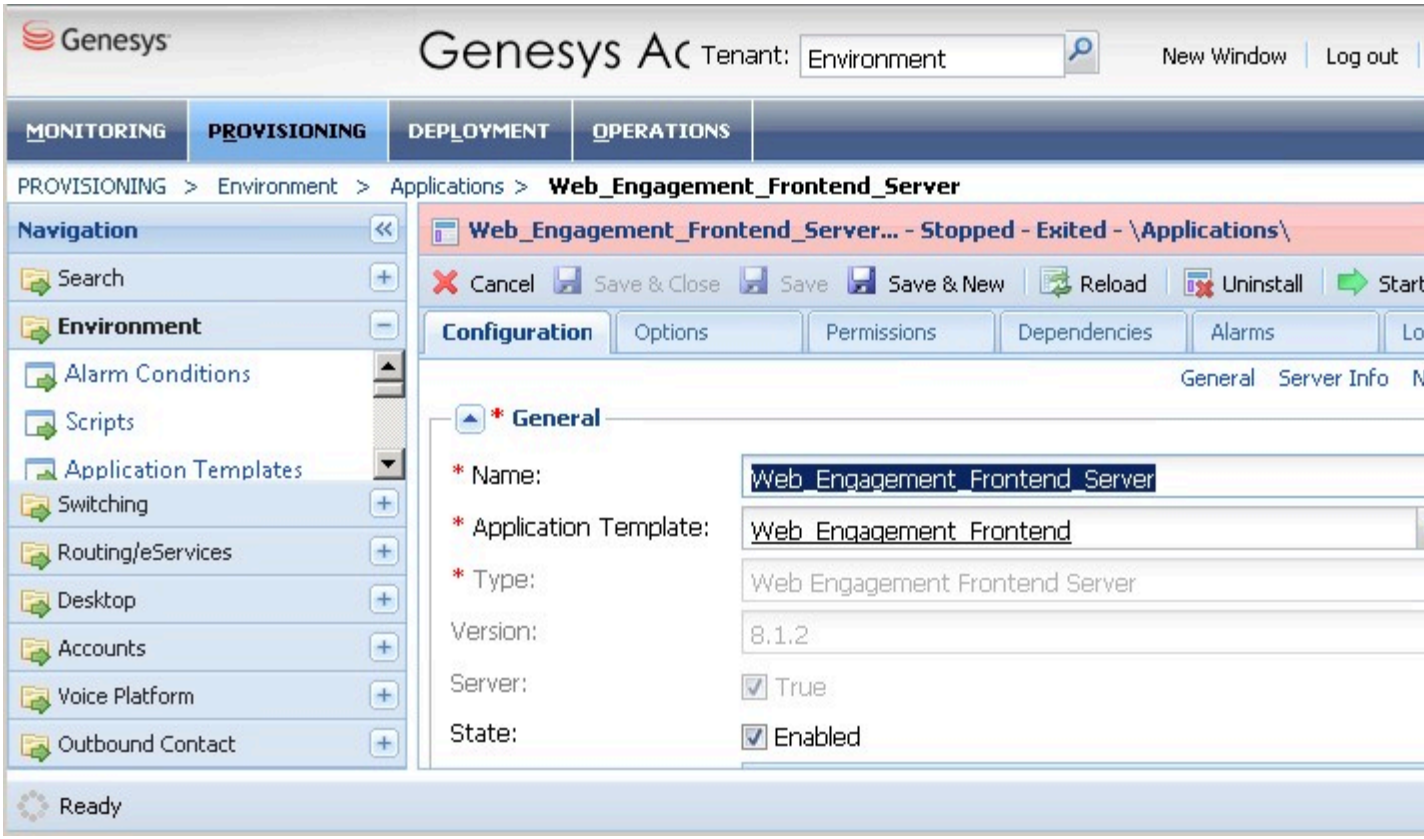
Creating the Frontend Server Application

Prerequisites

- You completed Importing the Frontend Server Application Template.

Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. In the **Tasks** panel, click **Create New Application**.
3. In the **Select Application Template** panel, click **Browse for Template** and select the Frontend Server template that you imported in Importing the Frontend Server Application 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 **Web_Engagement_Frontend.xml** file or select the **Web_Engagement_Frontend_Generic.xml** file if you chose **Web_Engagement_Frontend_Generic.apd** in Step 4 of Importing the Frontend Server Application 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, `Web_Engagement_Frontend_Server`.
   - Enable the **State**.
   - Select the **Host** on which the Frontend Server will reside.
   - Click **Create**.
8. The **Results** panel opens.
9. Enable **Opens the Application details form after clicking 'Finish'** and click **Finish**. The Frontend Server application form opens and you can start configuring the Frontend Server application.



Frontend Server application opened in Genesys Administrator.

End

Back to top

Configuring the Frontend Server Application

> **Important**
> After completing this procedure, you can open the Backend Server application and add a connection to the Frontend Server.

Prerequisites

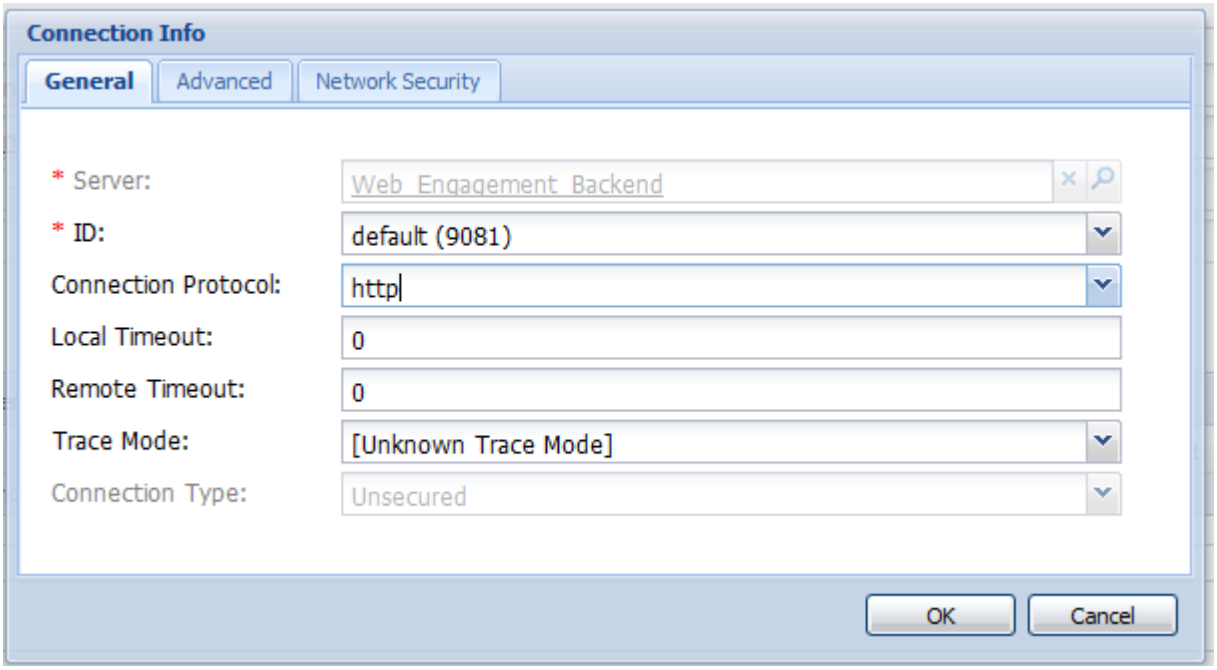
- You completed Creating the Frontend Server Application.

Start

1. If your Frontend Server application form is not open in Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select the application defined for the Web Engagement Frontend

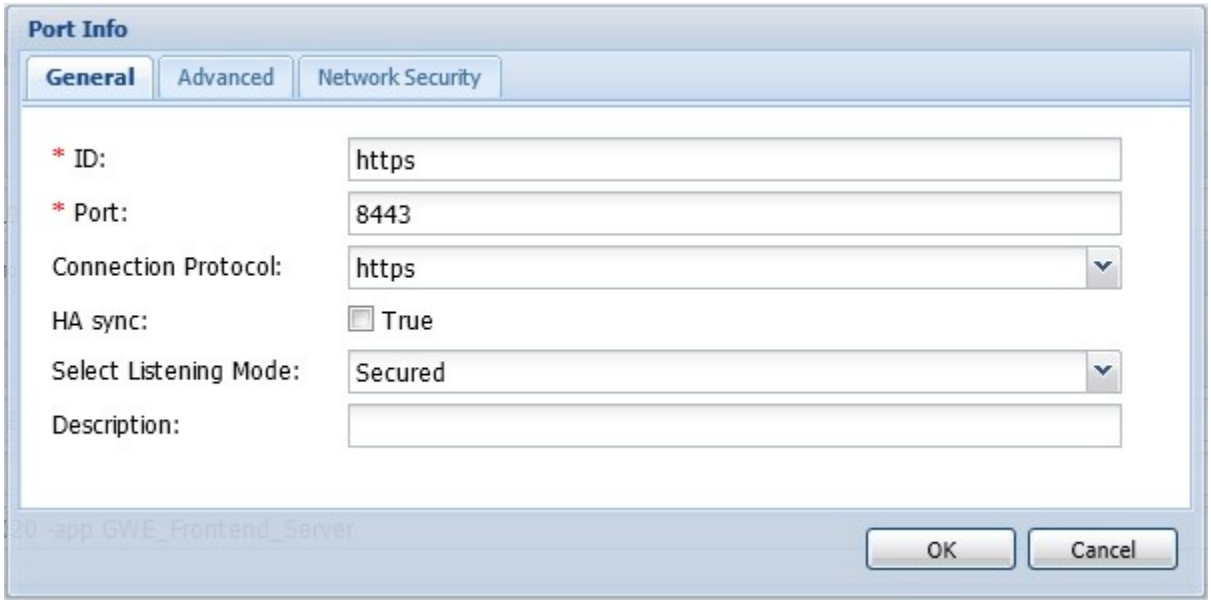
Server and click **Edit...**.

2. In the Connections section, click **Add**. The **Browse Applications** dialog opens. Select the Backend Server application and click **OK**. Optionally, you can repeat this step to configure a connection to Message Server, which will enable you apply the **network** logging option.
3. Select the Backend Server row in the list of connections and click **Edit**. In the **Connection Info** window, set `http` for the **Connection Protocol**. Click **OK**. **Note:** If you skip this step, the Web Engagement solution will not work.



Set the Connection Protocol to `http`.

4. Expand the **Server Info** pane.
5. In the Tenant section, click **Add** and select your tenant. For instance, `Environment`. Click **OK**.
   Note: The Frontend Server and Backend Server applications should belong to the same Tenant. See Multi-Tenancy for details.
6. If your **Host** is not defined, click the lookup icon to browse to the hostname of your application.
7. In the Listening Ports section, create the default port by clicking **Add**. The **Port Info** dialog opens.
   - Enter the **Port**. For instance, `8081`.
   - Choose `http` for the **Connection Protocol**.
   - Click **OK**.
8. Optionally, you can add a secure listening port by clicking the **Add** button. The **Port Info** dialog opens.
   - Enter `https` for the **ID** field. This specific ID is required in GWE 8.1.2.
   - Enter the **Port**. For instance, 8443.
   - Choose `https` for the **Connection Protocol**.
   - Choose `Secured` for the **Listening Mode**.
   - Click **OK**.



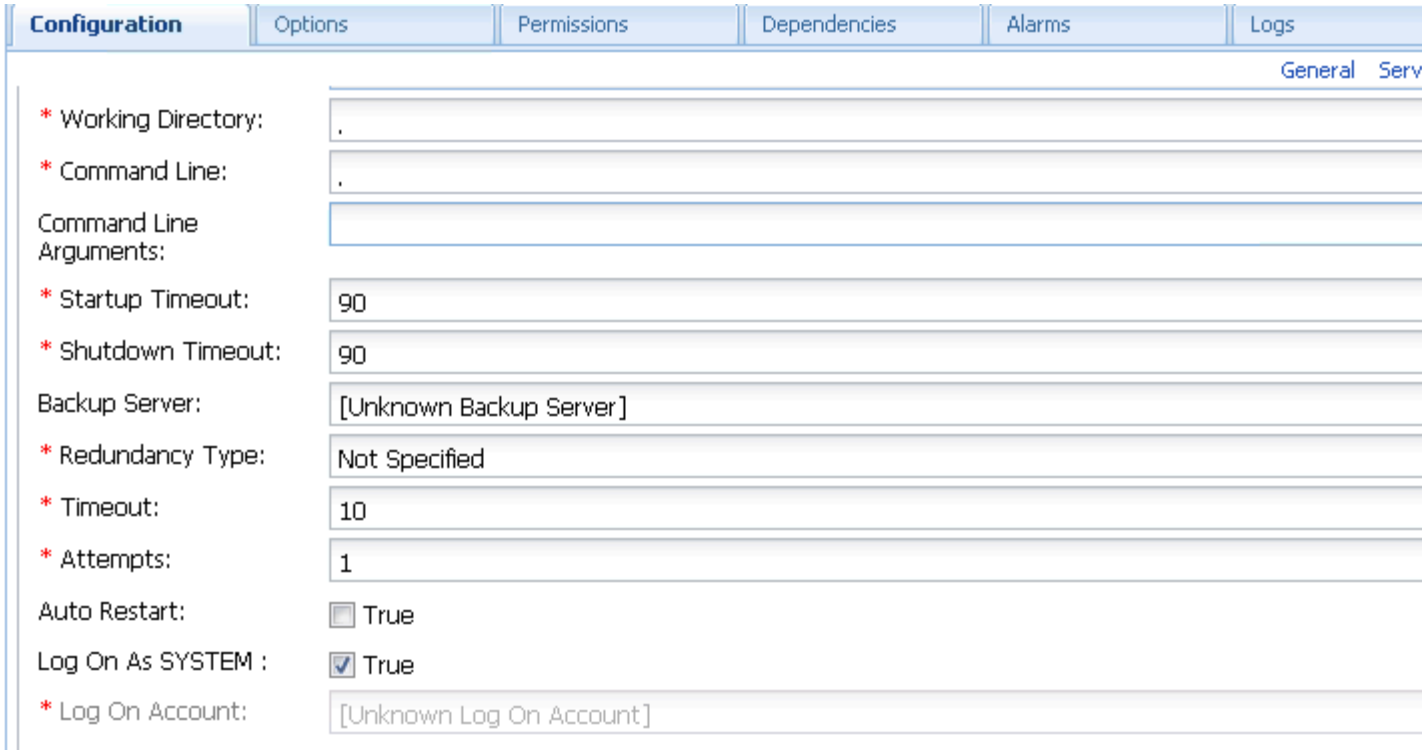
Secure listening port

9. Optionally, you can explicitly add a Jetty stop port for the Backend Server. If you do not define a stop port, it is auto-generated when you start the server. To specify the stop port, click the Add button. The `Port Info` dialog opens.
   - Enter `stop` for the **ID** field. This specific ID is required.
   - Enter the **Port**. For instance, `18081`.
   - Click **OK**. The `stop` port appears in the list of Listening ports.

ID	Port
default	8081
https	8443
stop	18081

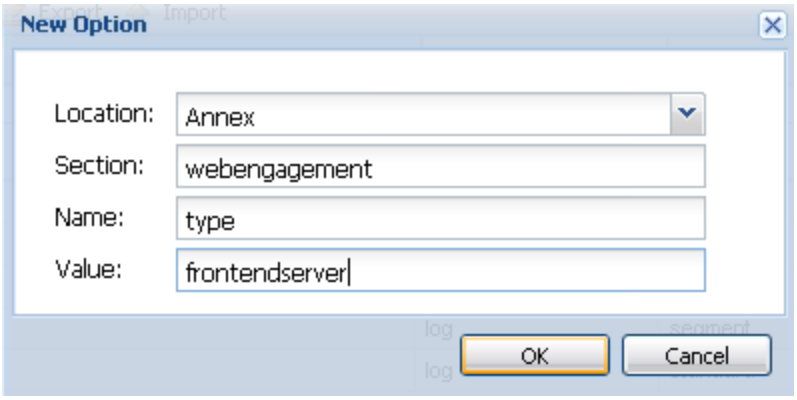
Listening ports

10. Ensure the **Working Directory** and **Command Line** fields contain "." (period). They are automatically populated when the Frontend Server is installed.



Commands for the Frontend Server.

11. Click **Save**.
12. The **Confirmation** dialog for changing the application's port opens. Click **Yes**.
13. (Optional) Select the **Options** tab. 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\Frontend_Server` to force the system to write logs in the console and in a file.
14. (Optional) You need to create a **webengagement** annex only if your Web Engagement Frontend Server application's type is Genesys Generic Server. Click **New** in the **Options** tab. The **New Option** dialog opens.
    - Select `Annex` for **Location**.
    - Enter `webengagement` for **Section**.
    - Enter `type` for **Name**.
    - Enter `frontendserver` for **Value**.



Annex for the Frontend.

- Click **OK**. You can see it by selecting `Advanced View (Annex)` in the **View** selector.

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

End

Next Steps

Open your Backend Server application and add a connection to the Frontend Server.

Back to top

Installing the Frontend Server

Install the Frontend Server on Windows or Linux.

Windows

Prerequisites

- Configuring the Frontend Server

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 **Select Installed Application** screen appears.
3. Select the Backend Server application you created in Creating the Backend Server Application.
4. Click **Next**. The **Connection Parameters to the Configuration Server** screen appears.
5. 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.)
6. Under **User**, enter the user name and password for logging on to Configuration Server.

7. Click **Next**. The **Select Application** screen appears.
8. Select the Web Engagement Frontend Server Application 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.
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 Frontend Server. When through, the **Installation Complete** screen appears.
13. Click **Finish** to complete your installation of the Frontend Server.

End

Linux

Prerequisites

- Configuring the Frontend Server

Start

1. Open a terminal in the Genesys Web Engagement CD/DVD or the Genesys Web Engagement IP, and run the **Setup.sh** file. The Genesys Installation starts.
2. Select the Backend Server application you created in Creating the Backend Server Application.
3. Enter the hostname of the host on which you are going to install.
4. Enter the connection information to log in to the Configuration Server:
   - The hostname. For instance, `demosrv.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.
5. Enter the key for the Web Engagement Frontend Server application that you created previously 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 Frontend, version 8.1.x has completed successfully.`

End

Back to top

Next Steps

- You can now 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 Development Tool

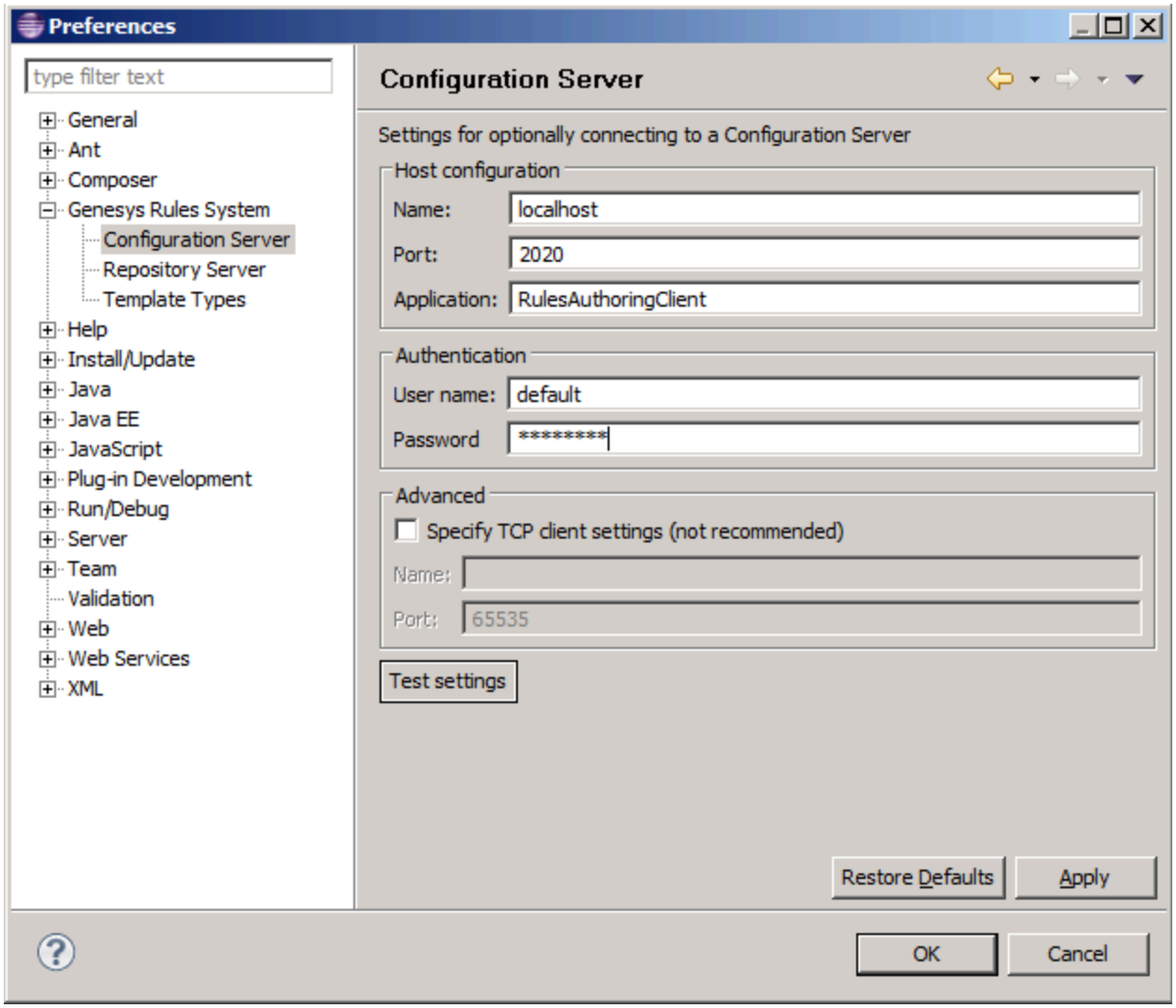
Configuring Genesys Rules Development Tool

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

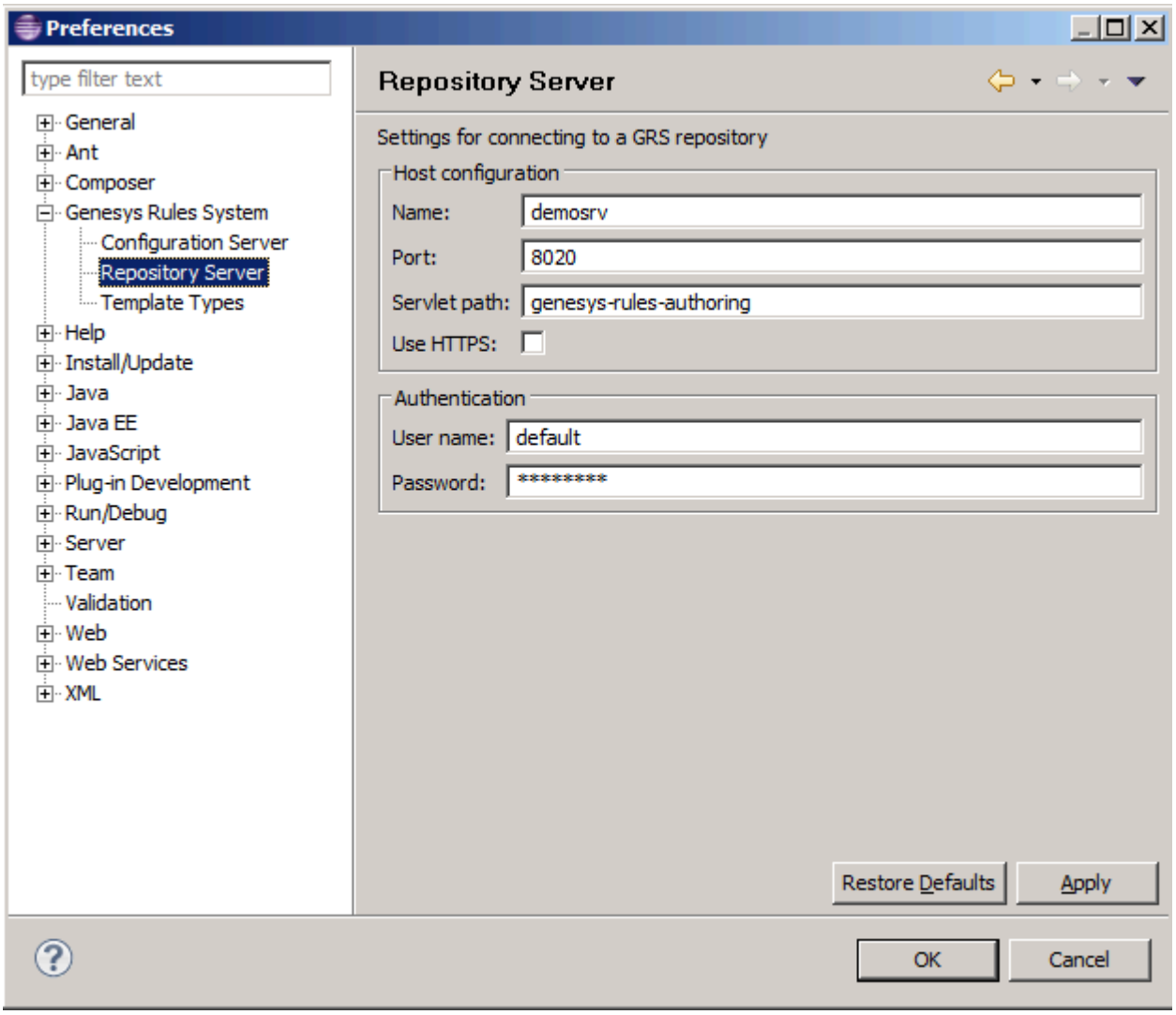
1. Open Genesys Rules Development Tool by starting Composer.
2. Navigate to **Window > Preferences**. The **Preferences** window opens.
3. Navigate to **Genesys Rules System > Configuration Server**. Edit the settings.
   - Enter the Configuration Server host name. For instance, `localhost`.
   - Enter 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.

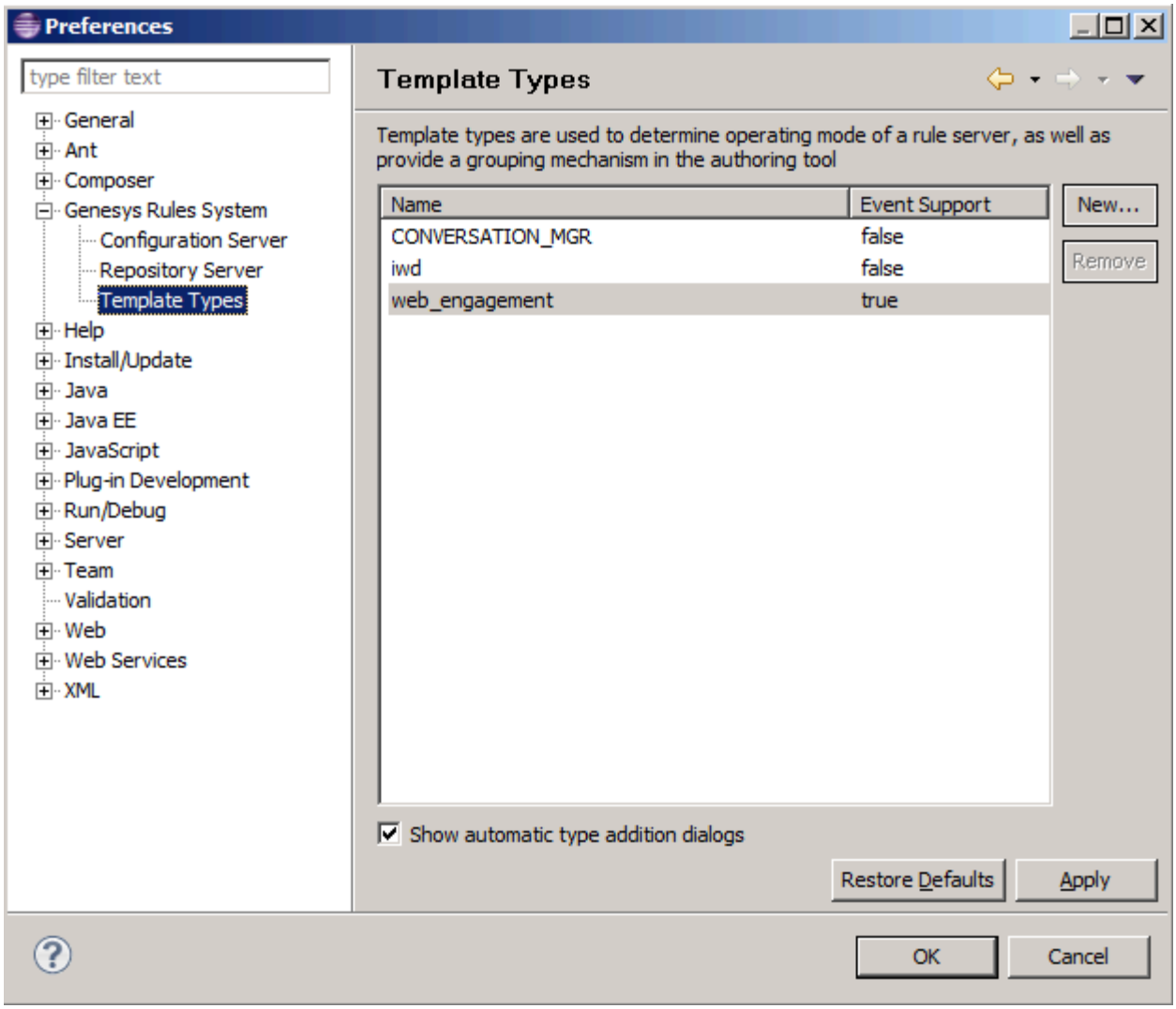
4. Navigate to **Genesys Rules System > Repository Server**. Edit the settings.
   - Enter the Repository Server host name. This is the name of the host specified for the GRS application (application with type Business Rules Execution Server) in Genesys CME. For instance, `localhost`.
   - Enter the 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 Genesys CME. 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

Genesys Rules Authoring Tool

Configuring Genesys Rules Authoring Tool

Prerequisites

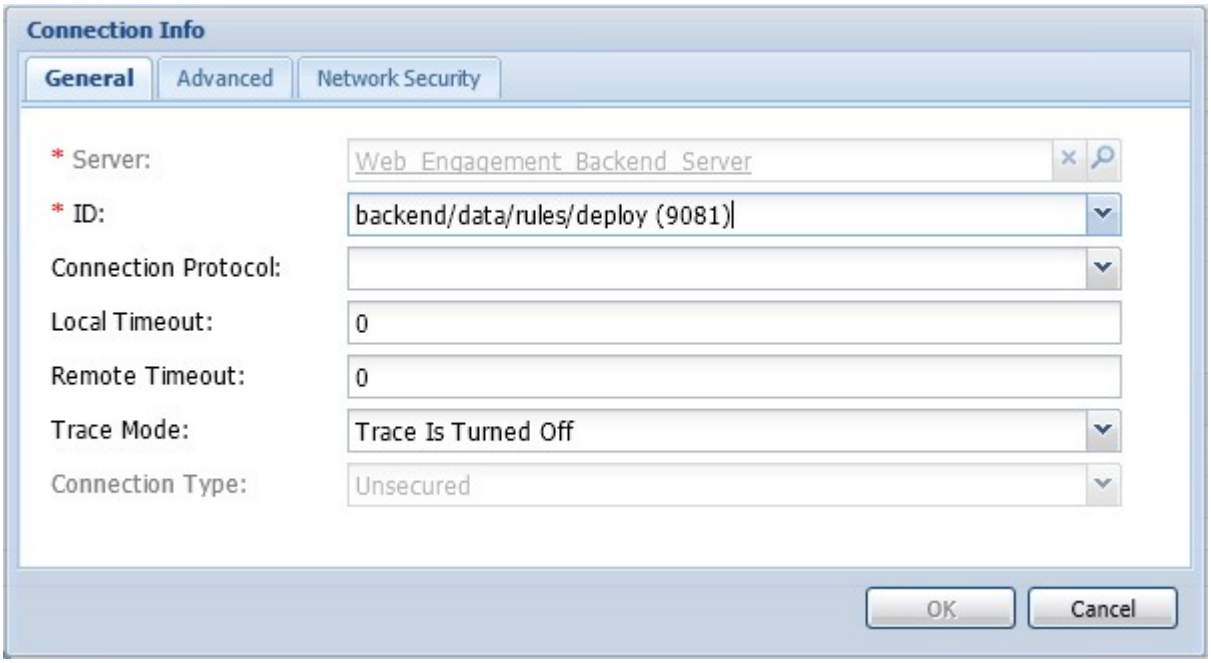
- Your environment includes Genesys Rules Authoring Tool (GRAT). See Genesys environment

prerequisites for compliant versions. For more information about installing GRAT, refer to the Genesys Rules System Deployment Guide.

Start

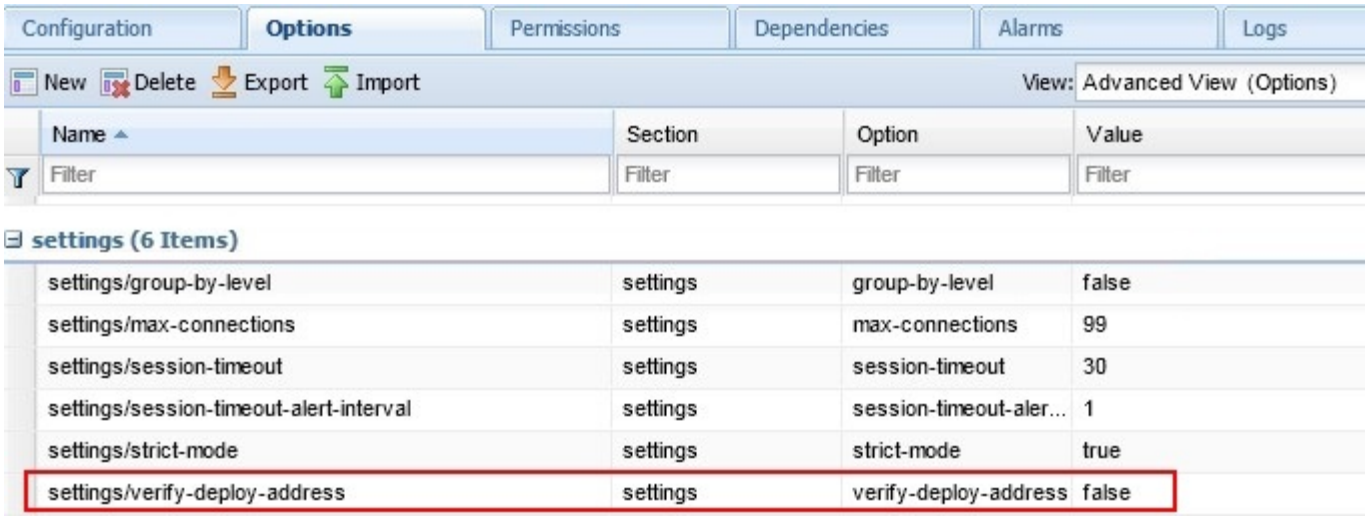
1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the GRAT application (the application with type Business Rules Application Server), and click **Edit**.
2. In the Connections section, click **Add...**.
3. Select the Genesys Web Engagement Backend Server and 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.
4. Select the Backend Server in the list of connections and click **Edit**.
5. In the **Connection Info** window, select `backend/data/rules/deploy` for the ID field. Click **OK**.



ID is set to the `backend/data/rules/deploy` 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 Backend Server, 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.



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

8. In the **Security** 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

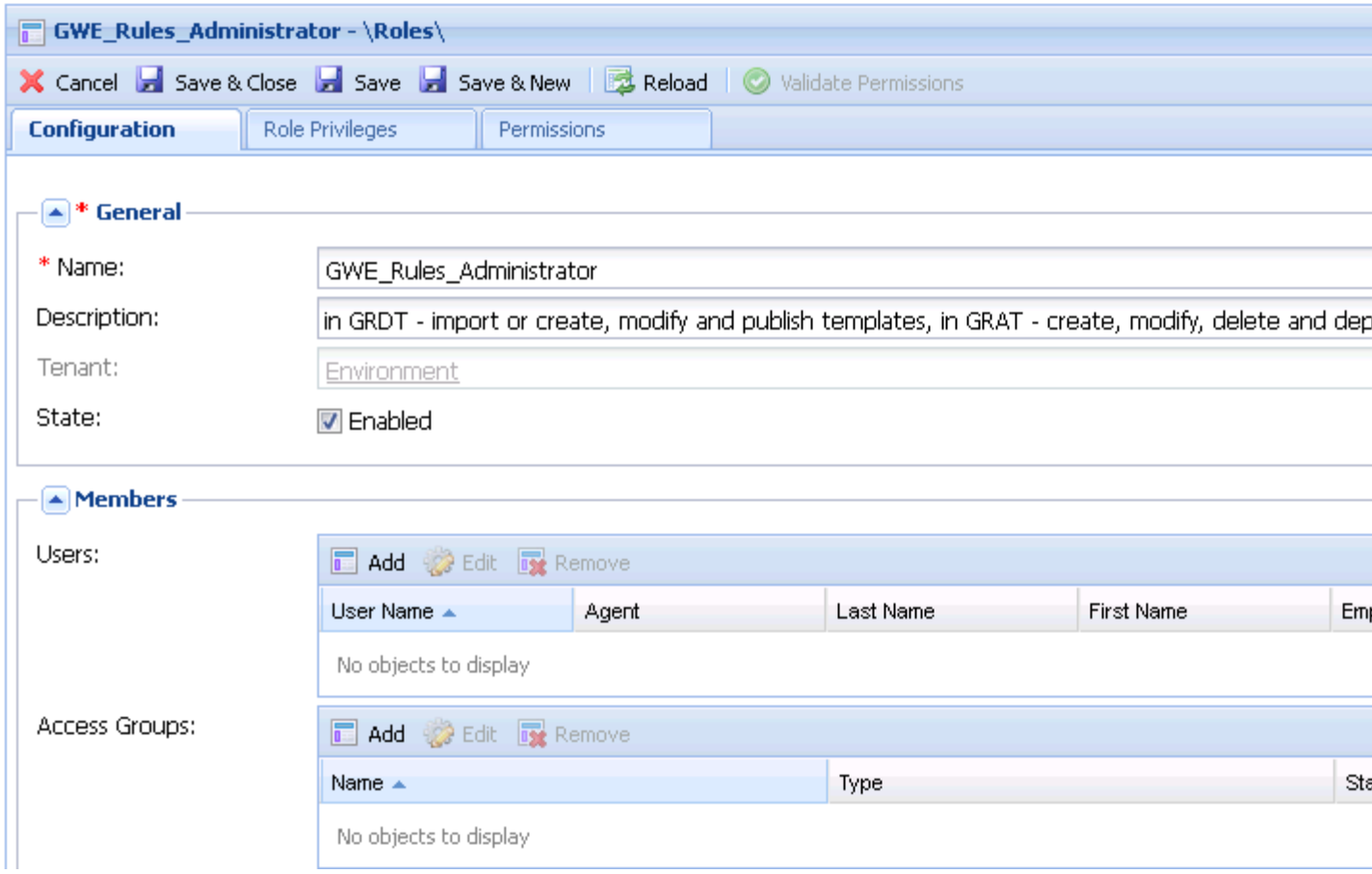
(Optional) Configuring Roles Settings for Rules Management

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 assigned 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).

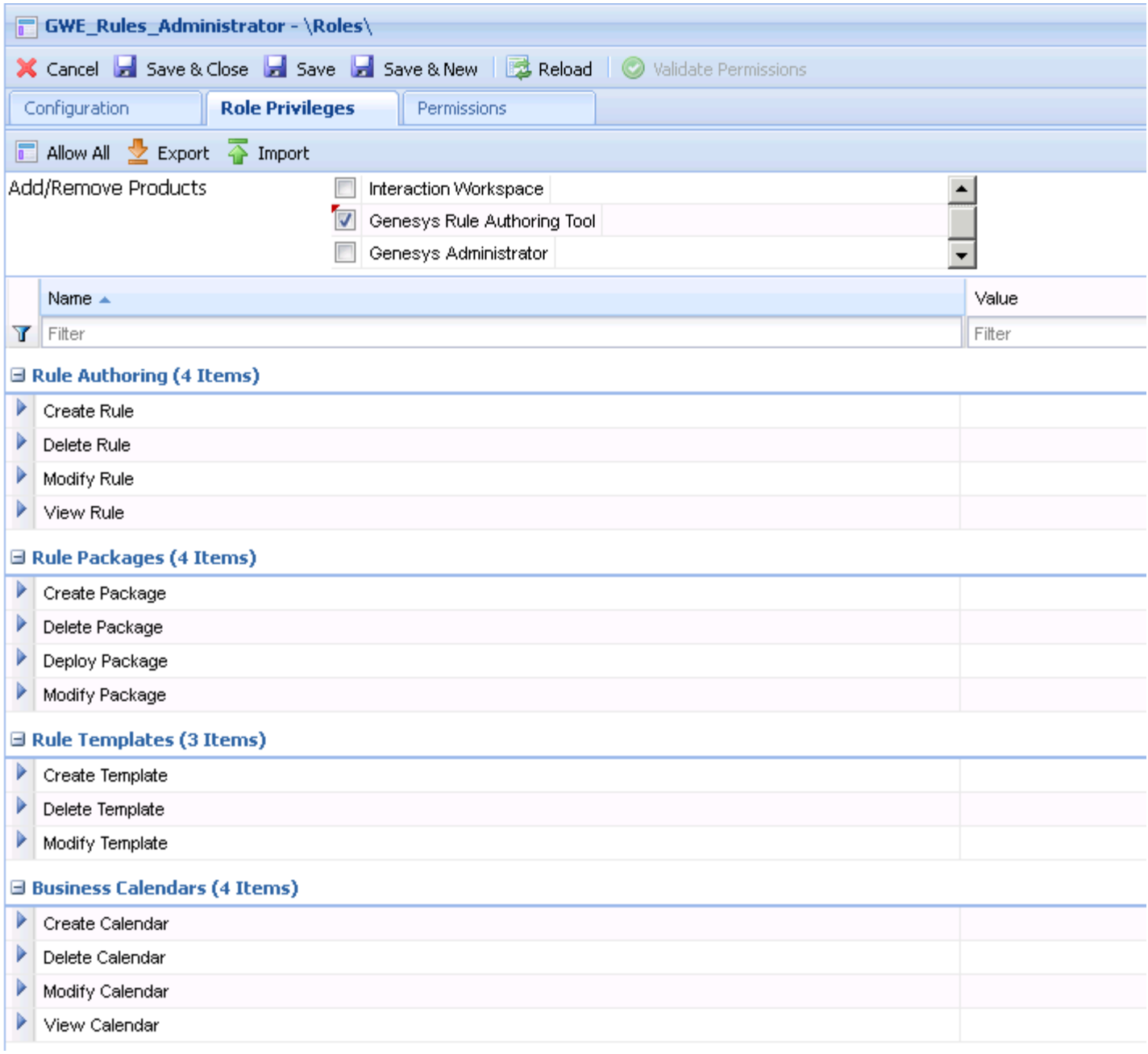
Start

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 Memebers section, you can specify who should have this role. Click **Add** to add as many access groups or users as you need.



New Role

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

Next Steps

- Configure the Generic Cassandra Settings for your Backend Server host.

Configuring Cassandra

Genesys Web Engagement version 8.1.2 includes an embedded Cassandra database (version 1.2.15). To enable Cassandra to function optimally, you must configure the embedded data storage on your Backend Server environment by completing the procedures on this page.

Configuring the Java Heap Size

Start

1. Navigate to your installation directory for the Backend Server and open the **setenv.bat** file with a text editor.
2. Find the "Java Virtual machine settings" section.
3. Modify the **-Xmx** and **-Xms** parameters according to your hardware configuration. Genesys recommends that you specify a Java Heap size of 1/4 of your system memory, but not greater than 8 GB. For example, your configuration could look like the following:

   ```
   set JAVA_OPTS=%JAVA_OPTS% '''-Xms4096m -Xmx4096m''' -XX:MaxPermSize=250m
   -XX:+HeapDumpOnOutOfMemoryError
   ```

   Consult the Cassandra documentation at http://www.datastax.com/documentation/cassandra/1.2/cassandra/operations/ops_tune_jvm_c.html for more information about Cassandra clusters and memory.
4. Save your changes.

End

Configuring the Embedded Cassandra Server Properties

Prerequisites

- You completed Building your Application

Start

1. Navigate to the Web Engagement installation directory and edit the **\apps*application name*\environment\environment.local.properties** file. The values specified in this file are inserted into the **cassandra.yaml** and **wmbackend.properties** files during the application build step. All configuration parameters for Cassandra in **environment.local.properties** start with **wmdb.cassandra.cluster**. The parameters and their default values:

   ```
   # WMDB-Cassandra
   # Tokens in cassandra.yaml and wmbackend.properties files will be substituted by these
   parameter values.
   wmdb.cassandra.cluster.name=Cluster
   wmdb.cassandra.cluster.keyspaceName=WebMonitoring
   ```

```
wmdb.cassandra.cluster.defaultStrategyParams=
wmdb.cassandra.cluster.defaultStrategy=SimpleStrategy
wmdb.cassandra.cluster.defaultReplicationFactor=3
wmdb.cassandra.cluster.listenAddress=localhost
wmdb.cassandra.cluster.rpcAddress=localhost
wmdb.cassandra.cluster.rpcPort=19160
wmdb.cassandra.cluster.sslStoragePort=17001
wmdb.cassandra.cluster.storagePort=17000
wmdb.cassandra.cluster.seedNodes=127.0.0.1
wmdb.cassandra.cluster.dataDirectory=DATA_DIR_PLACEHOLDER
wmdb.cassandra.cluster.commitLogDirectory=COMMITLOG_DIR_PLACEHOLDER
wmdb.cassandra.cluster.savedCachesDirectory=SAVED_CACHES_DIR_PLACEHOLDER
```

For information about all the Cassandra configuration properties, refer to the Cassandra documentation: http://www.datastax.com/documentation/cassandra/1.2/cassandra/configuration/configCassandra_yaml_r.html
Note: Making changes in the **environment.local.properties** file keeps the client and server connection parameters synchronized.

2. Update the Cassandra configuration parameters with values appropriate for your deployment. You will need to modify the following parameters:
   - **wmdb.cassandra.cluster.defaultStrategy**—The default value is `SimpleStrategy`, but if you plan to deploy the cluster across multiple data centers, consider using `org.apache.cassandra.locator.NetworkTopologyStrategy`.
     - **SimpleStrategy**
       - **wmdb.cassandra.cluster.defaultReplicationFactor**—The keyspace replication factor, for use with `SimpleStrategy` only. When choosing a replication factor value, take into account that by default Genesys Web Engagement instructs Cassandra to use the QUORUM consistency level for both writes and reads. Genesys recommends that you set the replication factor to a value of not less than 3. You can get a sense of the data consistency and cluster high availability of a configuration by using the Cassandra calculator that is available at http://www.ecyrd.com/cassandracalculator/.
     - **org.apache.cassandra.locator.NetworkTopologyStrategy**
       - **wmdb.cassandra.cluster.defaultStrategyParams**—Cross–data center replication strategy parameters for `NetworkTopologyStrategy`. For example, `DataCenter1:3, DataCenter2:2` indicates a replication factor of 3 for `DataCenter1` and 2 for `DataCenter2`.
       - The endpoint_snitch parameter in cassandra.yaml must be set to GossipingPropertyFileSnitch.
       - A **cassandra-rackdc.properties** file with the name of the selected data center MUST be placed in the **<webengagement_root>/servers/backend/resources** folder for every Backend Server.

         Details for **cassandra-rackdc.properties** are available at http://docs.datastax.com/en/cassandra/2.0/cassandra/architecture/architectureSnitchGossipPF_c.html.

> **Important**
>
> If you are using a multi–data center configuration, the consistency level applies to every data center. You must select the appropriate consistency level for your specific configuration.
>
> In particular, please note that the QUORUM consistency levels have limitations when applied to a multi–data center configuration. Note also that **NetworkTopologyStrategy** is not compatible with some QUORUM consistency levels when they are applied to a single node in a selected data center. For example:
> `DC1:1, DC2:1` does not work with LOCAL_QUORUM, QUORUM and EACH_QUORUM consistency levels
> `DC1:2, DC2:1` works with LOCAL_QUORUM for DC1 clients, but not with QUORUM and EACH_QUORUM
> `DC1:1, DC2:2` works with LOCAL_QUORUM for DC2 clients, but not with QUORUM and EACH_QUORUM
>
> `DC1:2, DC2:2` works with all QUORUM consistency levels.

- **wmdb.cassandra.cluster.seedNodes**—Choose your seed nodes, keeping in mind that if you plan to deploy the Cassandra cluster across multiple data centers, you should have at least one seed node from each data center. Set the value of **wmdb.cassandra.cluster.seedNodes** to a comma-separated list of the IPs for your seed nodes. For example, `135.225.54.236, 135.225.54.245`.
- **listen_address** and **rpc_address**—These parameters are likely to be different for each Backend Server / Cassandra node in the cluster and must be set directly in the **\servers\backend\etc\cassandra.yaml** file for each Backend Server. See Configuring Cassandra for the Cluster for details on setting these parameters.

3. Save your changes.

End

Next Steps

- If you are completing the Standalone deployment scenario, the next step is to install the GWE plug-ins for Interaction Workspace and Genesys Administrator Extension.
- If you are completing the Clustering deployment scenario, you can return to Configuring Cassandra for the Cluster.

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 **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. Typically, you need to do this as part of the Updating the Backend Server Application procedure when you are Deploying and Configuring the Genesys Web Engagement Cluster.

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: Genesys and Playground. Provisioning creates category objects to support these applications:

- Genesys sample application — Categories start with the **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 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_Rejected
  - Webengagement_Timeout
  - Webengagement_Failed

Location in Genesys Administrator: **Provisioning > Routing/eServices > Interaction Queues (or 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_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_Rejected.Routing
- Webengagement_Timeout.Routing
- Webengagement_Failed.Routing

Location in Genesys Administrator: **Provisioning > Routing/eServices > Orchestration (or 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 Interaction Workspace.

> ### 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 Interaction Workspace 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 Backend Server application. It creates an option in the **[endpoints:*Backend 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 Backend Server 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 Backend Server application

- The wes.connector.chatServer.queueKey in the **[service:wes]** section is populated with the *Backend 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 Backend Server application include the Frontend Server, Chat Server (or an Application Cluster object which contains a connection to Chat Server), Interaction Server, Orchestration Server, and the Stat Server applications.
- The connections for the Frontend Server application include the Backend Server.
- If you are using Genesys Generic Server templates, you have a **webengagement** section created in the annex for the Backend and Frontend server applications and the **type** option is set to:
  - `backendserver` for the Backend Server
  - `frontendserver` for the Frontend Server

    See Configuring the Frontend Server Application and Configuring the Backend Server Application for details.

Start

1. Navigate to the Web Engagement installation directory and open the **tools\provisioning** folder.
2. To launch provisioning, open a Windows Command Prompt (**cmd.exe**).
3. If this is your first time running the Provisioning Tool, type: `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 Frontend Server`*

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

   If this is *not* your first time running the Provisioning Tool, use 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 made manually on GWE-specific objects. `webengagement_provisioning.bat -host` *`Configuration Server host name or IP address`* `-port` *`Configuration Server port`* `-user` *`user`* `-password` *`password`* `-app-overwrite`

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

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

End

Tuning Your JVM

Web Engagement 8.1.2 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*\servers\frontend** and ***Web Engagement installation directory*\servers\backend**.

Tuning Your Memory Allocation

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

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

If you want to change these settings, update the appropriate values and restart the related Web Engagement Server (Frontend or Backend).

Enabling the JMX Port

You can monitor the JVM where your Web Engagement Server (Frontend or Backend) 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 (Frontend or Backend) 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.bat** 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.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 (Frontend or Backend). 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 Interaction Workspace

The Genesys Web Engagement Plug-in for Interaction Workspace allows you to enable chat and web callback engagement features in Interaction Workspace. See Genesys Web Engagement Plug-in for Interaction Workspace Help for details.

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

1. Installing the Plug-in for Interaction Workspace
2. Importing the Plug-in for Interaction Workspace Template
3. Adding a Connection to the Backend Server if you are following the Standalone deployment scenario. If you are following the Clustering deployment scenario, complete Adding a Connection to the Load Balancer for the Backend Servers.
4. Configuring Role-Based Access Control
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 Interaction Workspace Deployment Guide:
   - Procedure: Enabling agents to be Team Leads
   - Monitoring Chat Interactions

Installing the Plug-in for Interaction Workspace

Prerequisites

- Your environment includes Interaction Workspace. See Genesys environment prerequisites for compliant versions. For more information about installing Interaction Workspace, refer to the Interaction Workspace 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 Interaction Workspace 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 Interaction Workspace. 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 Interaction Workspace installation directory:

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

End

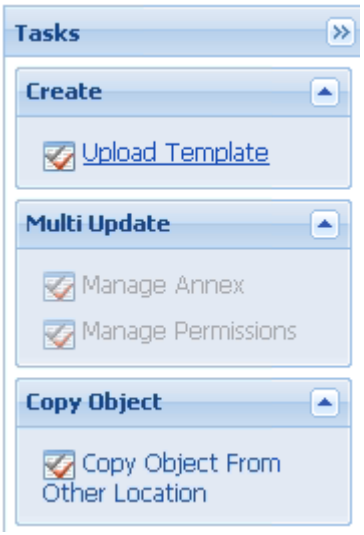
Importing the Plug-in for Interaction Workspace Template

Prerequisites

- You completed Installing the Plug-in for Interaction Workspace

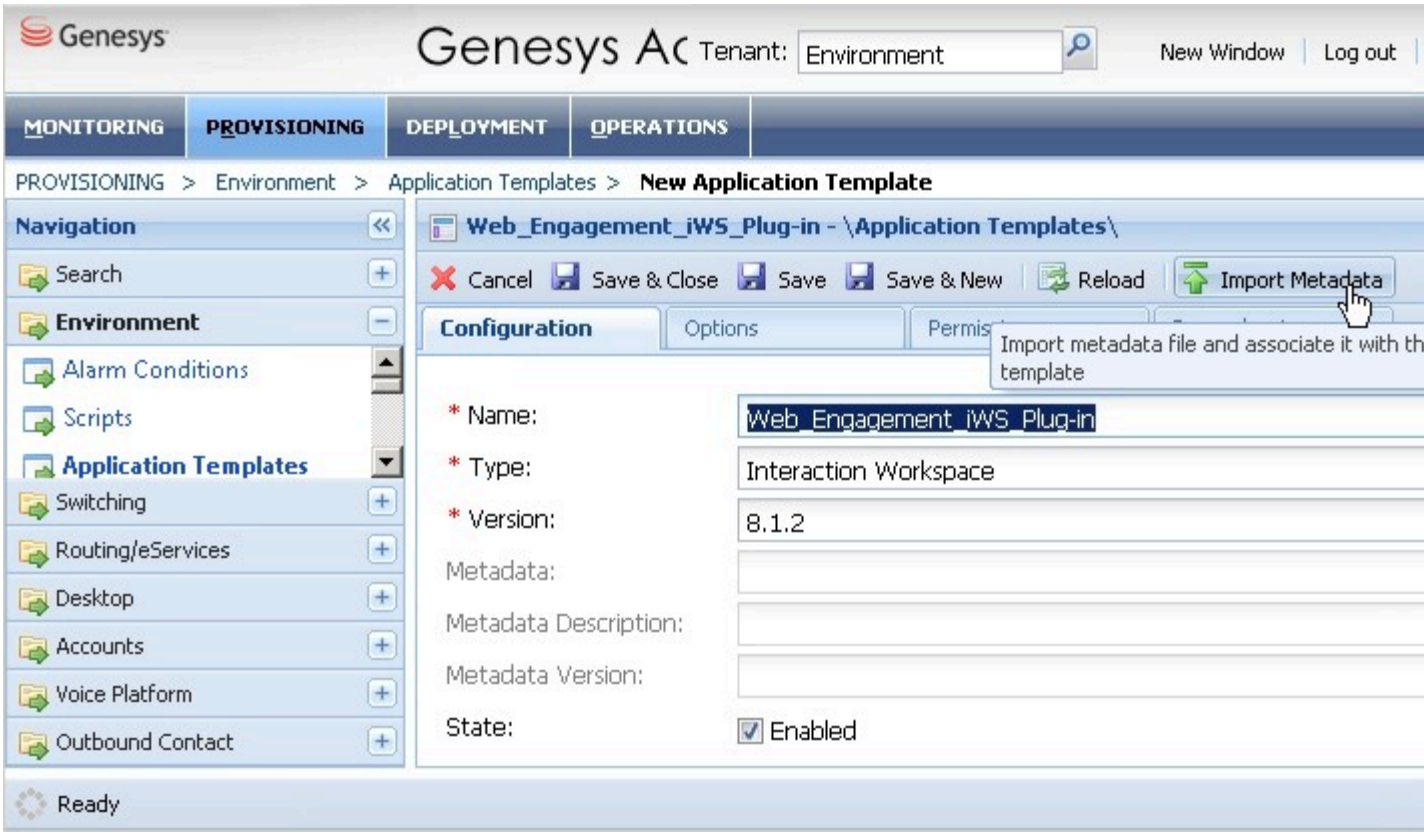
Start

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_iWS_Plug-in.apd** file. The **Configuration** tab for the new template opens.
5. Click **Import Metadata**.



Click **Import Metadata**.

6. Select the **Web_Engagement_iWS_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 Backend Server

Prerequisites

- You completed Importing the Plug-in for Interaction Workspace Template
- You are following the Standalone deployment scenario and need to add a connection to a single Backend Server.

Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the Interaction Workspace application, and click **Edit...**.
2. In the Connections section, click **Add**. The **Browse Applications** window opens.
3. Select the Web Engagement Backend Server application and click **OK**. The Backend Server is added to

the list of Connections.

4. Click **Save & Close**.

End

Adding a Connection to the Load Balancer for the Backend Servers

Prerequisites

- You completed Importing the Plug-in for Interaction Workspace Template
- You are following the Clustering deployment scenario and need to add a connection to the load balancer for the Backend Servers.

Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**, select the Interaction Workspace 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 Backend Servers — 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

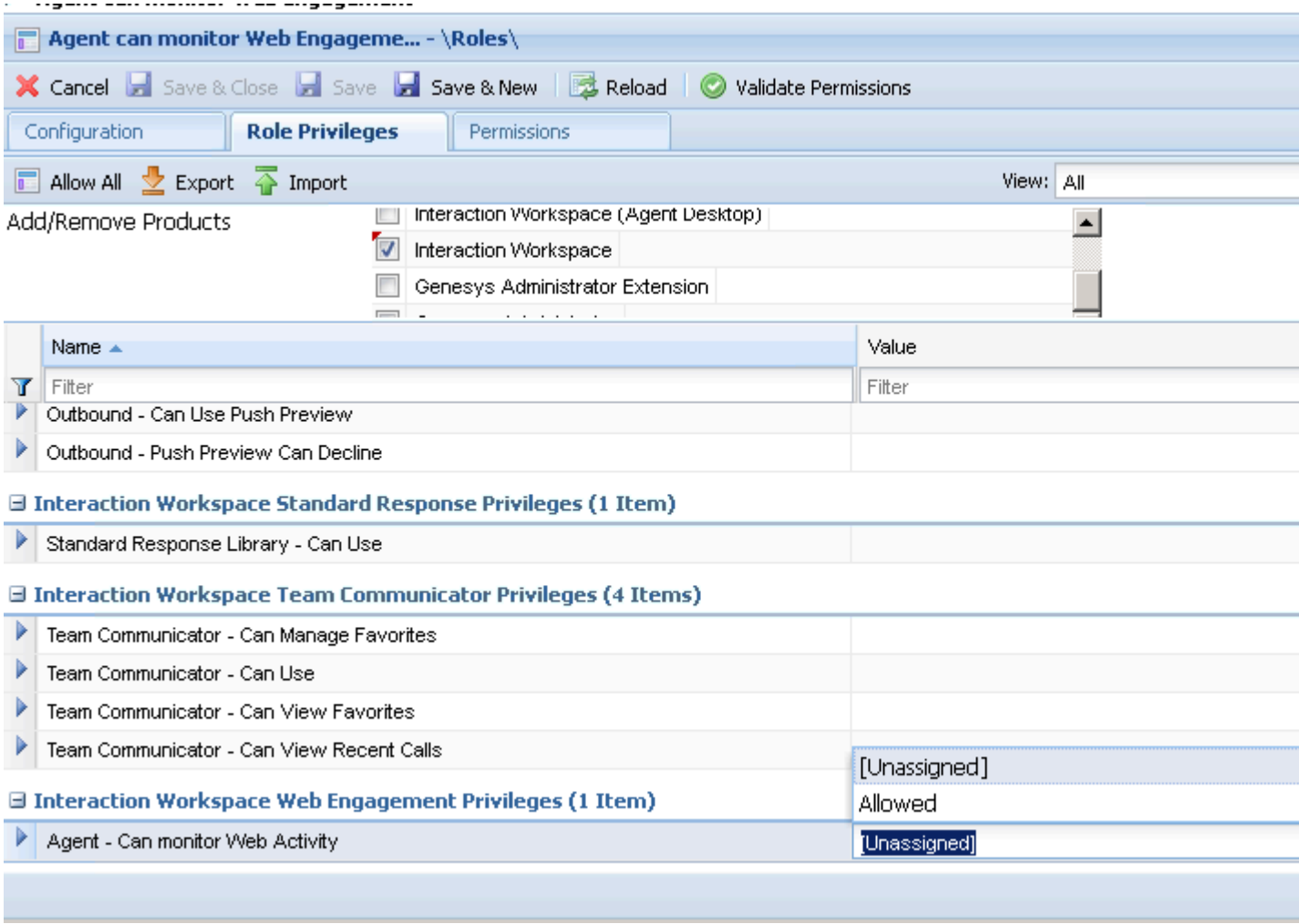
Complete this procedure to allow specific users or groups to manage Web Engagement in Interaction Workspace.

Prerequisites

- You completed Importing the Plug-in for Interaction Workspace Template

Start

1. In Genesys Administrator, navigate to **Provisioning > Accounts > Roles**.
2. Edit or create a Role responsible for managing Web Engagement in Interaction Workspace. For instance, create the Agent can Monitor Web Engagement role by clicking the **New** button.
3. Select the **Role Privileges** tab.
4. In the **Add/Remove Products** top panel, enable Interaction Workspace and expand the Interaction Workspace Web Engagement Privileges section.
5. Set the `Allowed` value for the **Agent - Can Monitor Web Activity** option.



Select Allowed

6. In the Members section of the **Configuration** tab, add the users or groups who should get this role.
7. 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 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

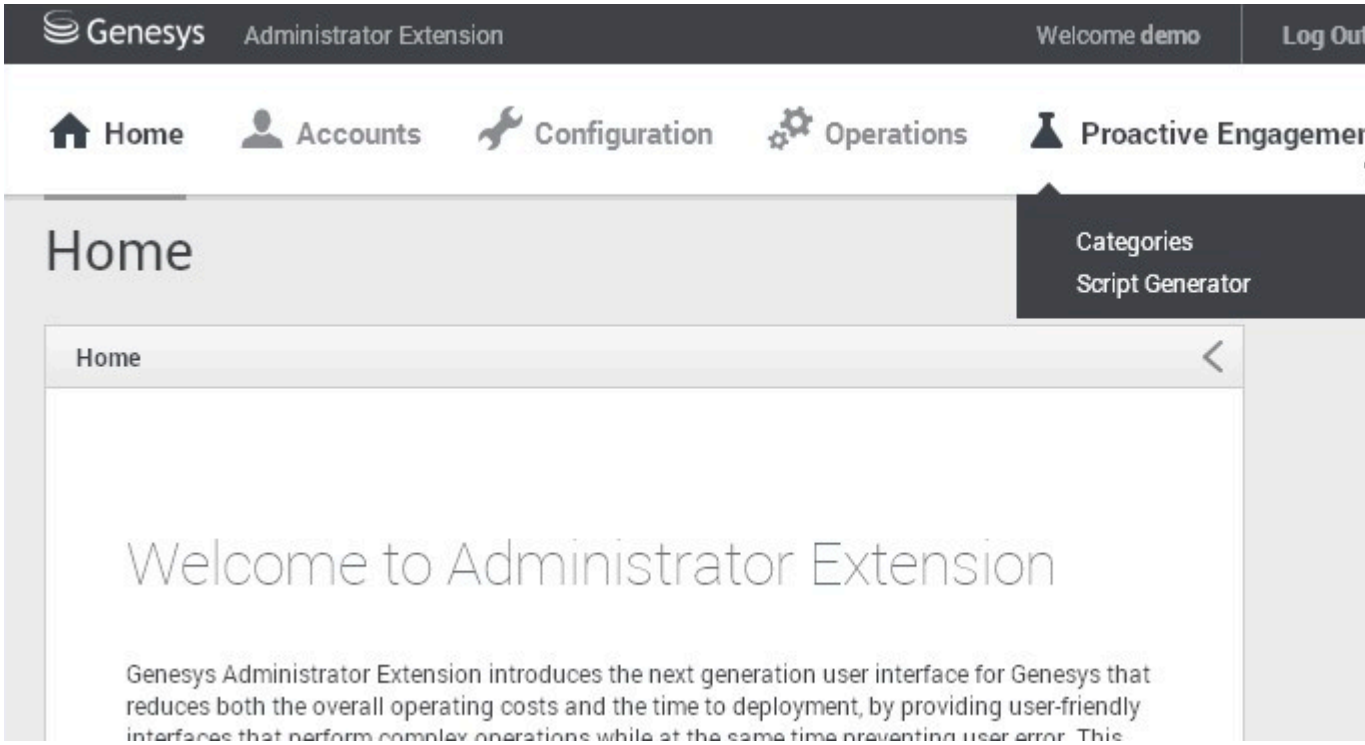
- Your environment includes Genesys Administrator Extension. See Genesys environment prerequisites for compliant versions. For more information about installing the Genesys Administrator Extension, refer to the Genesys Administrator Extension Deployment Guide.
- Installation must be driven from the host 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. Browse to the installation folder 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 Proactive 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 **Proactive Engagement** menu with the following Web Engagement items: Categories and Script Generator.



The Proactive 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.

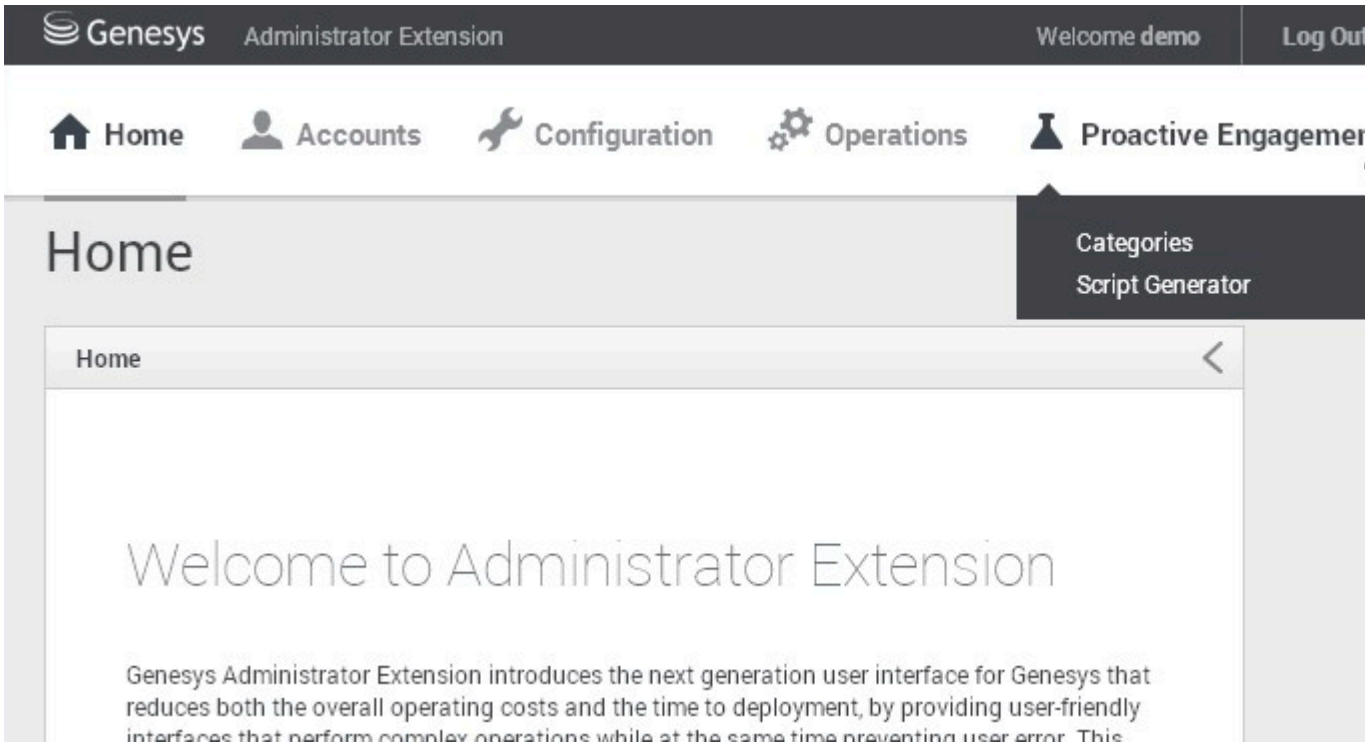
Start

1. In Genesys Administrator, navigate to `Provisioning > Environment > Applications`, select the Genesys Administrator Extension server, and stop it.
2. In your installation package, locate and double-click the `install.sh` file.
3. Enter the path where you want the plug-in to be installed.
4. Click *Install*. The Genesys Installation Wizard indicates it is performing the requested operation for the Genesys Proactive Engagement Plug-in for GAX. When through, the *Installation Complete* screen appears.

5. Click *Finish* to complete your installation. As a result of the installation, the `gax-webme-plugin-[version].jar` and `wmcategory-[version].jar` files are copied to the `[Genesys Administrator Extension Tomcat server]/webapps/gax/WEB-INF/lib/` folder.
6. In Genesys Administrator, navigate to `Provisioning > Environment > Applications`, select the Genesys Administrator Extension server, and start it.
7. 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 **Proactive Engagement** menu with the following Web Engagement items: Categories and Script Generator.



The Proactive Engagement menu in Genesys Administrator Extension.

End

Next Steps

- Now that you have installed Genesys Web Engagement and its plug-ins, you can refer to Configuring Features to enable GWE features such as the Pacing Algorithm, Authentication, SSL, Chat and Voice Channels, and UTF-8.

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 Frontend and Backend servers.
- 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. To support secure (TLS) connections to Configuration Server on Windows OS, if you use the JDK 6 64 bit, it is mandatory to do *one* of the following:

- Update Java Development Kit 6 64 bit with Java SE Development Kit 6, Update 38 (JDK 6u38) or older.
- Setup JDK 7.

> **Important**
>
> Genesys performs security testing with OWASP Zed Attack Proxy (ZAProxy) to make sure the Genesys Web Engagement solution is invincible to known attacks. For details, see Security Testing with ZAProxy.

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 Backend Server, Interaction Workspace, and the Engagement Strategy.

Next Steps

After you configure security for Genesys Web Engagement, you can configure features to enable additional functionality.

Secure Sockets Layer (SSL)

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 Standalone deployment.

For a Clustering 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.

> **Important**
>
> You must use the Java Development Kit version 1.6.0_29 or higher to support the JSSE keystore.

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.

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

Prerequisites

- You completed Loading an SSL Certificate and Private Key into a JSSE Keystore

Start

1. Open the Jetty SSL configuration file in a text editor: ***jetty_installation*/etc/jetty-ssl.xml**.
2. Find the **<New id="sslContextFactory" class="org.eclipse.jetty.http.ssl.SslContextFactory">** element and update the passwords:

```
<New id="sslContextFactory" class="org.eclipse.jetty.http.ssl.SslContextFactory">
     <Set name="KeyStore"><Property name="jetty.home" default="." />/etc/keystore</Set>
     <Set name="KeyStorePassword">OBF:<obfuscated_keystore_password></Set>
     <Set name="KeyManagerPassword">OBF:<obfuscated_keymanager_password></Set>
     <Set name="TrustStore"><Property name="jetty.home" default="." />/etc/keystore</Set>
     <Set name="TrustStorePassword">OBF:<obfuscated_truststore_password></Set>
</New>
```

Note: You can run Jetty's password utility to obfuscate your passwords. See http://wiki.eclipse.org/Jetty/Howto/Secure_Passwords.

3. 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 **jcert.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)

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 8.1 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 like Configuration Server, Message Server, or Chat Server, see Configuring TLS Parameters in Configuration Manager.

Configuring TLS for Web Engagement Servers

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

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

Configuring TLS Options

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

Option	Default Value	Mandatory	Changes Take Effect	Description
trusted-ca-type	none	no	after restart	Type of trusted storage Valid values: MSCAPI, PEM or JKS. If empty, TLS support is disabled.
trusted-ca	none	no	after restart	Specifies the name of the trusted store file which holds the public certificate to verify the server. Applicable for PEM and JKS trusted storage

Option	Default Value	Mandatory	Changes Take Effect	Description
				types only. Valid values: valid file name (including path)
trusted-ca-pwd	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 e-mail 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 Backend 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 Backend Server application.
4. Click the **Options** tab and navigate to the [security] section.
5. Set the **trusted-ca-type** 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 Backend 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 Backend Server application.
4. Click the **Options** tab and navigate to the [security] section.
5. Set the **trusted-ca-type** 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 Backend Server application host.
7. Set the **trusted-pwd** option to the password defined for your keystore in Step 2.
8. Click **Save & Close**.

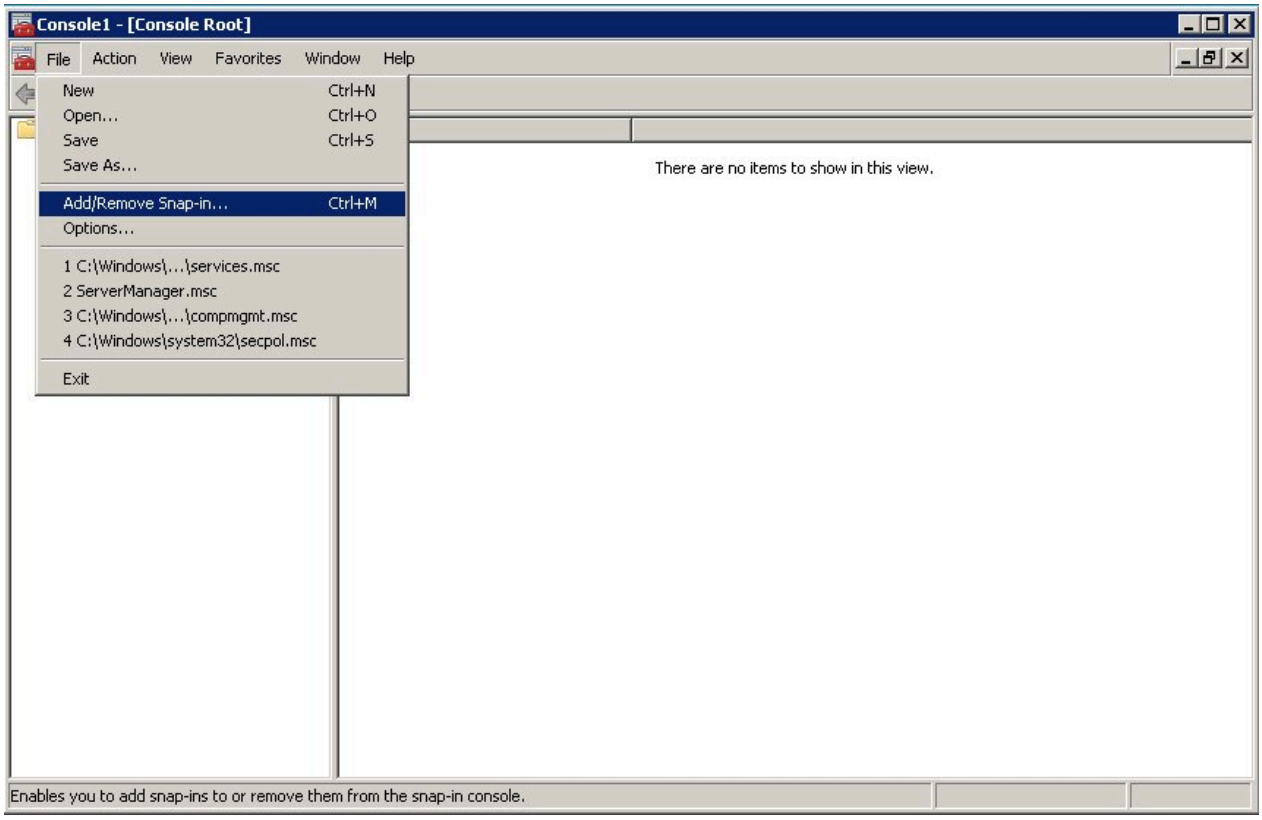
End

MSCAPI Trusted Store

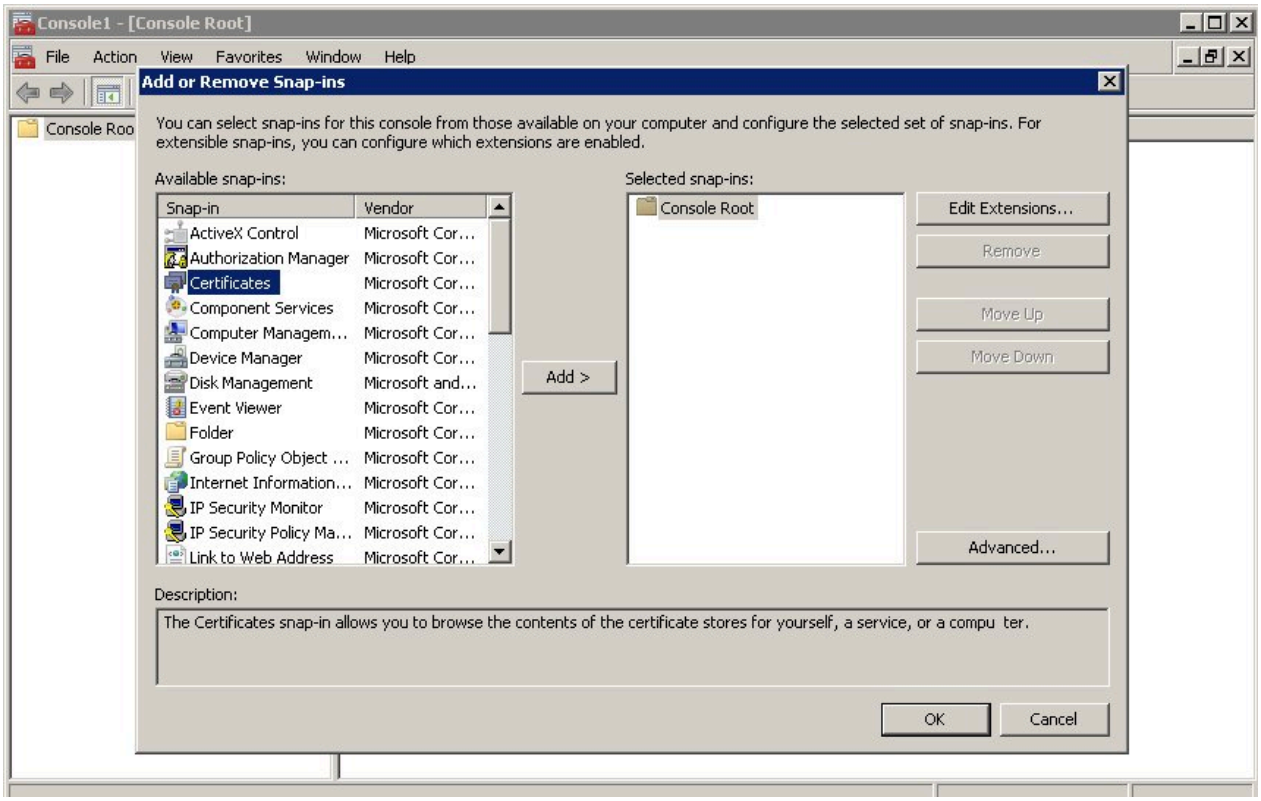
Complete the steps below to work with the MSCAPI certificate trusted store:

Start

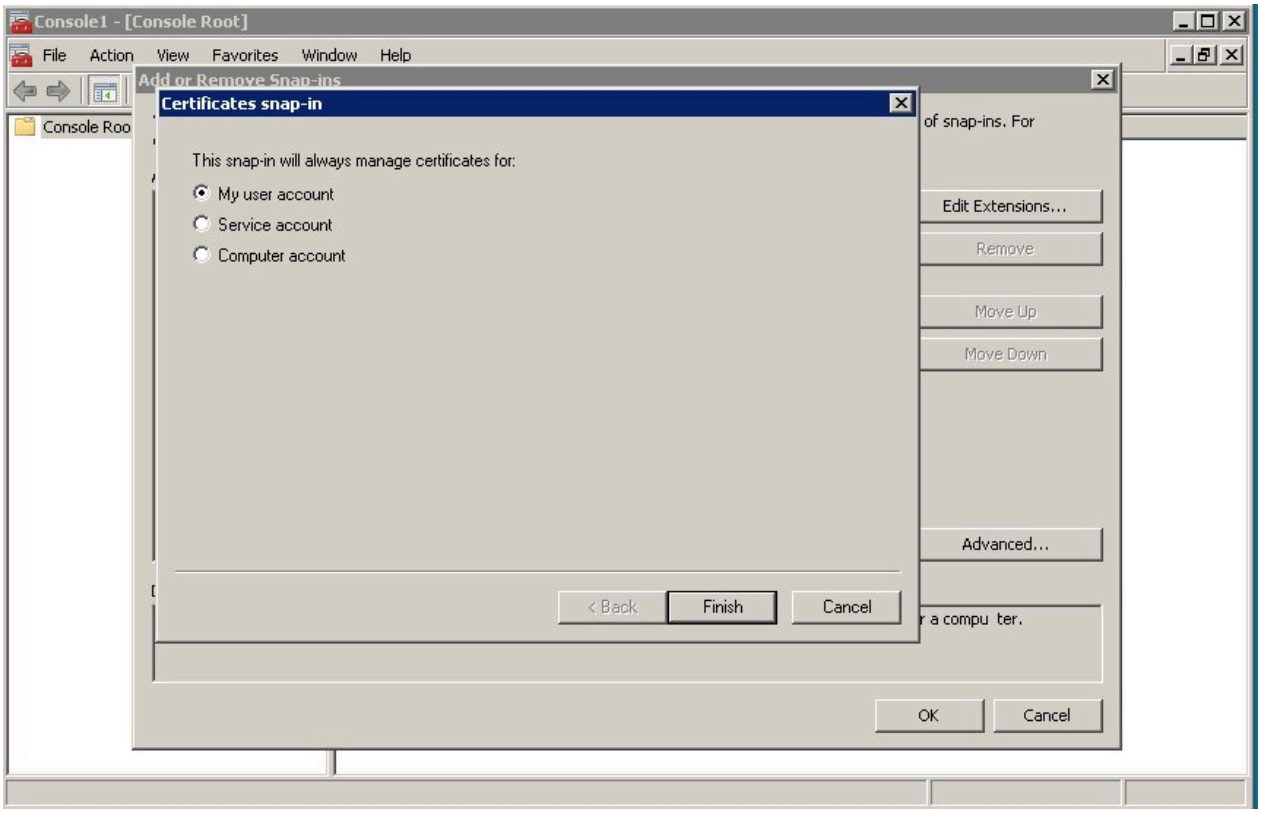
1. Configure and tune TLS for Genesys servers to use certificates signed by the same CA.
2. If the Web Engagement Backend Server is running on a different host, copy the trusted CA certificate to this host.
3. Import the CA certificate to WCS via Certificates Snap-in on the Web Engagement Backend Server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



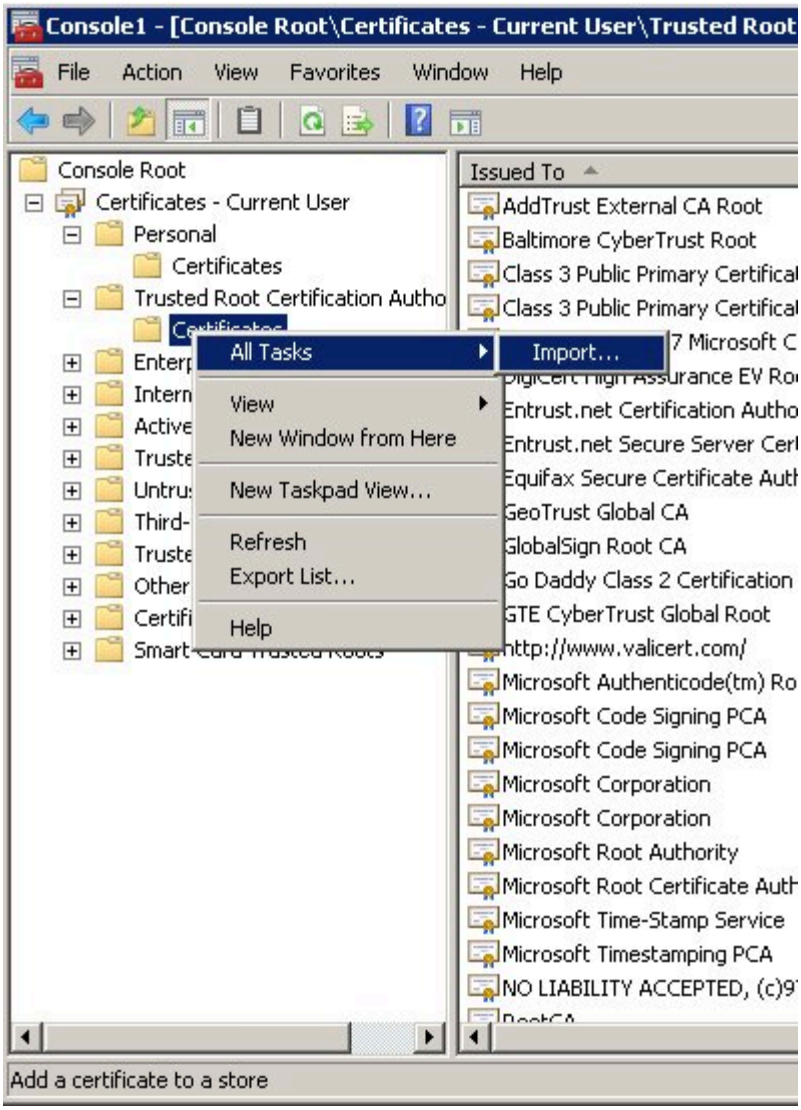
5. Select **Certificates** from the list of available snap-ins and click **Add**.



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. Oonce finished the imported certificate appears in the certificates list.



9. In Genesys Administrator, navigate to **Provisioning > Environment > Applications** and open your Backend Server application.
10. Click the **Options** tab and navigate to the [security] section.
11. Set the **trusted-ca-type** 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 Backend 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 Backend 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 Backend Server".

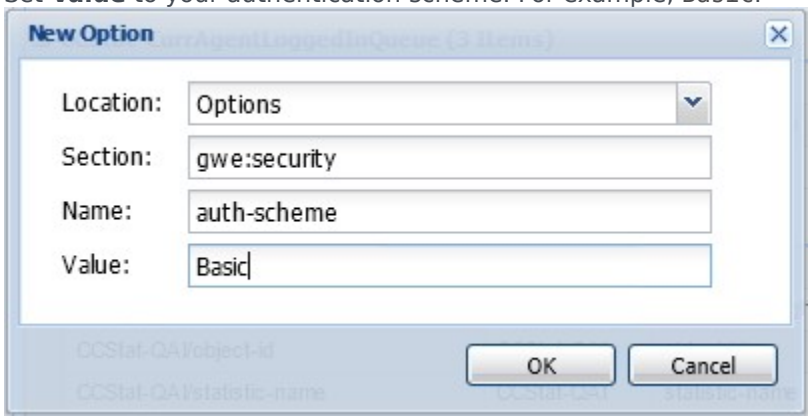
Start

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

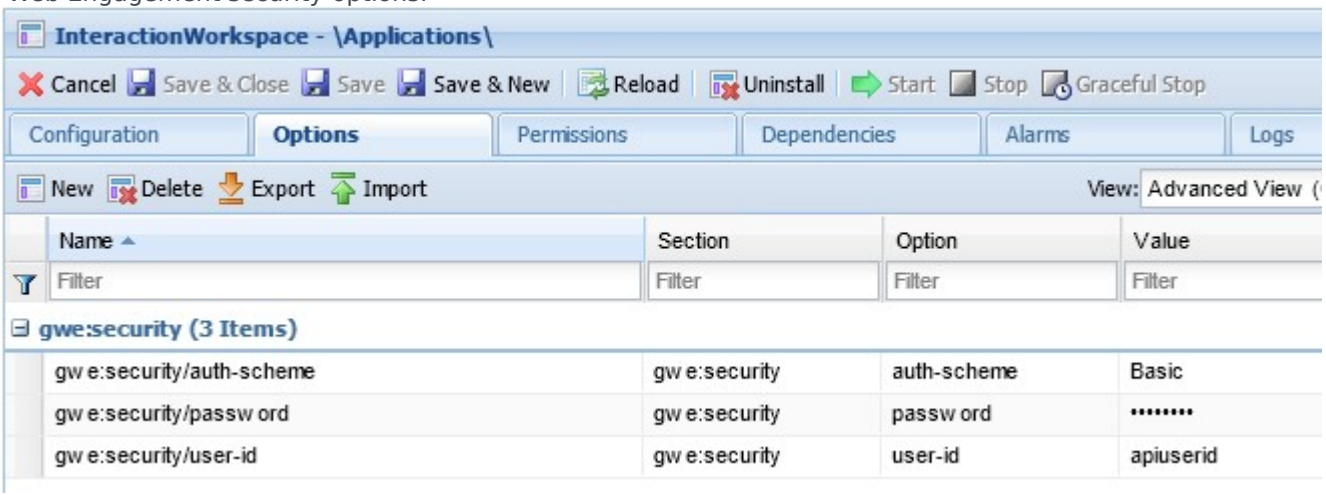
2. Click the options tab and then click **New**.
3. In the **New Option** window, configure the following:
   a. Set **Section** to `gwe:security`
   b. Set **Name** to `auth-scheme`
   c. Set **Value** to your authentication scheme. For example, `Basic`.



   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 Backend Server".
- Your SCXML strategy uses the REST API. See Customizing the Engagement Strategy for details.

Start

1. Go to **\apps*application_name*_composer-project\WebEngagement_EngagementLogic\src-gen** and open the **default.scxml** file in Composer, a text editor, or an XML editor.
2. Find the variables for **user** and **password** and set your authentication credentials. For example:

```
var user = 'user1';
 var password = 'password1';
```

3. Save your changes.

End

Next Steps

- Return to the Genesys Web Engagement Security page.

Features

Genesys Web Engagement includes additional features 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 Backend Server and Chat Server to support a chat channel or specify chat as the default channel of engagement.
- Web Callback Channel — Manually configure the Backend Server to support a web callback channel or specify web callback as the default channel of engagement.
- UTF-8 — Configure a UTF-8 connection between your Frontend and Backend servers and Configuration Server.

Next Steps

After you configure the features you plan to use in Genesys Web Engagement, you can continue following the steps your deployment scenario:

- If you are completing the Standalone deployment, refer to the Developer's Guide to develop a GWE application.
- If you are completing the Clustering deployment, return to Deploying and Configuring the Genesys Web Engagement Cluster.

Pacing Algorithm

Genesys Web Engagement includes an intelligent pacing algorithm that predict 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.

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
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

Complete this procedure to enable the pacing algorithm to predict traffic for your Web Engagement solution.

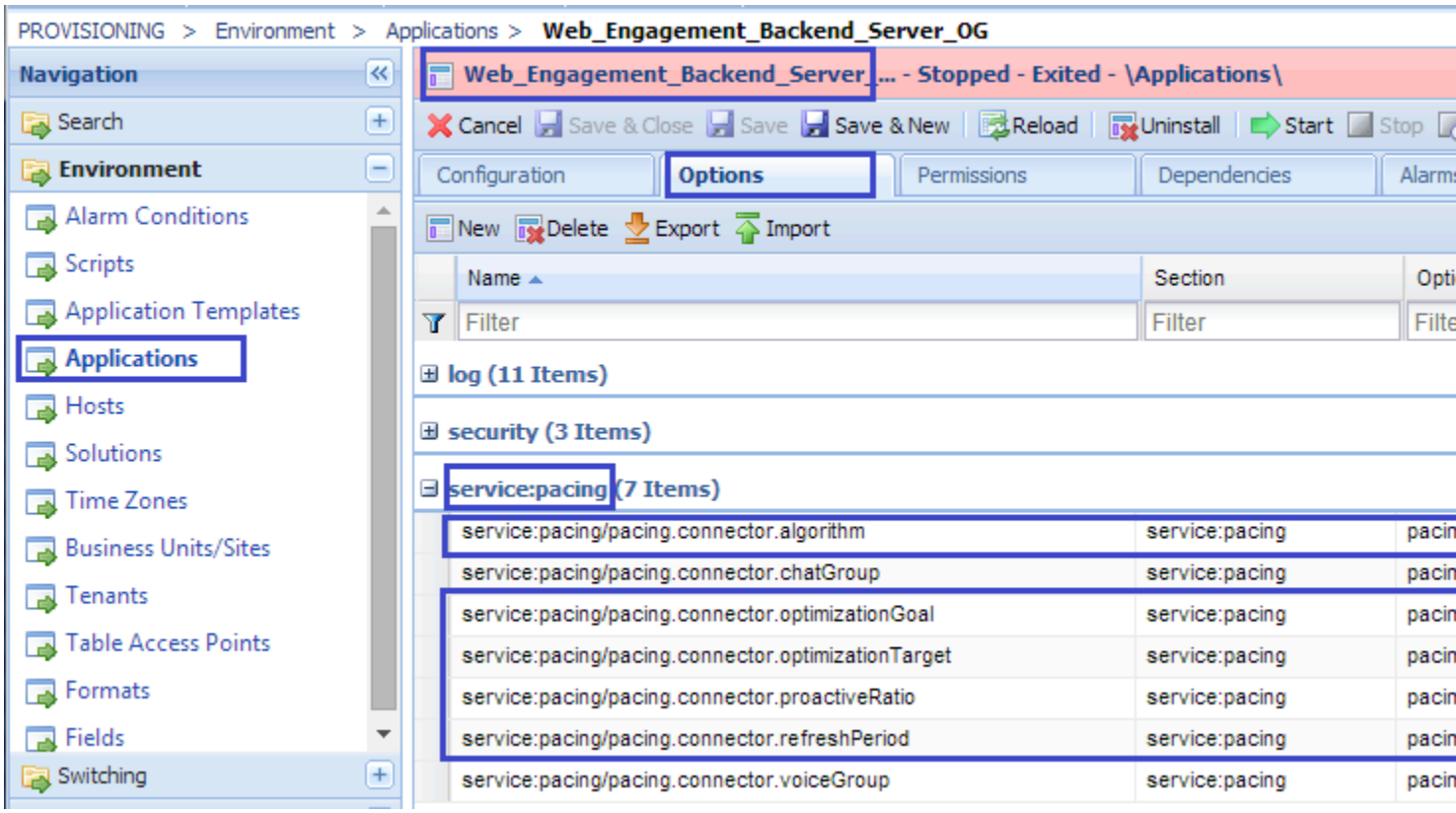
Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**.
2. Edit the Web Engagement Backend Server application and click the **Options** tab.
3. In the [service:pacing] section, set the pacing.connector.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 pacing.connector.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 pacing.connector.optimizationGoal option. The value you set for this option depends on the value you set for **pacing.connector.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 pacing.connector.proactiveRatio option.
   This option basically controls the minimal percentage of agents that will be reserved for processing proactive engagement interactions.

7. Set the pacing.connector.refreshPeriod option. This option controls how frequently the pacing algorithm provides predictions. Genesys recommends that you use values from 1 to 5.



Pacing algorithm settings

End

Next Steps

- Configuring the Agent Groups

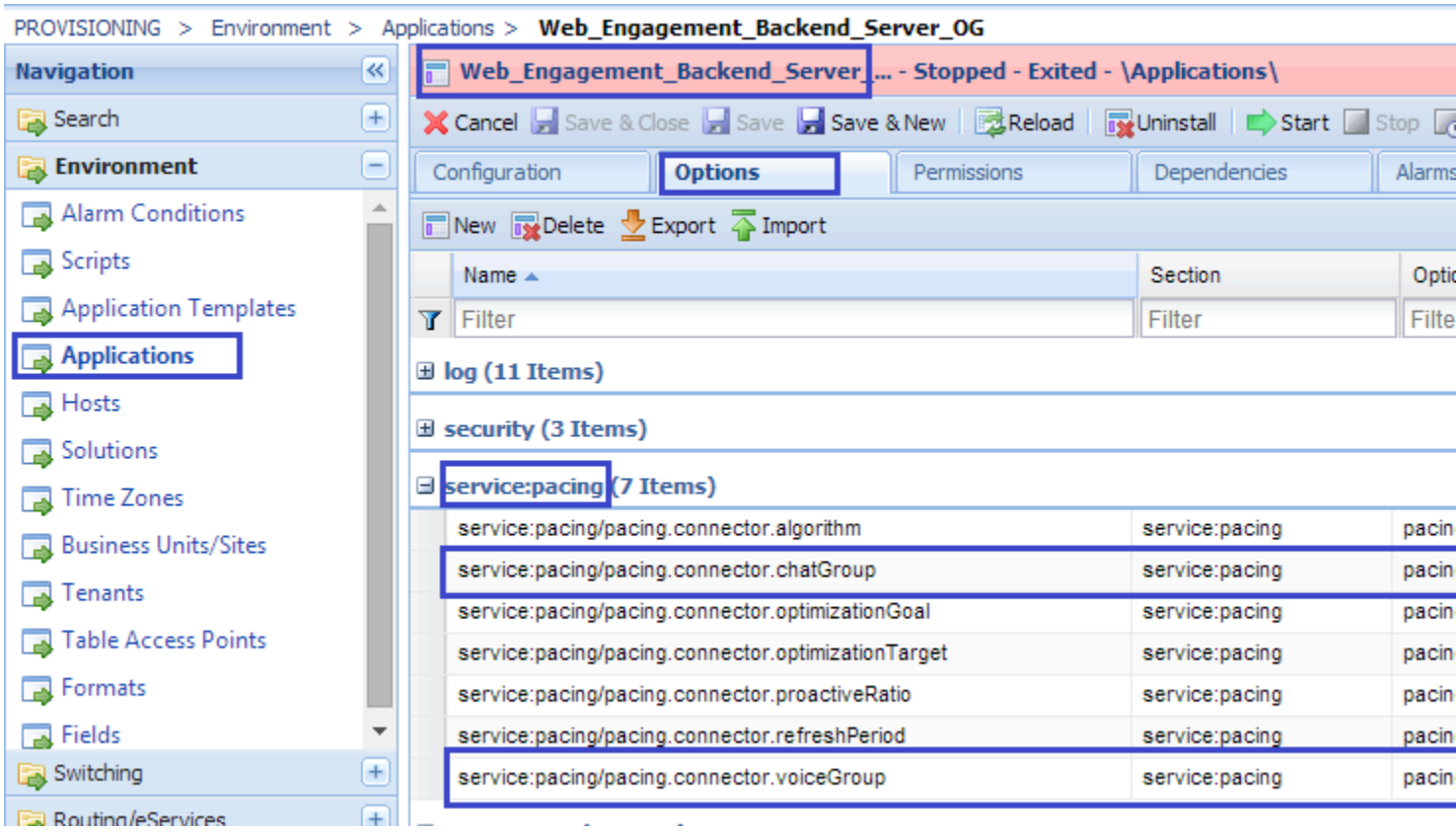
Configuring the Agent Groups

In the previous procedure, you set the **pacing.connector.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.

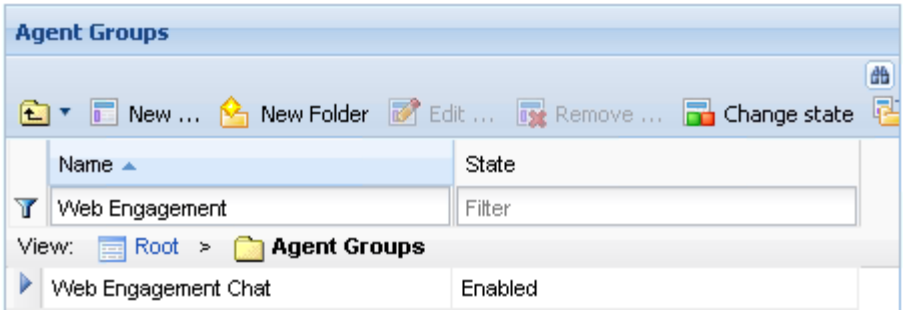


Configure new agent groups

> **Important**
>
> You can use your own groups instead by changing the values of the pacing.connector.chatGroup and pacing.connector.voiceGroup options to the names of other groups you configured.

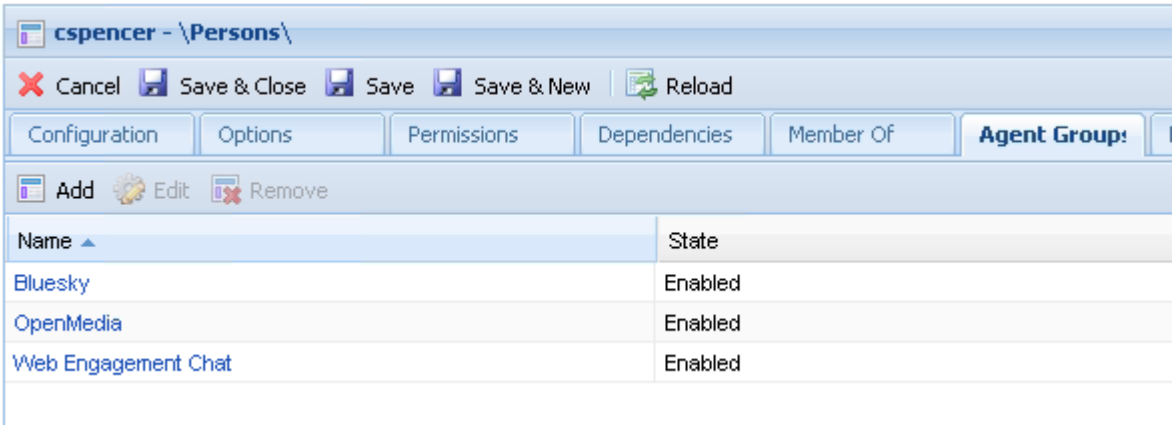
Start

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

Next Steps

- Return to the Genesys Web Engagement Features page.

Chat Channel

When you install Genesys Web Engagement, the Provisioning Tool automatically configures the Backend Server 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 Backend Server 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 wmsg.connector.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.

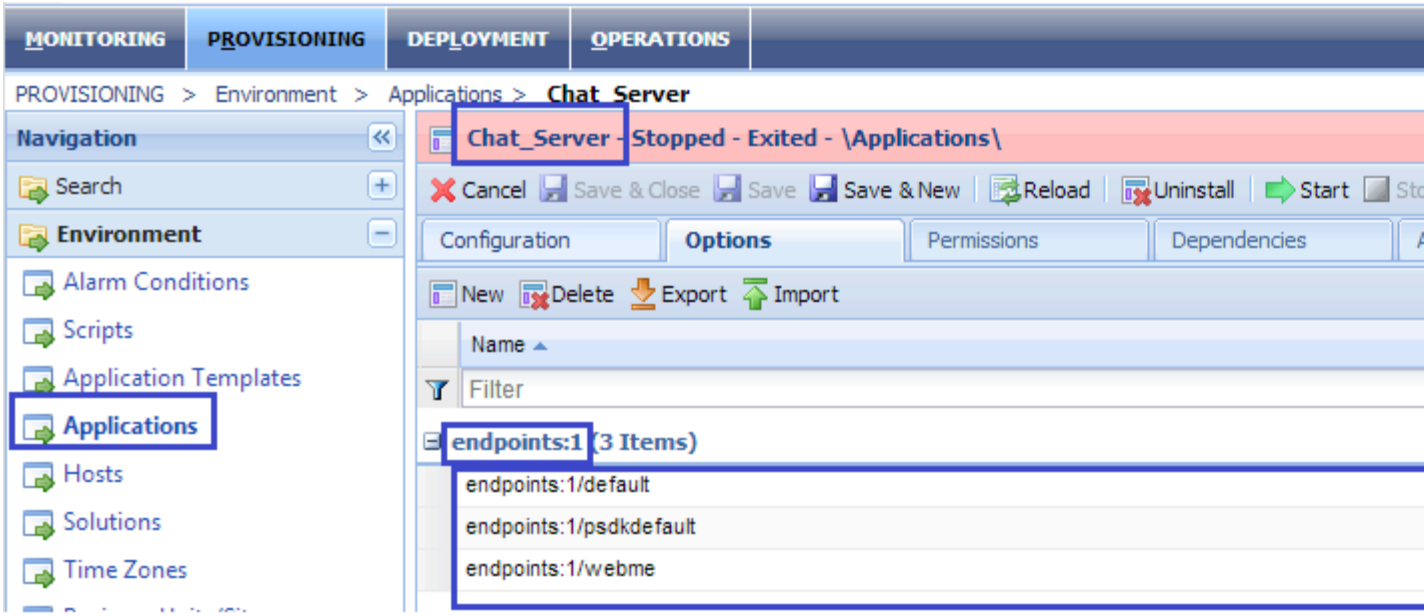
Configuring the Backend Server and Chat Server to Support a Chat Channel

Prerequisites

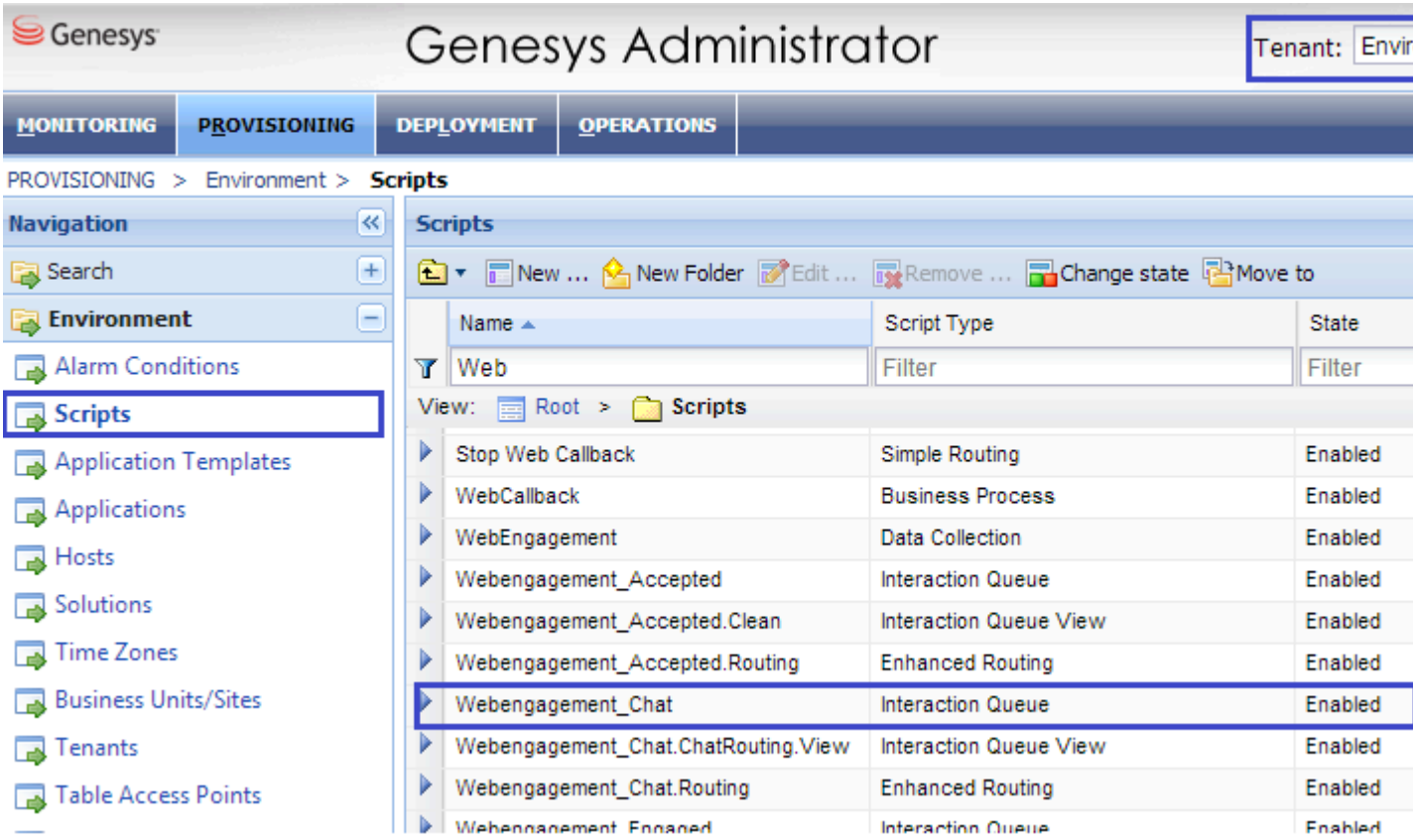
- On your Backend Server application, you have a connection to one of following:
  - Chat Server — See Configuring the Backend Server Application for details.
  - A cluster of Chat Servers — See Configuring a Connection to a Cluster of Chat Servers (Optional) for details.

Start

1. In Genesys Administrator, open your Chat Server application - either the one you connected to directly on the Backend Server, 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.



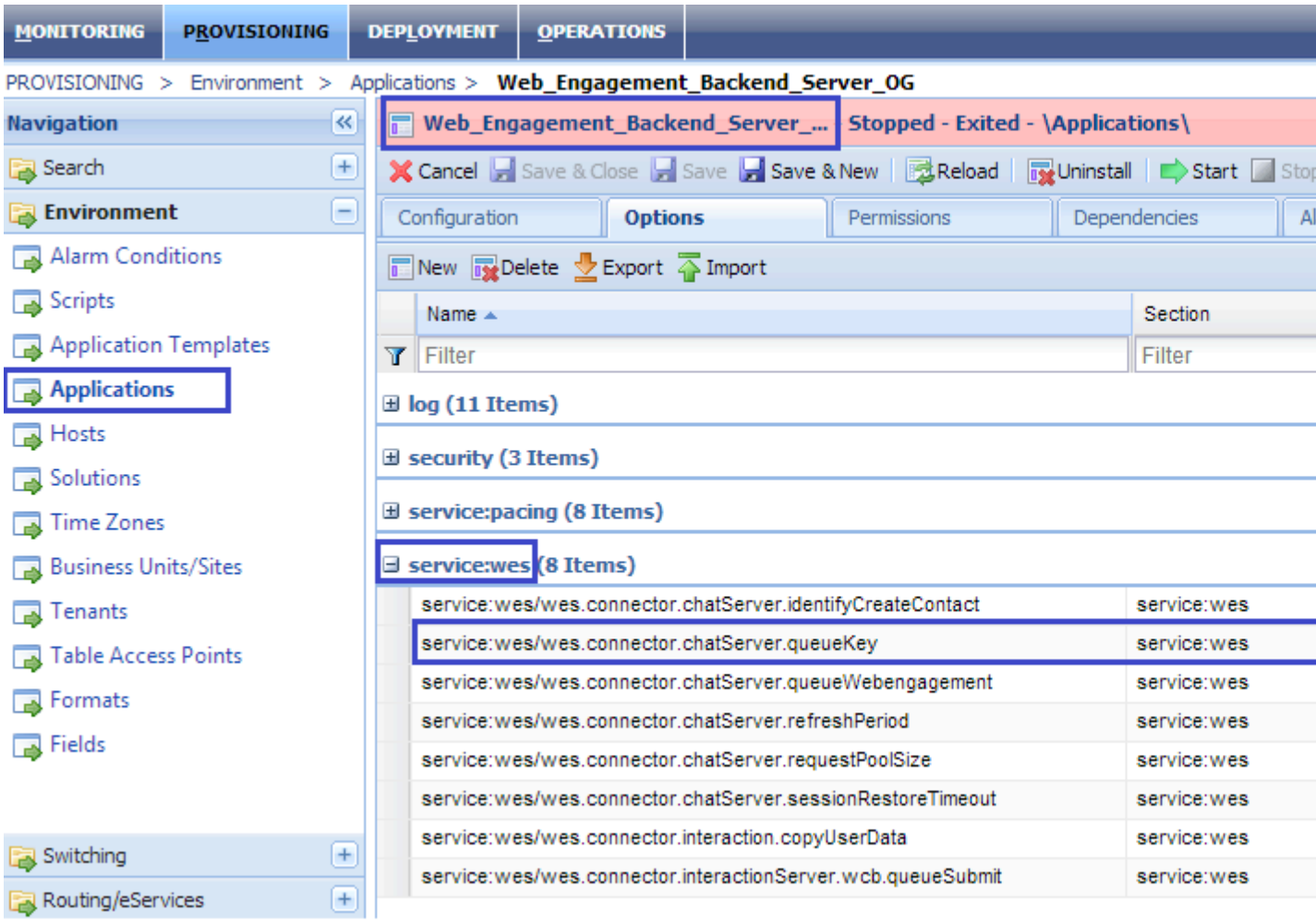
The **webme** option is set to `Webengagement_Chat`



The `Webengagement_Chat` Interaction Queue

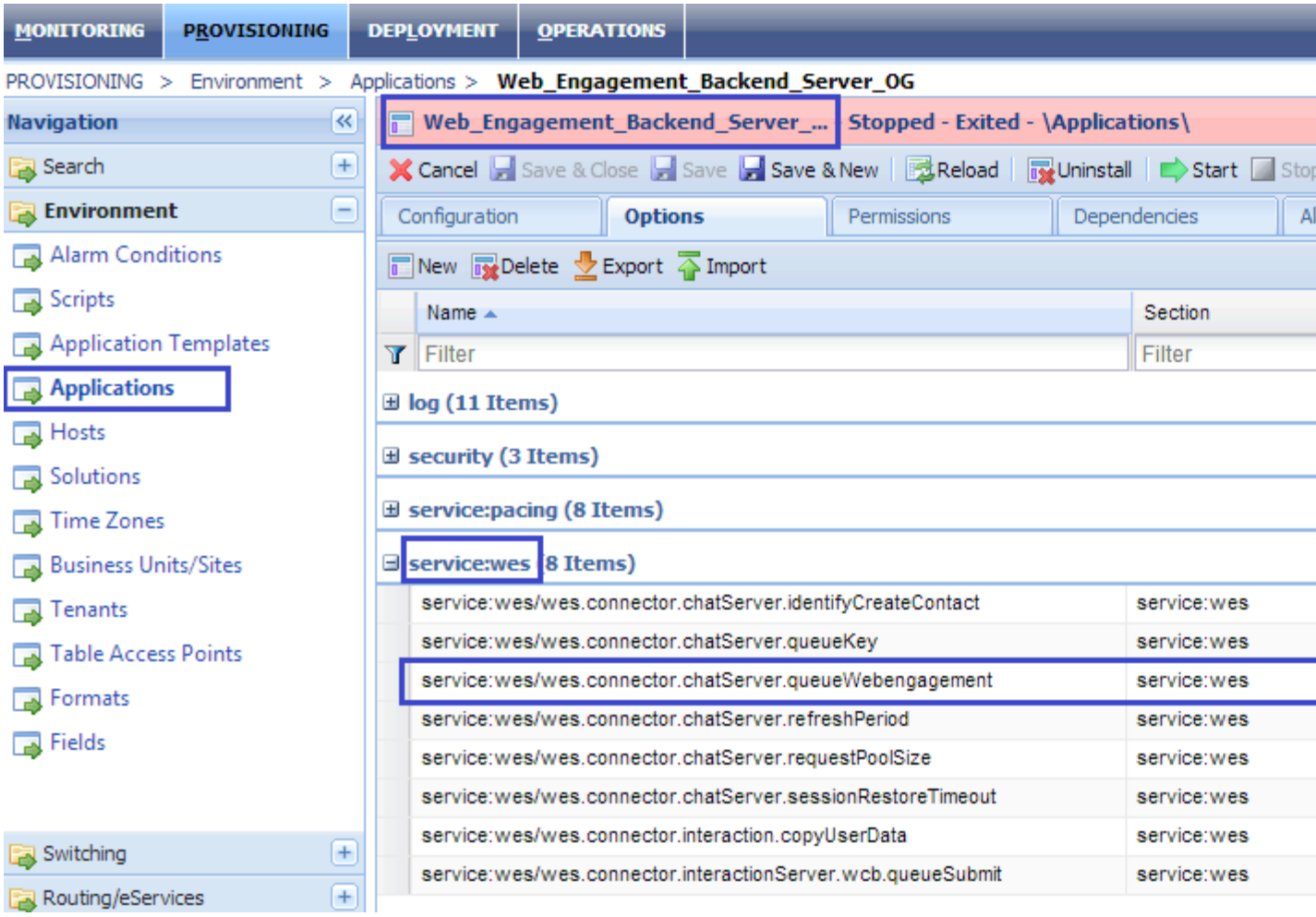
4. Configure the Chat Server endpoint for the Web Engagement Backend Server application by opening

the Backend Server application and select the **Options** tab. In the [service:wes] section, set the value of the wes.connector.chatServer.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*.



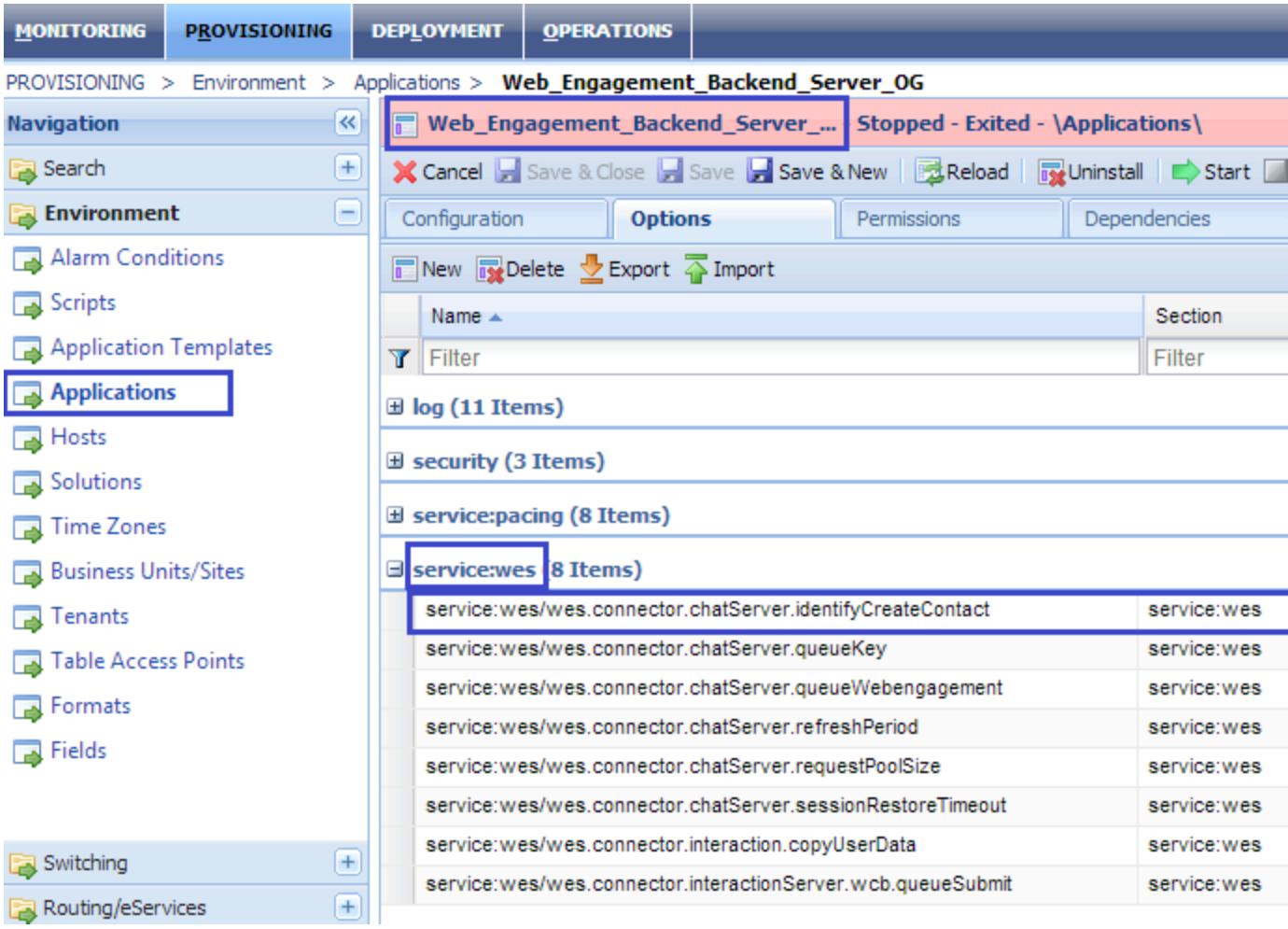
The **wes.connector.chatServer.queueKey** option is set to `1:webme`

5. Specify the Interaction Queue that is used as a starting point to route chat interactions. In the [service:wes] section, set the value of the wes.connector.chatServer.queueWebengagement option to the same queue you specified for the Chat Server endpoint in Step 3.



The **wes.connector.chatServer.queueWebengagement** option is set to `Webengagement_Chat`

6. Configure how contact management will behave when a chat session is instantiated. In the [service:wes] section, set the wes.connector.chatServer.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).

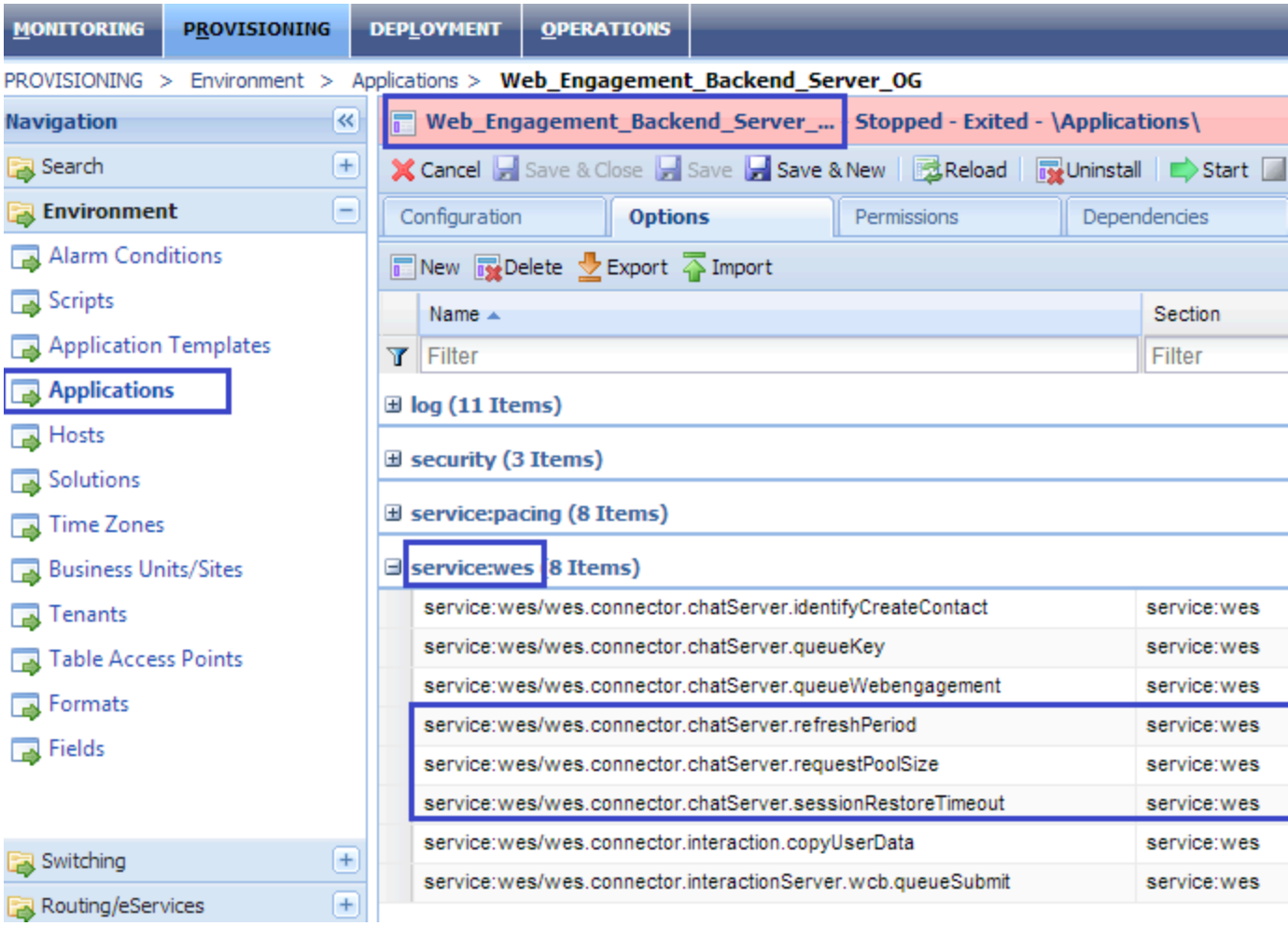


The **wes.connector.chatServer.identifyCreateContact** option is set to 2

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.

1. Configure the chat session behavior by setting the following three options in the [service:wes] section:
   - wes.connector.chatServer.refreshPeriod — Specifies the frequency (in seconds) of chat session updates in the chat widget. The allowed range is from 1 to 5 seconds.
   - wes.connector.chatServer.requestPoolSize — Specifies the count of threads that serve the communication between the chat widgets and Chat Server(s)
   - wes.connector.chatServer.chatSessionRestoreTimeout — 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 `service:wes` 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 wmsg.connector.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 Backend Server.
3. Set the chat channel as the default channel of engagement. In the [service:wmsg] section, set the value

of the **wmsg.connector.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 Backend Server to support working with a voice (web callback) channel.

If you need to, you can configure this manually by completing the "Configuring the Backend 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 wmsg.connector.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.

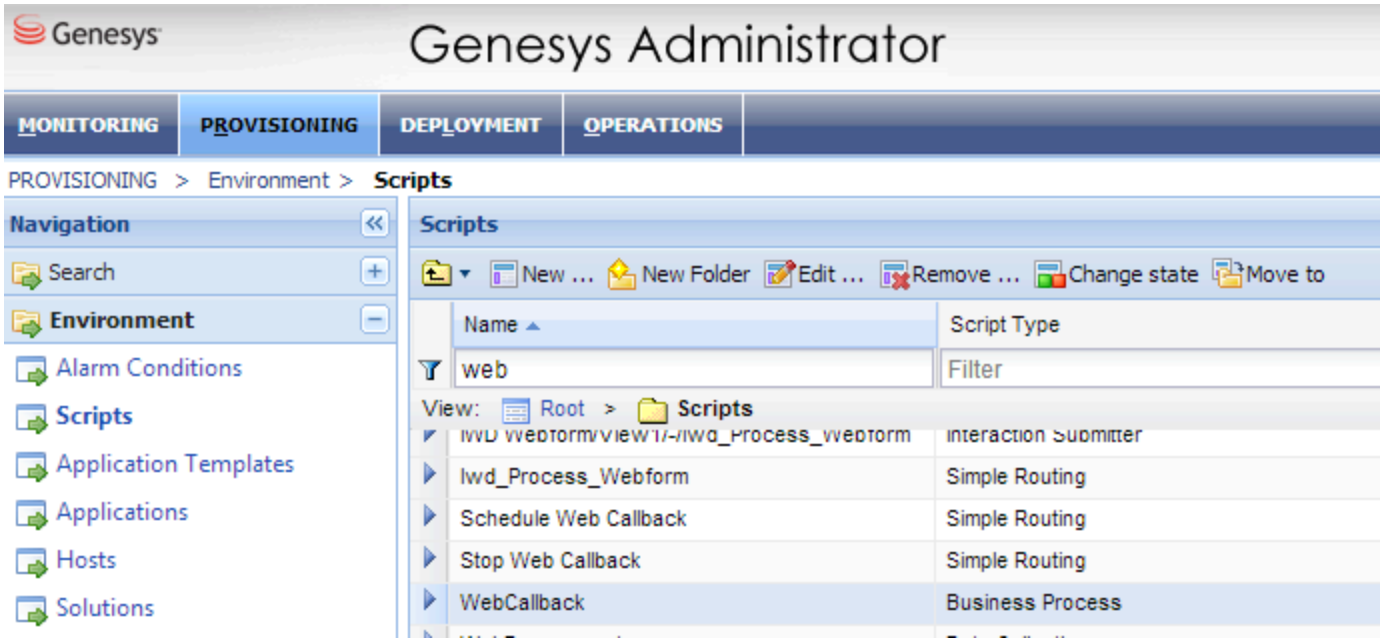
Configuring the Backend Server to Support a Voice (web callback) Channel

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.

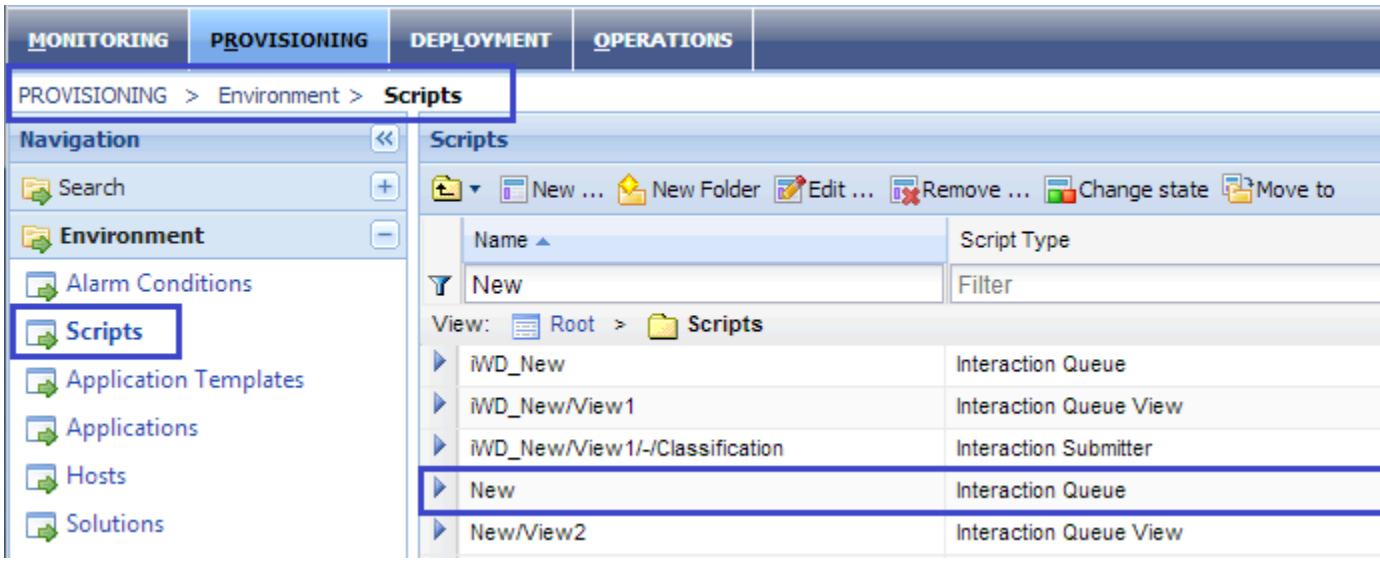
Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Scripts** and confirm the following:
   - The Web Callback business process is installed:



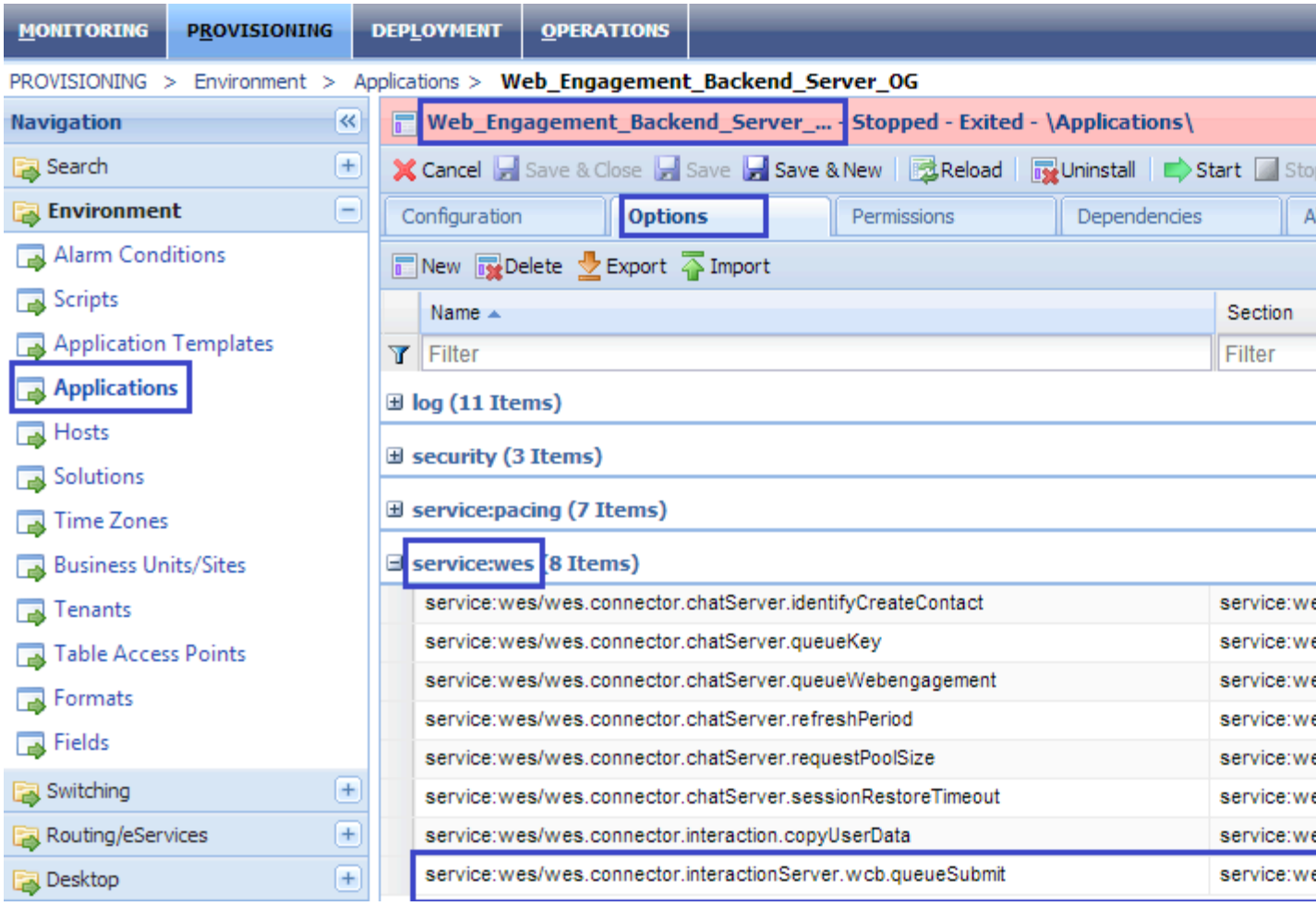
The Web Callback business process

- An Interaction Queue named New exists:



An Interaction Queue named New

2. Navigate to **Provisioning > Environment > Applications** and open the Web Engagement Backend Server application to configure the web callback endpoint.
3. Select the **Options** tab.
4. In the [server:wes] section, set the value of the wes.connector.interactionServer.wcb.queueSubmit option to New.



Set the **wes.connector.interactionServer.wcb.queueSubmit** 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 pacing.connector.voiceGroup 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 wmsg.connector.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 Backend Server.
3. Set the web callback channel as the default channel of engagement. In the [service:wmsg] section, set the value of the **wmsg.connector.defaultEngagementChannel** option to `proactiveVoice`.

End

Next Steps

- Return to the Genesys Web Engagement Features page.

UTF8

You can configure your Frontend and Backend servers to support UTF-8 in Configuration Server, which in turn supports multi-language categories. Although the Backend Server does not read categories from Configuration Server, Genesys recommends that you specify the same settings for both the Backend and Frontend servers - either both servers use UTF-8 support or both servers do not use it.

Configuring a UTF-8 Connection to Configuration Server

Complete the following steps for both your Frontend and Backend server.

Prerequisites

- Your version of Configuration Server supports UTF-8. For details, see the compliant versions for mandatory components.

Start

1. Navigate to the installation directory for your Frontend or Backend Server and open the **setenv.bat** file for Windows — or the **setenv.sh** file for Linux — with a text editor. For example: **C:\WebME\servers\frontend\setenv.bat**.
2. Find the following string: **:: set JAVA_OPTS=%JAVA_OPTS% -Dgenesys.cfgServerUseUtf8=true**.
3. Remove the two colons (**::**) at the start. This converts the string from a comment to a command to use UTF-8. Your string should now look like this: **set JAVA_OPTS=%JAVA_OPTS% -Dgenesys.cfgServerUseUtf8=true**
4. Save your changes.

End Next Steps

- Return to the Genesys Web Engagement Features page.

Deploying and Configuring the Genesys Web Engagement Cluster

Complete the procedures on this page if you are following the Genesys Web Engagement Clustering deployment scenario, which is appropriate for a production environment.

Review the Prerequisites

Before you begin, make sure to complete the following prerequisites:

1. You installed the Web Engagement Frontend and Backend servers on the same host in a test environment.
2. In the test environment, you successfully developed and tested an application for serving your site, *including definitions of categories and rules deployment*.
3. In this procedure, it is assumed that your test and production environments share the same Configuration Management Environment (CME). If this is not the case, you should clone the Frontend Server and Backend Server application objects into your production CME before you begin.
4. You stopped your Frontend Server, Backend Server, and Proxy, if they are running.

Back to top

Setting Up Load Balancer Access for Web Engagement-Specific Config Layer Objects

Start

1. Open Genesys Administrator and navigate to **Provisioning > Environment > Applications**. Select

the application defined for the Web Engagement Backend Server and click **Edit...**

2. Change the host of the application to the host planned for your Backend cluster load balancer. See Load Balancing for more information.
   Note: You may need to create this host object. For details about creating host objects in Genesys Administrator, see the *Configuring Hosts* section in the Management Framework 8.1 Deployment Guide.
3. Change *all* ports for the application to the port for the Backend cluster load balancer.
4. Run the Provisioning Tool using the **-overwrite** parameter. This prepares all the related objects in your Configuration Management Environment (scripts, transactions, and so on) to work with load balancing. See Automatic Provisioning for more information about using the tool.

End

You should use this Backend Server application with changed host and port (to point to the load balancer) as the connection to the Interaction Workspace application. See Installing the Plug-in for Interaction Workspace for details. Back to top

Configuring the Frontend Server Nodes

Complete the steps below for each planned node in your Frontend Server cluster.

Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select your Frontend Server application and click **Edit...**
2. Clone the Frontend Server application by clicking '*Save & New*. Update the following fields:
   - Enter an application name.
   - Remove the connection to the Backend Server.
   - Specify the host and ports (including the secure port, if needed) where the Frontend Server node will run.
   - Select the **Options** tab and configure the following:
     - In the **[log]** section, update the **all** option and provide a distinct file name for this application. For example, `c:\logs\WebEngagement\Web_Engagement_Frontend_Node_1`. This lets you to distinguish between the logs produced by the different nodes in the cluster.
     - In the **[settings]** section, set the value of the **loadbalancer** option to
       schema`://`*BackendLoadBalancerHost*`:`*BackendLoadBalancerPort*`/backend`
       Where:
       - *schema* — http or https
       - *BackendLoadBalancerHost* — The FQDN or IP address of the host for the Backend cluster load balancer. This should be the value set in step 2 of the Update the Backend Server Application procedure.
       - *BackendLoadBalancerPort* — The port of the Backend cluster load balancer.
3. Copy the ***GWE_Installation_Home*\servers\frontend** directory to the host planned for this node.
4. In the copied folder, change the following value in either **setenv.bat** (for Windows) or **setenv.sh** (for

Linux):

- Set the **APP_NAME** option as the name of the application on this Node.

5. Click **Save & Close**.

End

Back to top

Configuring the Backend Server Nodes

> ### Important
> You should plan to host your Backend Servers in a secure zone, along with your Chat Server(s), in order to protect the data. The Backend 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.

Complete the steps below for each planned node in your Backend Server cluster.

Start

1. In Genesys Administrator, navigate to **Provisioning > Environment > Applications**. Select your Backend Server application and click **Edit...**
2. Clone the Backend Server application by clicking **Save & New**. Update the following fields:
   - Enter an application name.
   - Remove the connection to the Frontend Server.
   - Specify the host and ports (including secure port, if needed) where the Backend Server node will run.
   - Select the **Options** tab and configure the following:
     - In the **[log]** section, update the **all** option and provide a distinct file name for this application. For example, `c:\logs\WebEngagement\Web_Engagement_Backend_Node_1`. This lets you to distinguish between the logs produced by the different nodes in the cluster.
     - In the **[settings]** section, set the value of the **loadbalancer** option to *schema*`://`*FrontendLoadBalancerHost*`:`*FrontendLoadBalancerPort*`/frontend`
       Where:
       - *schema* — http or https
       - *FrontendLoadBalancerHost* — The FQDN or IP of the host for the Frontend load balancer.
       - *FrontendLoadBalancerPort* — The port of the Frontend load balancer.
3. Copy the ***GWE_Installation_Home*\servers\backend** directory to the host for this node.
4. In the copied folder, change the following value in either **setenv.bat** (for Windows) or **setenv.sh** (for Linux):
   - Set the **APP_NAME** option as the name of the application on this Node.

5. In the copied folder, open the **etc\cassandra.yaml** file and confirm that the **data_file_directories**, **commitlog_directory**, and **saved_caches_directory** parameters are correct. See Configuring Cassandra for the Cluster for details.

End Back to top

Configuring Load Balancing

Genesys Web Engagement allows you to implement load balancing for your Backend Server and Frontend Server clusters using any third-party load balancer that supports cookies and encoding-based routing methods.

For load balancing details and a sample configuration for Apache, see Load Balancing. Back to top

Configuring Cassandra for the Cluster

Genesys Web Engagement version 8.1.2 includes an embedded Cassandra database (version 1.2.15). When you create a cluster of Backend Servers you are also creating a cluster of embedded Cassandra databases that need to be configured to work together. For information about hardware considerations for Cassandra nodes, refer to the Cassandra documentation: http://www.datastax.com/documentation/cassandra/1.2/cassandra/architecture/architecturePlanningHardware_c.html and the Sizing information for GWE.

Complete the steps below on each of your Backend Servers to configure the embedded Cassandra nodes.

Prerequisites

- You installed the Backend Server on each node. See Configuring the Backend Server Nodes for details.
- You know the IP address of each node in your Backend Server cluster.
- Determine which nodes will be seed nodes. (Cassandra nodes use the seed node list for finding each other and learning the topology of the ring.)
- You completed Generic Cassandra Configuration procedures on each of your nodes.

You can find more information about configuring a Cassandra cluster in the Cassandra documentation: http://www.datastax.com/documentation/cassandra/1.2/cassandra/initialize/initializeSingleDS.html

Start

1. For each Backend Server, open the the **backend/etc/cassandra.yaml** file with a text editor.
2. Edit the specified strings:

```
num_tokens: 256
 listen_address: <backend-node_ip_address>
 rpc_address: 0.0.0.0
```

```
seed_provider:
   - class_name: org.apache.cassandra.locator.SimpleSeedProvider
     parameters:
          - seeds:  "<seed_node1_ip>,<seed_node2_ip>,..."
```

- `<backend-node_ip_address>` — The IP address of the Backend Server.
- `<seed_node1_ip>,<seed_node2_ip>,...` — The IP address of each Backend Server seed node in your cluster.

3. Save your changes.

End Back to top

Enabling SSL for the Cluster

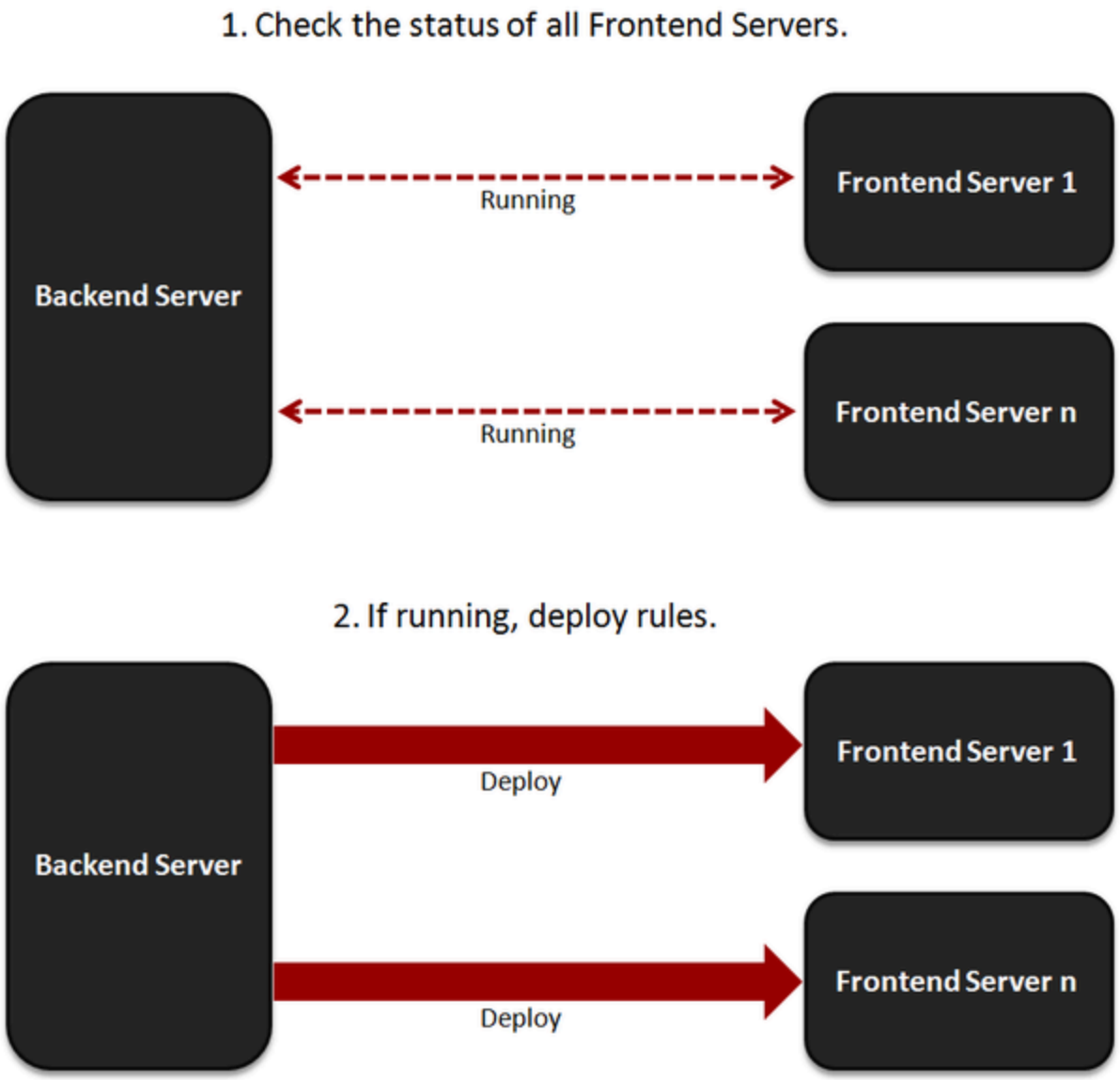
The Jetty web server supplied with the Genesys Web Engagement solution includes a self-signed certificate that you can use for a Standalone deployment. For a Clustering deployment, you should use a certificate issued by a third-party Certificate Authority.

For details, see the procedures to enable SSL. Back to top

Configuring Rules Deployment for the Cluster

Complete this procedure to configure the Rules Authoring Server and one of your Backend Servers to enable rules deployment over the Frontend Server cluster. Rules are deployed in two stages:

1. Genesys Web Engagement detects whether all the Frontend Servers are running. If one is down, Web Engagement stops the rules deployment.
2. The rules are deployed to the Frontend Servers. If one Frontend Server goes down, the deployment is marked as unsuccessful. **Note:** Web Engagement cannot rollback the rules from the already processed Frontend Servers. After an unsuccessful deployment, the CEP rules might have an incorrect state (the rules might be inconsistent between cluster nodes).



Rules deployment stages

Prerequisites

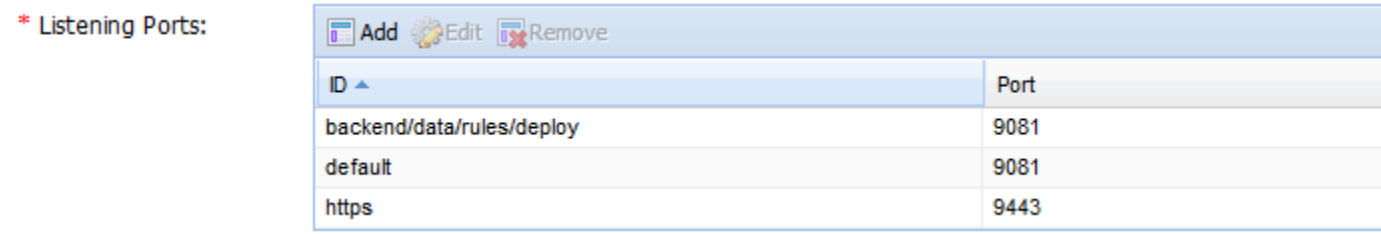
- You must choose one of your Backend Servers (any node in the cluster) to be in charge of rules deployment over the cluster. After this configuration step, this Backend Server will be able to publish rules for all the Frontend Servers of the cluster.

Start

Important

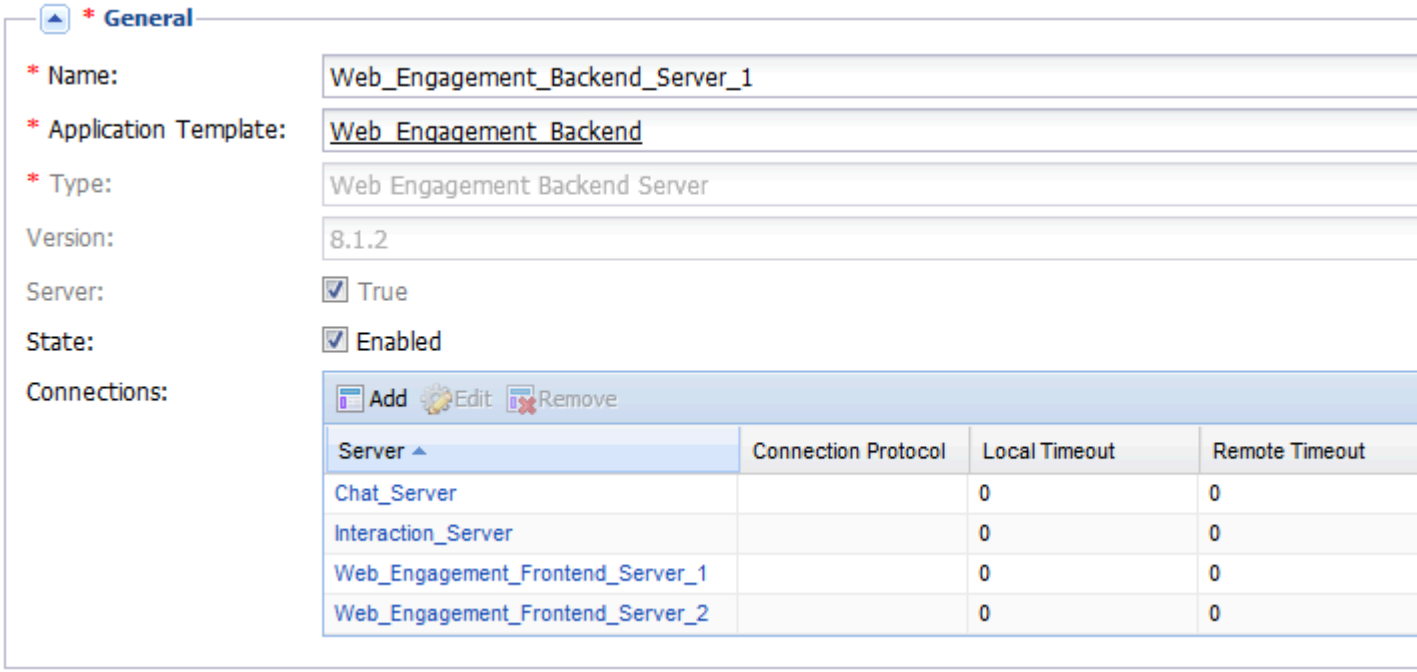
In the following steps, Web_Engagement_Backend_server_1 is used as the Backend Server in charge of rules deployment.

1. In Genesys Administrator, select the Backend Server you want to handle rules deployment and click **Edit...** to check its configuration. Select the **Configuration** tab. Make sure that the listening ports include a port with the following:
   - The **ID** is `backend/data/rules/deploy`
   - The **Protocol** is `http`.



Listening ports

2. If you have more than one Frontend server in your cluster, add all the Frontend Servers to the Connections on the Backend Server responsible for rules deployment.



Connections to all the Frontend Servers

3. Click **Save & Close**.

4. Open your Genesys Rules Authoring Server Application. In the Connections section, add a connection to the Backend Server for rules deployment.
5. Click **Save & Close**.

End

Next Steps

- You can now deploy rules. But first, make sure that the Backend Server in charge of rules deployment and all the Frontend servers are up and running. See Starting the Server Clusters for details.

Back to top

Starting the Server Clusters

Start

1. Start the Frontend load balancer and the Backend load balancer.
2. Start the Backend cluster:
   - Start the Backend Server on the node(s) that is the cluster's seed and wait until it is running.
   - Start the Backend Server on the other nodes of the cluster and wait until they are running.
3. Start the Frontend cluster:
   - Start the Frontend Server on the nodes of the cluster and wait until they are running.

End

Tip

When installing the cluster, there may be problems with Fully Qualified Domain Names (related to re-defining the FQDN to the desired IP). You must check that there are no issues each time you deploy Web Engagement into the cluster.

Categories are updated by Web Engagement servers "on the fly" and no additional configuration is necessary.

To change actual rules in the deployed Web Engagement cluster, see Managing Rules in the User's Guide.

Back to top

Load Balancing

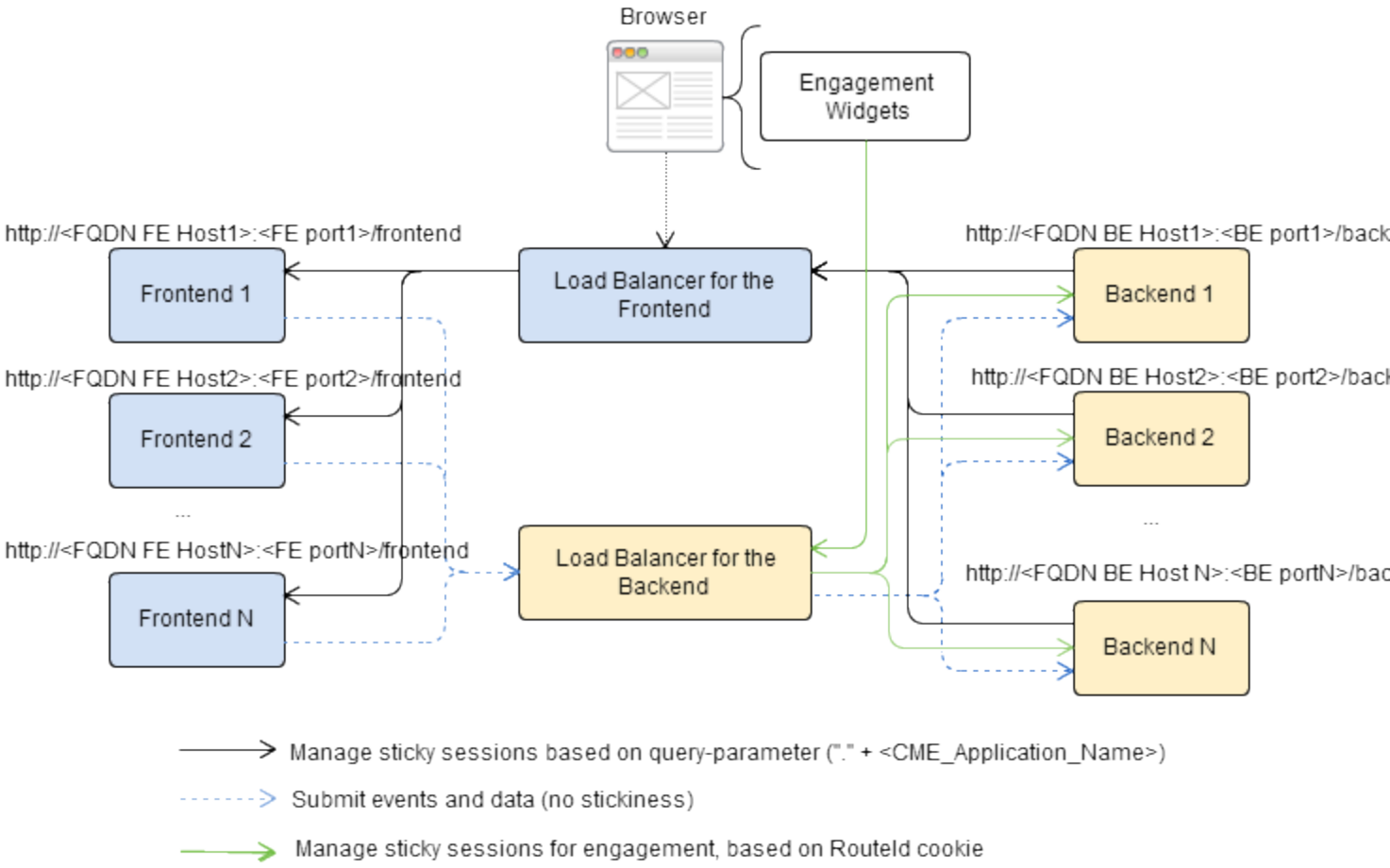
Genesys Web Engagement supports any third-party load balancer as long as the load balancing features include cookie support and URL encoding-based routing methods. You should deploy your load balancers at the latest stage of your Web Engagement deployment.

The following points are important for you to consider when setting up load balancing:

- Frontend and Backend load balancers are mandatory elements for a production deployment. Interconnecting the Frontend and Backend servers directly is only appropriate for a development environment.
- 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 web sockets 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.

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 for the Frontend Server implements sticky sessions for the deployed Web Engagement application based on a query-parameter in this format: `"." + Frontend_Server_Application_Name_In_Genesys_CME`. The Frontend Server is responsible for specifying the related query-parameter.
- The load balancer for the Backend Server implements sticky sessions to route IP addresses when customers are engaged based on the **RouteId** cookie. The load balancer is responsible for specifying this cookie. This example uses **RouteId** as the name of the cookie, but you can use any name.

Sticky Sessions

Genesys Web Engagement uses sticky sessions as follows:

- The load balancer for the Frontend Server implements sticky sessions based on URL encoding static parameters for the deployed Web Engagement application. Web Engagement creates the parameter as follows:
  `"." + Frontend_Server_Application_Name_In_Genesys_CME`.

> **Important**
>
> If you use Apache, you do not need to add "." to your static parameter; this will be done by the Web Engagement Server.

    - The load balancer for the Frontend Server uses this URL parameter instead of a cookie because it also communicates with the Backend Server, which does not have access to cookies.
- The load balancer for the Backend Server implements sticky sessions based on cookies to route IP addresses when the customers are engaged. The load balancer for the Backend Server must support the following features:
    - Cookie-based stickiness to enable engagement.
    - Storage of sticky parameters into cookies.

> **Important**
>
> All the cookies are created by the load balancing system, not by the Genesys Web Engagement servers.

In both cases, GWE requires sticky sessions not only for performance reasons, but also to switch over ongoing transactions if a node (Frontend or Backend) fails.

Pure Cookie-based Stickiness

Since version 8.1.200.39, Genesys Web Engagement supports cookie-based load balancing for both Frontend and Backend servers.

It is possible to implement load balancing for the Frontend Server using cookie-based sticky sessions. In order to do so, the load balancer is required to set the following cookie:

Cookie Name	Cookie Value
FRONTEND_ROUTEID	"." + Frontend_Server_Application_Name_In_Genesys_CME

The load balancer for the Backend Server implements sticky sessions based on cookies to route IP addresses when customers are engaged. The cookie should be set by the load balancer and there are no specific requirements for the cookie's name or value.

> **Tip**
>
> When using *pure* cookie-based stickiness, you can configure a single load balancer to balance frontend and backend traffic.

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 an example 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 Frontend Servers and three Backend Servers configured to work in the cluster (see Deploying and Configuring the Genesys Web Engagement Cluster for details).
- For this configuration example, you installed and configured two instances of Apache, version 2.2 (preferably on different hosts): one to serve the Frontend cluster and another one to serve the Backend cluster.

> **Tip**
> When using *pure* cookie-based stickiness, you can configure a single Apache Load Balancer to balance frontend and backend traffic.

Configuring the Apache Load Balancer for the Frontend Cluster

> **Important**
> Your frontend load balancer must be configured to match this format:
> `http(s)://host:port/frontend`

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
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

3. Add the following configuration script to the end of the **httpd.conf** file:

```
# loadbalancer configuration for GWE Frontend (sticky sessions)
 ProxyPass / balancer://my_cluster/
 <Proxy balancer://my_cluster/>
     BalancerMember [http:// http://]<Node1 IP address>:<listeningPort1> route=NodeName1
     BalancerMember [http:// http://]<Node2 IP address>:<listeningPort2> route=NodeName2
     ...
     BalancerMember [http:// http://]<NodeN IP address>:<listeningPortN> route=NodeNameN
 ProxySet stickysession=alias
 </Proxy>
 #<NodeN IP address> is the IP address of your node, and <listeningPortN>, the associated
 listening port of your frontend application
 #query parameter alias for Frontend cluster sticky session
 #NodeNameN is name of Frontend Server application related to this node
```

4. Save your changes. The load balancer for the Frontend cluster is now configured.
5. Your Frontend Server nodes are healthy if the load balancer receives a successful response on requests to `http(s)://`*`Frontend`* `Server Host:Frontend Server Port (secured port for https)/`*`frontend/about`*

End

Example: Load Balancer Configuration for the Frontend Cluster

```
ProxyRequests Off
<Proxy balancer://mycluster/>
   BalancerMember http://198.51.100.1:8083/frontend route=GWE_Frontend_103
   BalancerMember http://198.51.100.2:8084/frontend route=GWE_Frontend_52
ProxySet stickysession=alias
</Proxy>
ProxyPass /frontend/ balancer://mycluster/
```

Configuring the Apache Load Balancer for the Backend Cluster

> **Important**
>
> Your Backend load balancer must be configured to match this format:
> `http(s)://`*`host`*`:`*`port`*`/backend`

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:

```
# loadbalancer configuration for GWE Backend (cookie-based)
ProxyRequests Off
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
env=BALANCER_ROUTE_CHANGED
<Proxy balancer://mycluster/>
    BalancerMember [http:// http://]<Node1 IP address>:<listeningPort1> route=NodeName1
    BalancerMember http://<Node2 IP address>:<listeningPort2> route=NodeName2
    ...
    BalancerMember http://<NodeN IP address>:<listeningPortN> route=NodeNameN
ProxySet stickysession=ROUTEID
</Proxy>
ProxyPass /backend/ balancer://mycluster/
```

4. Save your changes. The load balancer for the Backend cluster is now configured.
5. Your Backend Server nodes are healthy if the load balancer receives a successful response on requests to `http(s)://`*`Backend`* `Server Host:Backend Server Port (secured port for https)/`*`backend/about`*

End

Example: Load Balancer Configuration for the Backend Cluster

```
ProxyRequests Off
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://mycluster/>
    BalancerMember http://198.51.100.1:9083/backend route=1
    BalancerMember http://198.51.100.2:9084/backend route=2
        ProxySet stickysession=ROUTEID
</Proxy>
ProxyPass /backend/ balancer://mycluster/
```

Enabling Balancer Manager on Your Load Balancers (Optional)

Complete this procedure to enable the Apache balancer manager on your load balancers. This will make the manager available at the **/balancer** URL.

Start

Complete the following steps for the Frontend and Backend Apache load balancers.

1. Edit the **./conf/httpd.conf** file and add the following to the **end** of the file:

```
<Location /balancer>
 SetHandler balancer-manager
 Order Deny,Allow
 #Deny from all # Specify which connections should be denied
 Allow from all # Specify which connections should be allowed
 </Location>
```

2. Save your changes. The balancer manager is now enabled.

End

Configuring a Single Apache Load Balancer for Cookie-based Balancing of the Frontend and Backend Clusters

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 support of Virtual Hosts in the `httpd.conf` file:

```
<VirtualHost *:80>
     ProxyRequests Off
     <Proxy *>
         order allow,deny
         Allow from All
     </Proxy>
     ProxyPass /frontend http://localhost:<frontend server port>/frontend
     ProxyPass /backend http://localhost:<backend server port>/backend
     ProxyPassReverse /frontend http://localhost:<frontend server port>/frontend
     ProxyPassReverse /backend http://localhost:<backend server port>/backend
 </VirtualHost>
 Listen <frontend server load-balancing port>
 Listen <backend server load-balancing port>
```

4. Add a `VirtualHost` for the Frontend servers to the end of the `httpd.conf` file:

```
<VirtualHost *:<frontend server load-balancing port>>
     Header add Set-Cookie "FRONTEND_ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
 env=BALANCER_ROUTE_CHANGED
```

```
    <Proxy balancer://fcluster>
       BalancerMember http://<IP of host with Frontend Server 1>:<Port of Frontend
Server 1>/frontend route=<'.' + Frontend_Server_CME_App_Name>
       ProxySet stickysession=FRONTEND_ROUTEID
    </Proxy>
    ProxyPass /frontend balancer://fcluster
    <Location /balancer-manager>
        SetHandler balancer-manager
        Order Deny,Allow
        Allow from all
     </Location>
</VirtualHost>
```

5. Add a `VirtualHost` for the Backend servers to the end of the `httpd.conf` file:

```
<VirtualHost *:<backend server load-balancing port>>
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
env=BALANCER_ROUTE_CHANGED
<Proxy balancer://bcluster/>
    BalancerMember http://<IP of host with Backend Server 1>:<port Backend Server
1>>/backend route=1
    ProxySet stickysession=ROUTEID
</Proxy>
ProxyPass /backend/ balancer://bcluster/
ProxyPassReverse /backend/ balancer://bcluster/
<Location /balancer-manager>
SetHandler balancer-manager
Order Deny,Allow
Allow from all
</Location>
</VirtualHost>
```

6. Save your changes. You have now configured a single load balancer for the Backend and Frontend clusters.

End

Example: Single Load Balancer for Backend and Frontend Cluster

```
<VirtualHost *:80>
ProxyRequests Off
<Proxy *>
order allow,deny
Allow from All
</Proxy>
ProxyPass /frontend http://localhost:8081/frontend
ProxyPass /backend http://localhost:9081/backend
ProxyPassReverse /frontend http://localhost:8081/frontend
ProxyPassReverse /backend http://localhost:9081/backend
</VirtualHost>
Listen 8081
Listen 9081
<VirtualHost *:8081>
Header add Set-Cookie "FRONTEND_ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/"
env=BALANCER_ROUTE_CHANGED
<Proxy balancer://fcluster>
   BalancerMember http://192.168.100.1:8081/frontend route=GPE_Frontend_1
   BalancerMember http://192.168.100.2:8082/frontend route=GPE_Frontend_2
   BalancerMember http://192.168.100.3:8082/frontend route=GPE_Frontend_3
ProxySet stickysession=FRONTEND_ROUTEID
</Proxy>
ProxyPass /frontend balancer://fcluster
```

```
<Location /balancer-manager>
SetHandler balancer-manager
Order Deny,Allow
Allow from all
</Location>
</VirtualHost>
<VirtualHost *:9081>
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://bcluster/>
    BalancerMember http://192.168.100.1:9081/backend route=1
    BalancerMember http://192.168.100.2:9082/backend route=2
    BalancerMember http://192.168.100.2:9083/backend route=3
    ProxySet stickysession=ROUTEID
</Proxy>
ProxyPass /backend/ balancer://bcluster/
ProxyPassReverse /backend/ balancer://bcluster/
<Location /balancer-manager>
SetHandler balancer-manager
Order Deny,Allow
Allow from all
</Location>
</VirtualHost>
```

Nginx

The information below includes sample Nginx configurations for both the load blanacer for the Frontend Server cluster and the load balancer for the Backend Server cluster.

Prerequisites

- You already deployed your Web Engagement application into a production (or production-like) environment and have at least two Frontend Servers and three Backend Servers configured to work in the cluster (see Deploying and Configuring the Genesys Web Engagement Cluster for details).
- For this configuration sample, you installed and configured two instances of Nginx (preferably on different hosts): one to serve the Frontend cluster and another one to serve the Backend cluster.

To configure your Nginx load balancers, edit the **./conf/nginx.conf** file and modify the configuration according to the samples provided below for the Frontend and Backend load balancers. For details about the configuration, consult the Nginx documentation.

Configuration Sample: Nginx Load Balancer for the Frontend Cluster

```
events {
    worker_connections  1024;
}
http {
```

```
    include       mime.types;
    default_type  application/octet-stream;
     # to handle longer names of GPE server applications
    map_hash_bucket_size 64;

log_format  main  '$remote_addr - $remote_user [$time_local] "$request" "$arg_alias"';

    access_log  logs/nginx_access.log main;
    error_log logs/nginx_error.log warn;

    upstream http_frontend_cluster {
        server 135.17.36.107:8088 fail_timeout=30s;
        server 135.225.51.237:8088 fail_timeout=30s;
        server 135.17.36.10:8088 fail_timeout=30s;
    }
    map $arg_alias $http_sticky_frontend {
        .GPE_F_n1_107 135.17.36.107:8088;
        .GPE_F_n3_237 135.225.51.237:8088;
        .GPE_F_n2_10 135.17.36.10:8088;
    }
    map $http_upgrade $connection_upgrade {
        default upgrade;
        ''      close;
    }
    server {
        listen 8088;

        location @fallback {
            proxy_pass http://http_frontend_cluster;
        }

        location /frontend {
            # Allow websockets, see http://nginx.org/en/docs/http/websocket.html
            proxy_http_version 1.1;
            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 $test HTTP;
            }
            if ($http_sticky_frontend) {
                set $test "${test}-STICKY";
            }
            if ($test = HTTP-STICKY) {
                proxy_pass http://$http_sticky_frontend$uri?$args;
                break;
            }
            if ($test = HTTP) {
                proxy_pass http://http_frontend_cluster;
                break;
```

```
            }
            return 500 "Misconfiguration";
        }
    }
}
```

Configuration Sample: Nginx Load Balancer for the Backend Cluster

```
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
     # to handle longer names of GPE server applications
    map_hash_bucket_size 64;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" "$arg_alias"
"$cookie_routeid"';

    access_log  logs/nginx_access.log main;
    error_log logs/nginx_error.log warn;

    upstream http_backend_cluster {
        server 135.17.36.107:9088 fail_timeout=5s;
        server 135.225.51.237:9088 fail_timeout=5s;
        server 135.17.36.10:9088 fail_timeout=5s;

    }
    map $cookie_ROUTEID $http_sticky_backend {
        default 0;
        .1 135.17.36.107:9088;
        .2 135.225.51.237:9088;
        .3 135.17.36.10:9088;
    }
    map $upstream_addr $serverid {
        135.17.36.107:9088 .1;
        135.225.51.237:9088 .2;
        135.17.36.10:9088 .3;
    }
    map $http_upgrade $connection_upgrade {
        default upgrade;
        ''      close;
    }
    server {
        listen 9088;

        location @fallback {
            proxy_pass http://http_backend_cluster;
        }
        location /backend {
            # Allow websockets, see http://nginx.org/en/docs/http/websocket.html
            proxy_http_version 1.1;
            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 $test HTTP;
            }
            if ($http_sticky_backend) {
                set $test "${test}-STICKY";
            }
            if ($test = HTTP-STICKY) {
                proxy_pass http://$http_sticky_backend$uri?$args;
                break;
            }
            if ($test = HTTP) {
            add_header Set-Cookie "ROUTEID=$serverid;Max-Age=31536000;";
                proxy_pass http://http_backend_cluster;
                break;
            }
            return 500 "Misconfiguration";
        }
    }
}
```

Next Steps

- If you are completing the Clustering deployment scenario, you can return to Configuring Load Balancing.

Configuration Options

Web Engagement Backend Server

The Backend Server has the following configuration option sections:

Section	Options
log *Configure the logs generated by the Backend Server.*	all standard trace verbose segment expire affectedLoggers time_format time_convert
service:wes *Settings for the Backend Server Engagement component.*	wes.connector.chatServer.requestPoolSize wes.connector.chatServer.queueKey wes.connector.chatServer.queueWebengagement wes.connector.chatServer.identifyCreateContact wes.connector.chatServer.refreshPeriod wes.connector.chatServer.requestTimeout wes.connector.chatServer.chatSessionRestoreTimeout wes.connector.interaction.copyUserData wes.connector.interactionServer.wcb.queueSubmit
service:wmsg *Settings for the Genesys Web Engagement Service Gateway.*	wmsg.connector.engagementExpirationTime wmsg.connector.registrationFormExpirationTime wmsg.connector.interactionServer.autoStopWebengagementIntr wmsg.connector.interactionServer.queueAccepted wmsg.connector.interactionServer.queueEngaged wmsg.connector.interactionServer.queueFailed wmsg.connector.interactionServer.queueQualified wmsg.connector.interactionServer.queueRejected wmsg.connector.interactionServer.queueTimeout wmsg.connector.defaultEngagementChannel wmsg.connector.userIdentifier wmsg.connector.phoneNumber
service:wmdb *Settings for the Cassandra database that is embedded in the Backend Server.*	wmdb.retention.entity.all wmdb.retention.entity.<object> wmdb.retention.time-unit
service:pacing *Parameters for the pacing algorithm*	pacing.connector.algorithm pacing.connector.optimizationGoal pacing.connector.optimizationTarget pacing.connector.proactiveRatio pacing.connector.chatGroup pacing.connector.voiceGroup pacing.connector.refreshPeriod

Section	Options
settings *General settings for the Backend Server.*	event-mode loadbalancer
security *Settings to enable security for accessing the Backend Server.*	auth-scheme user-id password trusted-ca-type trusted-ca trusted-ca-pwd

Web Engagement Frontend Server

The Frontend Server has the following configuration option sections:

Section	Options
log *Configure the logs generated by the Frontend Server.*	all standard trace verbose segment expire affectedLoggers time_format time_convert
settings *General settings for the Frontend Server.*	loadbalancer
service:cep *Settings for Complex Event Processing (CEP).*	cep.domainSeparation

Backend Server log Section

all

Default Value: `stdout`
Valid Values (log output types):

`stdout`	Log events are sent to the Standard output (`stdout`).
`stderr`	Log events are sent to the Standard error output (`stderr`).
`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.Setting the `all` log level option to the `network` output enables an application to send log events of the `Standard`, `Interaction`, and `Trace` levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database.
`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: After start/restart

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
```

standard

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

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`

trace

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

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`

verbose

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

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.

segment

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

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.

expire

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`.
`<number> day`	Sets the maximum number of days before log files are deleted. Specify a number from `1`—`100`.

Changes Take Effect: After restart

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

> **Warning**
> If an option's value is set incorrectly — out of the range of valid values — it will be automatically reset to `10`.

affectedLoggers

Default Value:
Valid Values: The names of loggers, separated by a semicolon (;), specified in the `LOG4J2.xml`. For example: `com.genesyslab.webme.commons;PROTOCOL;org.apache.cassandra`
Changes Take Effect: Immediately
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.

time_format

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
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
```

time_convert

Default Value: `local`
Valid Values:

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

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

Backend Server service:wes Section

wes.connector.chatServer.requestPoolSize

Default Value: 10

Valid Values: 1 – 99

Changes Take Effect: After start/restart

Description: Specifies the number of threads used for processing requests to Chat Server. Use the following formula to calculate the recommended value: <peak_chat_sessions_count> / (<refreshPeriod> * 5) <refreshPeriod> - The value of the wes.connector.chatServer.refreshPeriod option. <peak_chat_sessions_count> - The maximum number of simultaneous chat sessions planned for one Backend Server.

wes.connector.chatServer.queueKey

Default Value: 1:webme

Valid Values: Key defined in the endpoints:<tenantID> section of the Chat Server application.

Changes Take Effect: Immediately

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.

wes.connector.chatServer.queueWebengagement

Default Value: WebEngagement_Chat

Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The minus, dot, and underscore characters. Space characters are allowed inside the string only (not at the beginning or at the end).

Changes Take Effect: After restart

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 in the key set in the wes.connector.chatServer.queueKey option.

wes.connector.chatServer.identifyCreateContact

Default Value: 3

Valid Values: 1, 2 or 3

Changes Take Effect: Immediately

Description: Controls whether a contact for a created chat interaction should be identified and/or

created in the Universal Contact Server database (transferred as attached data to Chat Server, which is responsible for details of processing). The valid values have the following effect:

- 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

The default value is applied if the option is absent or specified incorrectly.

wes.connector.chatServer.refreshPeriod

Default Value: 2

Valid Values: `1 – 5`

Changes Take Effect: After start/restart

Description: Specifies how often, in seconds, to refresh the chat content in the chat widget.

wes.connector.chatServer.requestTimeout

Default Value: 5

Valid Values: `1 – 10`

Changes Take Effect: After start/restart

Description: Specifies the timeout, in seconds, for requests to Chat Server. The default value is used if the option is absent or specified incorrectly.

wes.connector.chatServer.chatSessionRestoreTimeout

Default Value: `30`

Valid Values: `5 – 300`

Changes Take Effect: After start/restart

Description: Specifies the time, in seconds, during which the Backend Server tries to restore a broken chat session. A session can be broken, for example, if Chat Server crashes.

wes.connector.interaction.copyUserData

Default Value: `rule;attempt_number`

Valid Values: `all, no, <key_name1;key_name2;...key_nameN>`

Changes Take Effect: Immediately

Description: Specifies the mode Web Engagement uses to copy UserData from the Open Media webengagement interaction to the chat or webcallback engagement interaction. The following explains the results for each valid value:

- `all` — All keys are copied.
- `<key_name1;key_name2;...key_nameN>` — Only the keys in the list are copied.
- `no` — No UserData is copied
- Option is omitted, or has a blank or empty value — No UserData is copied.

> **Important**
> Trailing or leading spaces are considered part of the key name.

wes.connector.interactionServer.wcb.queueSubmit

Default Value: `New`

Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The minus, dot, and underscore characters. Space characters are allowed inside the string only (not at the beginning or at the end).

Changes Take Effect: Immediately

Description: Specifies the Interaction Queue that is used as the entry point to webcallback routing strategy.

Backend Server service:wmsg Section

wmsg.connector.engagementExpirationTime

Default Value: `30`
Valid Values: `10 – 60`
Changes Take Effect: Immediately

Specifies the time, in 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 enagagement invitation is displayed to the visitor.

wmsg.connector.registrationFormExpirationTime

Default Value: `120`
Valid Values: `10—1800`
Changes Take Effect: Immediately

Specifies the time, in seconds, during which the registration form is considered to be valid when waiting for the form to be completed by the visitor.

If the registration form is not completed before the expiration time, the engagement attempt is considered invalid. **Note:** Even after this period, the visitor is able to trigger a chat or voice interaction, but is also able to receive one more engagement invite.

wmsg.connector.interactionServer.autoStopWebengagementIntr

Default Value: `true`
Valid Values: `true`, `false`
Changes Take Effect: Immediately

A Boolean that configures whether or not a webengagement OM interaction is stopped right after being placed in a terminal queue (Accepted, Rejected, Timedout, Failed). If `true`, the interaction will be stopped after being placed in a terminal queue.

> **Important**
>
> The `wmsg.connector.interactionServer.autoStopWebengagementIntr` option was introduced in Genesys Web Engagement 8.1.200.39.

wmsg.connector.interactionServer.queueAccepted

Default Value: `Webengagement_Accepted`
Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The underscore and space characters.
Note: The specified value must be the name of the the Interaction Queue object as provisioned in Configuration Manager. Changes Take Effect: Immediately

A valid name of a queue used for accepted interactions. An interaction is placed in this queue if the visitor accepts the engagement proposal (the disposition code is set to `acceptCall`); Genesys Web Engagement then stops the interaction.

wmsg.connector.interactionServer.queueEngaged

Default Value: `Webengagement_Engaged`
Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The underscore and space characters.
Changes Take Effect: Immediately

A valid name of a queue used for engagement interactions. An interaction is placed in this queue after a positive engagement decision is made in the Engagement Logic SCXML strategy. The interaction usually is not stopped until it is located in this queue.

wmsg.connector.interactionServer.queueFailed

Default Value: `Webengagement_Failed`
Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The underscore and space characters.
Changes Take Effect: Immediately

A valid name of a queue used for failed engagement interactions. An interaction is placed in this queue in three cases:

- If its disposition code is set to `timeout`.
- If the interaction was cleaned from another queue.
- If the Orchestration Server strategy notified the Backend Server that the interaction should not be processed to the real engagement.

Web Engagement Backend Server stops the interaction as soon as it is placed in the Failed queue.

wmsg.connector.interactionServer.queueQualified

Default Value: `Webengagement_Qualified`
Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The underscore and space characters.
Changes Take Effect: Immediately

A valid name of a queue used for creating engagement interactions. Each time Genesys Web Engagement creates an interaction, the new interaction is added to this queue. The interaction is not stopped until it is located in this queue.

wmsg.connector.interactionServer.queueRejected

Default Value: `Webengagement_Rejected`
Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The underscore and space
Changes Take Effect: Immediately

A valid name of a queue used for rejected engagement interactions. An interaction is placed in this queue if the visitor rejects the engagement proposal (the disposition code is set to `reject`); Genesys Web Engagement then stops the interaction.

wmsg.connector.interactionServer.queueTimeout

Default Value: `Webengagement_Timeout`

Valid Values: Valid Values: Letters A to Z and a to z. Numbers 0 through 9. The underscore and space

Changes Take Effect: Immediately

Description: Specifies the name of the Interaction Queue where the webengagement iteraction is placed after a visitor's browser sends the Timeout disposition code for the engagement invitation.

wmsg.connector.defaultEngagementChannel

Default Value: N/A
Valid Values: `proactiveChat, proactiveCallback`
Changes Take Effect: Immediately

A valid name for the default Engagement Channel. When specified, this option turns off detection of an agent's availability by the pacing service. The specified value is selected for engagements by the out-of-the-box Engagement Logic SCXML strategy.

> **Important**
>
> The `wmsg.connector.defaultEngagementChannel` option is intended for development purposes only and should not be used in a production environment.

wmsg.connector.userIdentifier

Default Value: `EmailAddress`

Valid Values: A valid User Data key

Changes Take Effect: Immediately

Description: Specifies the name of the User Data key that is used by Contact Server as the key for the customer identification process. This is the primary key for the Identity object in Genesys Web Engagement.

wmsg.connector.phoneNumber

Default Value: `PhoneNumber`

Valid Values: A valid User Data key

Changes Take Effect: Immediately

Description: Specifies the name of the key that is used by Interaction Workspace to obtain the phone number from the interaction's User Data.

Backend Server service:wmdb Section

The configuration options in the `service:wmdb` section are used to control the settings for the Cassandra database embedded in the Backend Server.

wmdb.limitIndexing

Default Value: `true`
Valid Values: `true`, `false`
Changes Take Effect: After start/restart

This option limits the types of indexing produced by Web Engagement, thereby boosting the performance of the Backend Server and decreasing the amount of data generated by indexing operations. **Note:** when this option is enabled, some of the methods in the History REST API will not work. When this option is disabled, Web Engagement will generate all indexing data.

wmdb.retention.entity.all

Default Value: 1
Valid Values: Any integer
Changes Take Effect: After start/restart

Specifies the time to live (TTL) for all entities in the Cassandra database, in the time units set in the wmdb.retention.time-unit option. The entities affected by this option are: `event, page, session, visit, engagementattempt, useragent,` and `identity`. This option is mandatory.

wmdb.retention.entity.<object>

Default Value: The value of the wmdb.retention.entity.all option.
Valid Values: Any integer
Changes Take Effect: After start/restart

Specifies the time to live (TTL) for the selected entity, in the time units set in the wmdb.retention.time-unit option. Possible values for `<object>` are: `event, page, session, visit, engagementattempt, useragent,` and `identity`. This option is not mandatory.

wmdb.retention.time-unit

Default Value: `day`
Valid Values: `sec, min, hour, day, month`
Changes Take Effect: After start/restart

Defines the time units for the expiration period set in the wmdb.retention.entity.all and wmdb.retention.entity.<object> options. This option is mandatory.

Backend Server service:pacing Section

The configuration options in the `service:pacing` section are used for the pacing algorithm.

pacing.connector.algorithm

Default Value: `SUPER_PROGRESSIVE`

Valid Values:

- `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.

Changes Take Effect: After start/restart

Description: Specifies which type of pacing algorithm should be used by GWE. The type of pacing algorithm determines whether to predict proactive traffic, reactive traffic, or both, for different sized agent groups. If you chose a dual algorithm (`SUPER_PROGRESSIVE_DUAL` or `PREDICTIVE_B_DUAL`), you must specify a value for the pacing.connector.proactiveRatio option.

> **Important**
>
> You can use either algorithm for any size agent group, but it might cause a loss in quality of the predictions if, for example, you use Super-Progressive for an agent group of 50 agents.

pacing.connector.optimizationGoal

Default Value: 3

Valid Values: A positive integer between 1 and 99.

Changes Take Effect: After start/restart

Description: Specifies the percentage goal for the optimization target; the value you set for `pacing.connector.optimizationGoal` depends on the value you set for pacing.connector.optimizationTarget.

- If your optimization target is `ABANDONMENT_RATE`, Genesys recommends that you use small values, such

as 3 to 5. For example, a value of 3 means that no more than 3% of interactions will be abandoned.

- If your optimization target is BUSY_FACTOR, Genesys recommends that you use big values, such as 70 to 85. For example, a value of 75 means that no less than 75% of the time, agents will be busy with an interaction-related activity.

pacing.connector.optimizationTarget

Default Value: ABANDONEMENT_RATE

Valid Values: ABANDONEMENT_RATE, BUSY_FACTOR

Changes Take Effect: After start/restart

Description: Specifies the optimization target for the pacing algorithm. The possible 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.

This option is associated with the optimizationGoal option, which defines the percentage to use for the current optimization target.

pacing.connector.proactiveRatio

Default Value: 0

Value Values: 0—100

Changes Take Effect: After start/restart

Description: Specifies the minimum percentage of agent resources that are reserved to handle proactive interactions. If the value is set to 0, no resources are specifically allocated to handle proactive interactions (note that proactive traffic is still allowed). If the value is set to 100, all resources are allocated to handle proactive interactions and no reactive interactions are allowed.

You should specify a value for this option if you set pacing.connector.algorithm to a dual algorithm (SUPER_PROGRESSIVE_DUAL or PREDICTIVE_B_DUAL).

pacing.connector.chatGroup

Default Value: Web Engagement Chat

Valid Values: One or more chat group names. You must separate multiple groups with a semi-colon (;). For example, Chat Group 1;Chat Group 2.

Changes Take Effect: After start/restart

Description: Defines the agent group that manages chat engagements.
Note: An agent can only belong to one of the chat groups. If an agent belongs to more than one chat group, the pacing algorithm will produce incorrect results unless you customize the Engagement Logic SCXML strategy. See Accessing Pacing Information from the Engagement Logic Strategy for details.

> **Important**
> Leading or trailing spaces are considered part of the group name. For example, `My Group 1; My Group 2` and `My Group 1;My Group 2` are different sets because the first set has a space after the semi-colon.

pacing.connector.voiceGroup

Default Value: `Web Engagement Voice`

Valid Values: One or more voice group names. You must separate multiple groups with a semi-colon (;). For example, `Voice Group 1;Voice Group 2`.

Changes Take Effect: After start/restart

Description: Defines the agent group managing voice engagements.
Note: An agent can only belong to one of the voice groups defined for the pacing algorithm. If an agent belongs to more than one voice group, the pacing algorithm will produce incorrect results unless you customize the Engagement Logic SCXML strategy. See Accessing Pacing Information from the Engagement Logic Strategy for details.

> **Important**
> Leading or trailing spaces are considered part of the group name. For example, `My Group 1; My Group 2` and `My Group 1;My Group 2` are different sets because the first set has a space after the semi-colon.

pacing.connector.refreshPeriod

Default Value: 1

Valid Values: 1 – 5

Changes Take Effect: After start/restart

Description: Defines the period, in seconds, to refresh the data retrieved from Stat Server. The pacing algorithm is executed as soon as the data is refreshed. This value option determines how fast the pacing algorithm makes predictions.

Backend Server settings Section

The configuration options in the `settings` section are general settings for the Backend Server.

event-mode

Default Value: `true`
Valid Values: `true`, `false`
Changes Take Effect: After start/restart

Enables or disables the event mode for collaboration with Genesys Rules System. This option is used by the Genesys Rules Authoring Tool when rules are deployed.

loadbalancer

Default Value: `""`
Valid Values: A valid HTTP load balancer URL.
Changes Take Effect: After start/restart

Specifies the address of the load balancer for a cluster of Frontend servers. If specified, this value is used as the address for delivering events from the Backend Server to the Frontend Server. If specified, this option **overrides** connections defined in a server's connection tab. If not specified, you must set a connection between the Frontend Server and Backend server directly (direct configuration method should only be used for development and testing). For more information on configuring load balancing, see Load Balancing.

Backend Server security Section

The options in the `security` section configure the following security-related settings for the Backend Server:

- Enable authentication for the History REST API and its clients. See Authentication for details.
- Enable TLS between Genesys Web Engagement servers and other Genesys servers (for example, Chat Server or Interaction Server).

auth-scheme

Default Value: `none`
Valid Values: `none` or `basic`
Changes take effect: After start/restart

Specifies the HTTP authentication scheme used to secure History REST API requests to the Backend Server. With the `Basic` scheme, clients must be authenticated with a user ID and password.

user-id

Default Value: No default value
Valid Values: Any string
Changes take effect: After start/restart

The user identifier (login) used in authentication for the History REST API. See auth-scheme.

password

Default Value: No default value
Valid Values: Any string
Changes take effect: After start/restart

The user password used in authentication for the History REST API. See auth-scheme.

trusted-ca-type

Default Value: `MSCAPI`
Valid Values:

- `MSCAPI` – MSCAPI certificate storage is used for TLS certificate verification.
- `PEM` – PEM certificate storage is used for TLS certificate verification. In this case, the trusted-ca option should also be specified and should contain the path to the PEM file.
- `JKS` – JKS certificate storage is used for TLS certificate verification. In this case, the trusted-ca option should also be specified and should contain the path to the JKS file. You should also set the trusted-ca-pwd option to the password for the JKS file.

Changes Take Effect: After start/restart

Specifies the type of trusted certificate authority. No TLS is applied for connections between this server and other Genesys servers if this option is absent.

trusted-ca

Default Value:
Valid Values: Path to the trusted store file (valid for PEM and JKS types, depending on value of the trusted-ca-type option).
Changes Take Effect: After start/restart
Specifies the path to the trusted store file (valid for PEM and JKS types, depending on value of the trusted-ca-type option).

trusted-ca-pwd

Default value:
Valid values: Password for the trusted store file (valid for JKS type only).
Changes Take Effect: After start/restart
Specifies the password for the trusted store file (valid for JKS type only).

Frontend Server log Section

all

Default Value: `stdout`
Valid Values (log output types):

`stdout`	Log events are sent to the Standard output (`stdout`).
`stderr`	Log events are sent to the Standard error output (`stderr`).
`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.Setting the `all` log level option to the `network` output enables an application to send log events of the `Standard, Interaction,` and `Trace` levels to Message Server. Debug-level log events are neither sent to Message Server nor stored in the Log Database.
`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: After start/restart

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
```

standard

Default Value: `stdout` Valid Values (log output types):

`stdout`	Log events are sent to the Standard output (`stdout`).
`stderr`	Log events are sent to the Standard error output (`stderr`).
`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

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
```

trace

Default Value: `stdout` Valid Values (log output types):

`stdout`	Log events are sent to the Standard output (`stdout`).
`stderr`	Log events are sent to the Standard error output (`stderr`).
`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

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
```

verbose

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

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.

segment

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

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.

expire

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.`
`<number> day`	Sets the maximum number of days before log files are deleted. Specify a number from `1—100.`

Changes Take Effect: After restart

Determines whether log files expire. If they do, sets the measurement for determining when they expire, along with the maximum number of files (segments) or days before the files are removed.

This option is ignored if log output is not configured to be sent to a log file.

> **Warning**
> If an option's value is set incorrectly — out of the range of valid values — it will be automatically reset to `10`.

affectedLoggers

Default Value:
Valid Values: The names of loggers, separated by a semicolon (;), specified in the `LOG4J2.xml`. For example: `com.genesyslab.webme.commons;PROTOCOL;org.apache.cassandra`
Changes Take Effect: Immediately
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.

time_format

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
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
```

time_convert

Default Value: `local`
Valid Values:

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

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

Frontend Server settings Section

loadbalancer

Default Value: ""
Valid Values: A valid HTTP load balancer URL.
Changes Take Effect: After start/restart

Specifies the address of the load balancer for a cluster of Backend servers. If specified, this value is used as the address for delivering events from the Frontend Server to the Backend Server. If specified, this option **overrides** connections defined in a server's connection tab. If not specified, you must set a connection between the Frontend Server and Backend server directly (direct configuration method should only be used for development and testing). For more information on configuring load balancing, see Load Balancing.

Frontend Server service:cep Section

cep.domainSeparation

Default Value: `false`
Valid Values: `true`, `false`
Changes Take Effect: After start/restart
Description: Specifies to the Complex Event Processing (CEP) engine whether rules should be separated by domains. If rules are separated, there is a dedicated CEP session for each sub-domain and domain. Events that are triggered in the sub-domain are also evaluated in the upper domain(s). If rules are not separated, a single CEP session is be used for all domains.