Web Services and Applications Deployment Guide

Web Services and Applications 8.6.0

4/11/2025

Table of Contents

Web Services and Applications Deployment Guide

Welcome to the *Web Services and Applications Deployment Guide*. This document provides information about deploying Web Services API, Workspace Web Edition, and Gplus Adapters.

About Web Services and Applications

Planning your Web Services and Applications Deployment

Deploying Web Services and Applications

Configuring Web Services and Applications

Monitoring

Deploying and Configuring Gplus Adapters

Overview

Web Services and Applications is a set of REST APIs and user interfaces that provide a web-based client interface to access Genesys services. The following UIs are currently offered:

- Workspace Web Edition
- Gplus Adapter for Salesforce

To work with the API and these applications, you must first follow the steps in this Deployment Guide to install and configure the Web Services API server component of the product. After that, you can work directly with the APIs or provide specific configuration for the applications you plan to use.

Web Services API

Web Services are the REST APIs that can be used by developers to create custom agent applications that integrate with Genesys. These applications can include features such as state management, call control, supervisor monitoring, and call recording.

- Related documentation: Web Services API Reference

Note: GWS also provides a set of RESTful APIs used by Genesys Outbound Contact Expert for managing Outbound interactions.

Workspace Web Edition

Workspace Web Edition is an HTML 5 thin-client application that provides agents and knowledge workers with non-intrusive access to the information, processes, and applications that they need to perform their jobs more efficiently and to ensure increased customer satisfaction.

- Related documentation: Web Services and Applications Configuration Guide, Workspace Web Edition Help, and Workspace Web Edition Developer's Guide and API Reference.

Gplus Adapter for Salesforce

Gplus Adapter for Salesforce is an integrated solution that enables Salesforce users to handle contact center interactions seamlessly within Salesforce.

- Related documentation: Gplus Adapter for Salesforce section in this guide, Web Services and Applications Configuration Guide, and Gplus Adapters User Guide

Architecture

This page describes the standard deployment architectures for Genesys Web Services (GWS) 8.6.

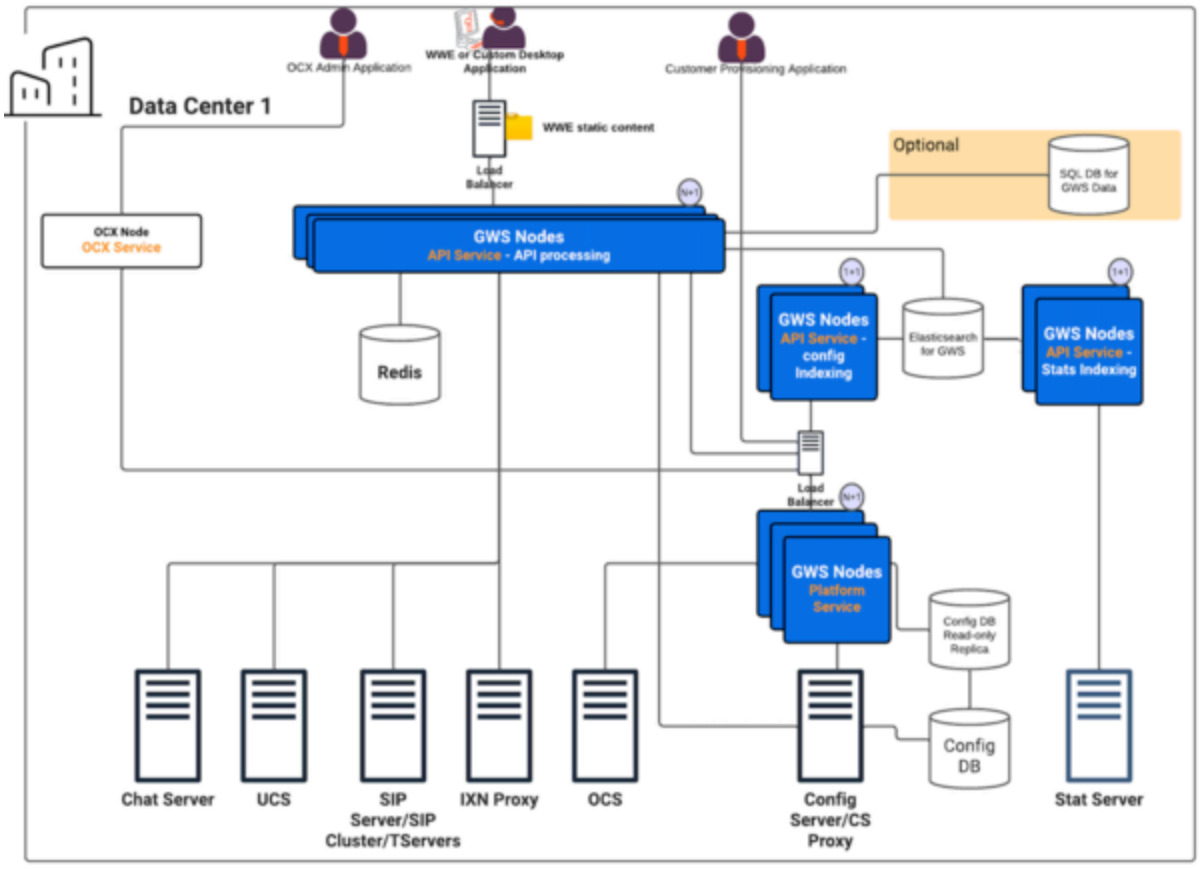
GWS comprises of the following three software components:

1. **GWS API Service** – can be configured through a configuration option to perform the following roles:
    1. **API Processing** – handles all API requests from Workspace Web Edition (WWE) or Custom Desktop Applications.
    2. **Configuration Indexing** – populates Elasticsearch with configuration information to support searches by WWE or Custom Desktop Applications.
    3. **Statistics Indexing** – populates Elasticsearch with statistical information to provide statistics to WWE or Custom Desktop Applications.
    4. **WWE Static Content** – provides the UI web content for WWE.
2. **GWS Platform Service**
    1. Configuration API
    2. Outbound API
3. **Agent Desktop Package** – WWE Static Content is provided as a separate package to install on Load Balancer for improved scalability.

> **Important**
>
> Nearly all GWS/WWE deployments require Elasticsearch for providing statistics and search capabilities within WWE. For deployments which use only GWS APIs for custom desktop applications, if the API usage does not invoke APIs that trigger Elasticsearch usage, then GWS can be deployed without index nodes and without Elasticsearch. If your deployment type belongs to this model, then consult with Genesys support.

Large deployment

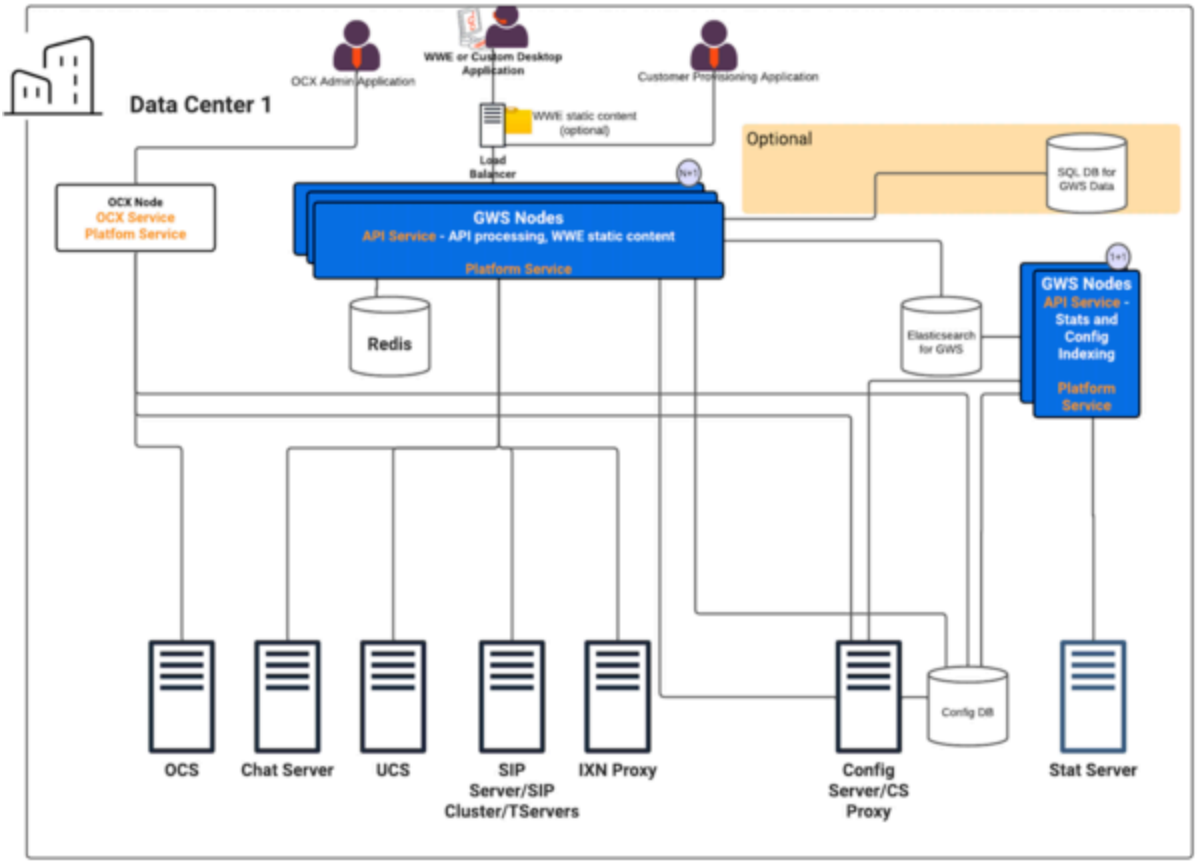
1. For maximum scalability and resiliency, the Web Services components are deployed on different nodes as described in the diagram below.
2. GWS API Service with API processing role operates with N+1 High Availability (HA).
3. GWS API Service with Configuration Indexing role operates with 1+1 High Availability (HA).
4. GWS API Service with Statistics Indexing role operates with 1+1 High Availability (HA).
5. GWS Platform Service operates with N+1 High Availability (HA)
6. WWE Static Content Service is deployed on Load Balancer



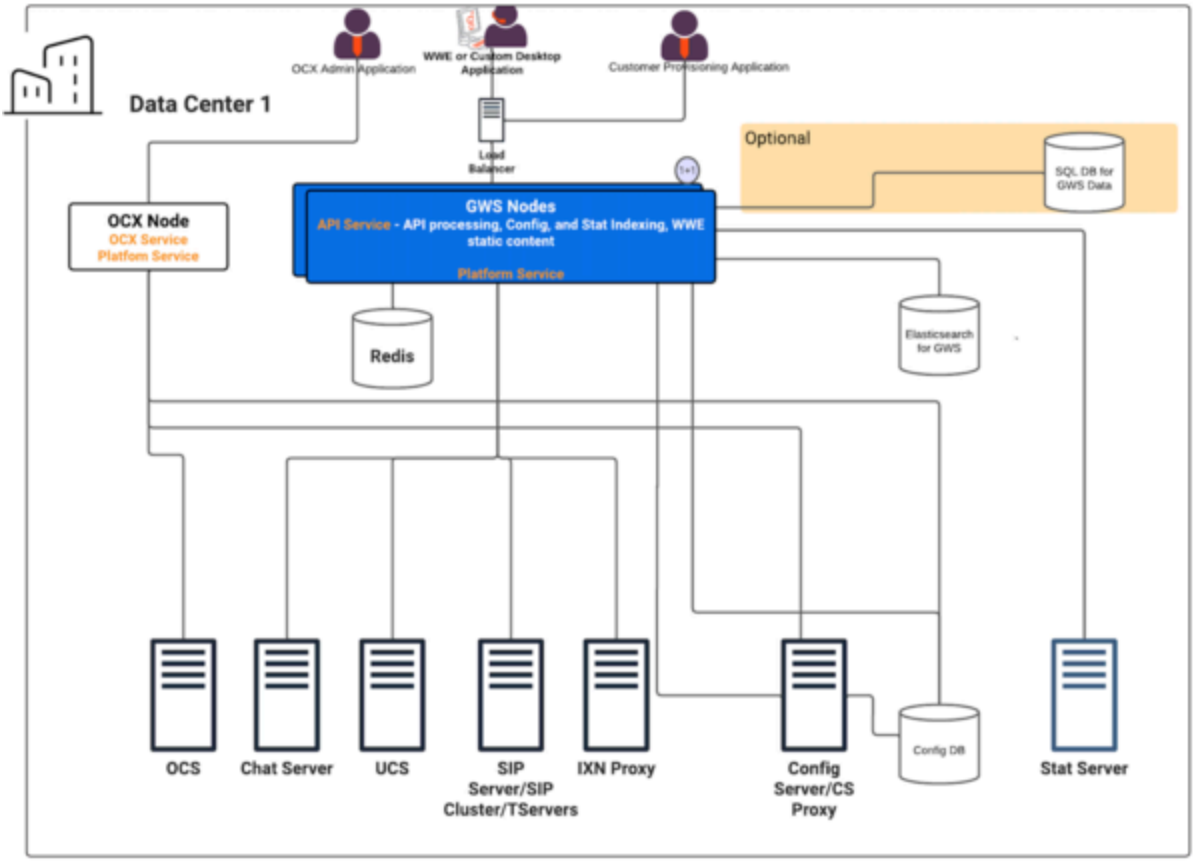
To optimize the hardware resource usage, the following are additional deployment references:

Medium Deployment

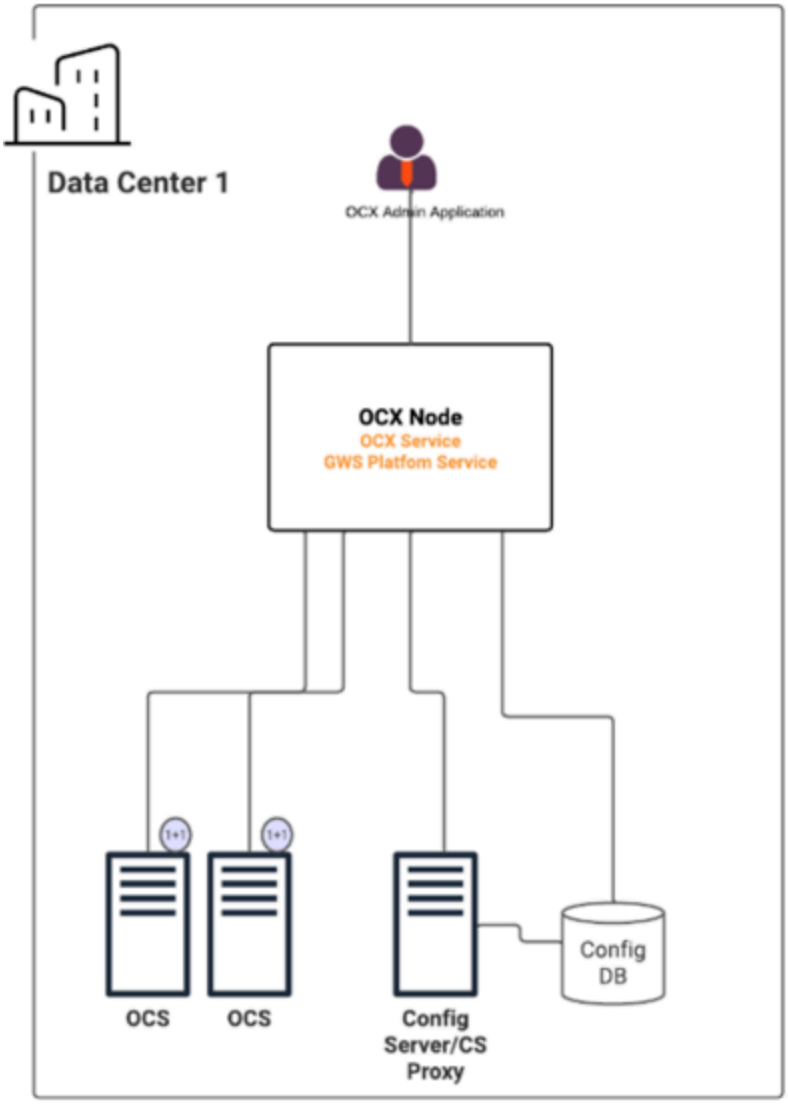
1. GWS API Service with API processing role operates with N+1 High Availability (HA).
2. GWS API Service with Statistics Config Indexing role operates with 1+1 High Availability (HA).
3. GWS API Service with Statistics Indexing role operates with 1+1 High Availability (HA).
4. GWS Platform Service operates with N+1 High Availability (HA)
5. WWE Static Content Service is deployed on Load Balancer


Data Center 1
OCX Admin Application
WWE or Custom Desktop Application
Customer Provisioning Application
WWE static content (optional)
Load Balancer
Optional
SQL DB for GWS Data
OCX Node
OCX Service
Platfom Service
GWS Nodes
API Service - API processing, WWE static content
Platform Service
N+1
Redis
Elasticsearch for GWS
GWS Nodes
API Service - Stats and Config Indexing
Platform Service
Config DB
OCS
Chat Server
UCS
SIP Server/SIP Cluster/TServers
IXN Proxy
Config Server/CS Proxy
Stat Server

Small Deployment



GWS Platform Service-only Deployment



This low-footprint deployment model is recommended when GWS 8.6 is required only in support of Outbound Contact Expert (no WWE Agent Desktop or custom agent desktop).

High-Availability Model for Web Services deployments

1. GWS Index nodes may be deployed in one of two ways:
    1. GWS Index nodes deployed in 1+1 HA, node configured to perform indexing of both configuration and statistics. Intended for deployments of fewer than 5,000 agents.
    2. GWS Index nodes deployed in two pairs of 1+1 HA, with one set configured to perform indexing of configuration, the other set configured to perform indexing of statistics. Intended for deployments of 5,000 agents and larger

2. The GWS SQL Database is optional. It supports storage of Custom Contacts and Custom Settings persistent settings – if these features are not utilized, then database is not required.
3. GWS 8.6 offers the following voice infrastructure support: SIP Server, SIP Cluster, and three T-Servers: Avaya, Cisco, Alcatel

Connections table

The following table lists the connection attributes between GWS and different components.

#	Service	Direction	Protocol	Local Port	Remote Peer	Remote Peer Port	Purpose
1	GWS	Inbound	HTTP	8080	Ngnix	Any	Web Services REST API to incorporate Genesys features into custom applications and integrations with the third-party software.
2	GWS	Outbound	TCP/TLS	Any	Platform API Service	Platform API listening port (8092) default.	Read/ Update configuration data.
3	Platform API Service	Outbound	TCP/TLS	Any	Configuration Server	Configuration Server/ Configuration Proxy listening port (8888).	Read/ Update configuration data.
4	GWS	Outbound	TCP	Any	Stat Server	Stat Server listening port (2060) default.	Obtain real-time agent state data.
5	GWS	Outbound	TCP	Any	SIP Servers/ SIP Cluster	SIP Server default listening port (5001) default. SIP Cluster default listening port (5004)	For making agent as well as call operations.

#	Service	Direction	Protocol	Local Port	Remote Peer	Remote Peer Port	Purpose
						default.	
6	GWS	Outbound	TCP	Any	Interaction Server	Interaction Server listening port (7040) default.	
7	GWS	Outbound	TCP	Any	Universal Contact Server (UCS)	UCS listening port (7130) default.	
8	GWS	Outbound	TCP	Any	Chat Server	Chat Server listening port (7160) default.	
9	GWS	Outbound	TCP	Any	Redis Cluster	6379 (default)	Storing session information.
10	GWS	Outbound	TCP	Any	MS SQL DB or PostgreSQL	1433 (default for MS SQL DB) or 5432 (default for PostgreSQL)	Persistent data storage
11	GWS	Outbound	TCP	Any	Elasticsearch	9200	Storing of configuration objects, statistics, and call information.
12	GWS	Outbound	TCP/TLS	Any	Configuration Server	Configuration Server/ Configuration Proxy listening port (8888)	Read/ Update configuration data.
13	GWS Platform Service	Outbound	TCP/TLS	Any	MS SQL DB or PostgreSQL	1433 (default for MS SQL DB) or 5432 (default for PostgreSQL)	Read configuration data directly from database.

Load balancing

Web Services supports any third-party load balancer that supports sticky sessions. You should configure session affinity (sticky sessions) based on `JSESSIONID`. After your load balancer is set up, you can use the following URL for health checks: `http://<Web_Services_Host>:<Web_Services_Monitoring_Port>/actuator/health/readiness`.

where,

- `Web_Services_Host` - is the host name where GWS application is hosted. In case you want to set a specific hostname, specify it in the **address** option in the **management** section of the **Application.yaml** file.
  For example:

  ```
  management:
    server:
      address: <address>
  ```

- `Web_Services_Monitoring_port` - is the value configured in the **port** option in the **management** section of the **Application.yaml** file.

  ```
  management:
    server:
      port: <port>
  ```

HTTPS Configuration for monitoring endpoint

If you want to enable TLS for the monitoring endpoint, the following additional configurations are needed.

For example:

```
management:
  server:
    port: <port>
    ssl:
      enabled: true
      key-store: <keystore path>
      key-store-password: <password>
      key-alias: <alias>
      key-password: <password>
  endpoints:
    web:
      exposure:
        include: 'health,prometheus,reindexdetails'
```

Important TLS Considerations

- Do not use IP address literals (for example, 192.168.1.215 or fe80::3831:400c:dc07:7c40) in your monitoring endpoint URL when TLS is enabled.
- SNI (Server Name Indication) requires that all hostnames used contain at least one dot (.). Avoid short or local hostnames (like localhost or myhost); use a fully qualified domain name (FQDN) such as

`monitoring.mycompany.com`.

- Ensure that your SSL certificate's Common Name (CN) or Subject Alternative Names (SANs) match the hostname used to access the service.
- The certificate specified in the key store must be properly configured, trusted, and should not be expired.

If you are configuring your solution to use Hypertext Transfer Protocol over Secure Socket Layer, you do not need to set up HTTPS between your load balancer and Web Services.

> **Important**
>
> Web Services and Applications does not currently support web sockets.

Caching

In Genesys Web Services 8.6, caching is done in **GWS API Service** and **GWS Platform Service**, however, the caching mechanism is different for each service which is noted as follows:

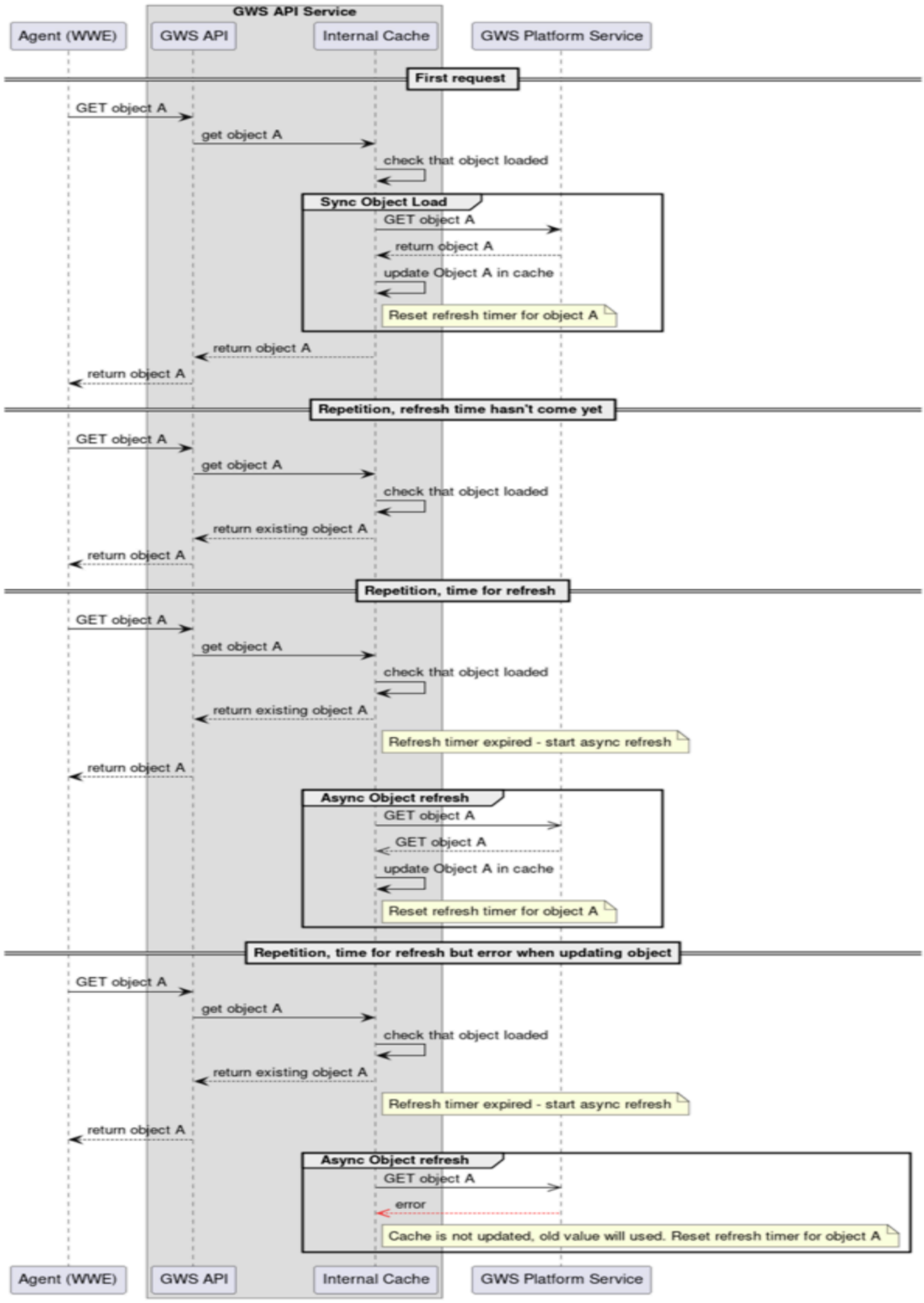
- **GWS API Service** implements its own small local cache.
- **GWS Platform Service** caches specific object types from Configuration database and keeps them in its own memory cache.

GWS API Caching

GWS locally caches a small subset of configuration data. The maximum size of the cached subset of data is limited and by default it is set to 1000 objects of same type (users, DNs, places, and so on). Adding additional objects will depend on the Heap size. Each object will be refreshed after the refresh timeout has lapsed (default is 5 min) in the background. Meanwhile, the object stored in cache will be returned immediately without waiting for the refresh completion. Since the object refresh is triggered by read requests, the *stale* objects will not be refreshed until accessed, which can significantly reduce the load on Configuration Database and Platform API.

For each object type, the refresh rate can be configured using the corresponding option in the **cachingSettings** section of the **application.yaml** file, for example, **placeTtl**, **usersTtl**, and so on. For full list of caching options, see Caching Settings.

GWS API Caching Sequence diagram


GWS API Service
Agent (WWE)
GWS API
Internal Cache
GWS Platform Service
First request
GET object A
get object A
check that object loaded
Sync Object Load
GET object A
return object A
update Object A in cache
Reset refresh timer for object A
return object A
return object A
Repetition, refresh time hasn't come yet
GET object A
get object A
check that object loaded
return existing object A
return object A
Repetition, time for refresh
GET object A
get object A
check that object loaded
return existing object A
Refresh timer expired - start async refresh
return object A
Async Object refresh
GET object A
GET object A
update Object A in cache
Reset refresh timer for object A
Repetition, time for refresh but error when updating object
GET object A
get object A
check that object loaded
return existing object A
Refresh timer expired - start async refresh
return object A
Async Object refresh
GET object A
error
Cache is not updated, old value will used. Reset refresh timer for object A
Agent (WWE)
GWS API
Internal Cache
GWS Platform Service

> **Important**
>
> As the contact center objects are cached at two levels, under normal circumstances, any object that is modified in Configuration Server will take a maximum of (API cache timeout + 1 second) to be reflected. If caching is disrupted for any unknown reason, then the Platform API's cache refresh will self-heal and restore the caching process.

> **Important**
>
> On top of caching, GWS API utilizes ElasticSearch to fulfill API requests that require searching or filtering operations. API Configuration Indexing nodes add new objects or update existing during re-indexation procedure that happens on regular basis by schedule that is 15 minutes by default (indexVerificationInterval configuration parameter). Hence, if new object has been added to ConfigServer or something has been updated, those changes will be visible by GWS API only when next re-indexation procedure is completed. That behavior is different in compare with 8.5 where changes are applied immediately in primary region where API SyncNode is deployed. Non-primary regions in 8.5 deployments behave the same way as 8.6 now.

GWS Platform API Caching

GWS Platform Service serves 'read API requests' using an internal in-memory cache or through Configuration Server. Note that not all object types are supported for caching. For the list of supported datatypes for which data is read directly from Configuration Server database, refer here.

Unsupported types and write operations (create/update/delete) are still handled through Configuration Server. All reads from database are done in subsequent manner through a single database connection by background processes. The background processes that support these requests are:

- A process for periodical full cache reevaluation that runs every 15 minutes (can be configured).
- A process for ongoing cache update procedure that runs every second picking up all changes in the database since the previous run, hence keeping the cache up-to-date. GWS Platform Service will retain and use existing cached information for the entire duration of any connectivity issues with the Configuration Database.

Supported object types for handling via in-memory cache are:

- CFGAccessGroup
- CFGActionCode
- CFGAgentGroup
- CFGAgentLogin

- CFGApplication
- CFGCallingList
- CFGCampaign
- CFGCampaignGroup
- CFGDN
- CFGDNGroup
- CFGEnumerator
- CFGEnumeratorValue
- CFGField
- CFGFilter
- CFGFolder
- CFGFormat
- CFGHost
- CFGPerson
- CFGPlace
- CFGPlaceGroup
- CFGRole
- CFGScript
- CFGService
- CFGSkill
- CFGSwitch
- CFGTableAccess
- CFGTenant
- CFGTransaction
- CFGTreatment

GWS Platform Caching Sequence diagram


GWS Platform Service
GWS API Service
Platform API
ConfigServer
Config DB
Service Initialization
Service instance started
Read configuration
Open connections to Database
Read all data from Tenants table
Compose Tenat objects
Read all data from <ObjectType> table
Compose <ObjectType> objects
Read ACL (permissions) data from table
Compose ACL objects
All data has been read from Database
Initialize dataset
Calculate VAGs
Calculate Permissions
Create initial data snapshot and write into cache
In-memory cache
Start backround cache update procedure
Schedule async cache refresh task
Open connection with ConfigServer
Service instance is ready to serve API requests
Update Cache
Check Database for changes (read HistoryLog)
no changes
Schedule next check task
Check Database for changes
There are changes
Recalculate updated objects
Recalculate affected VAGs
Update changed objects in cache
Schedule next check task
Refresh Cache
Read the latest record from HistoryLog
Read Tenant data
Compose Tenat objects
Read <ObjectType> data
Compose <ObjectType> objects
Read ACL (permissions) data
Compose ACL objects
All data has been read from Database
Initialize dataset
Calculate VAGs
Calculate Permissions
Stop cache update procedure
Read the latest record from HistoryLog
Recalculate updated objects
Recalculate affected VAGs
Create a new data snapshot and write into the cache
Data in the cache was swapped with a new one
Start backround cache update procedure
Schedule next async refresh task
Handle API Requests
API request to read objects
Apply request filters
Filter out objects by their permissions
API request to update object
Update object in Database
Write Delta object into HistoryLog
GWS API Service
Platform API
In-memory cache
ConfigServer
Config DB

Frequently Asked Questions

GWS API Caching

Do I have control over the cache refresh timer?

Yes, you do have control over refresh timeouts, you can configure them using Caching Settings.

Will I know how old the data is in the response? If the refresh fails, will I be notified in the next request that the data is outdated?

No, we don't provide any info on how old the data is since it might be composed from multiple data sources (caches) with different ages.

GWS Platform API Caching

What happens when a read API request is made and the configuration synchronization process fails more than once?

GWS Platform Service isn't ready until cache is populated once. After that, all read requests for supported data types are handled only via in-memory cache which is updated by background processes. If the update procedure fails then a full refresh is scheduled immediately (with delay in action). Also, cache is refreshed on a regular basis, every 15 mins.

CometD

Web Services uses CometD version 8, an HTTP-based routing bus that uses an Ajax Push technology pattern known as Comet. Comet is a web application model that allows an HTTP request to push data to a browser, even if the browser has not requested it. Note: CometD version 7 was used prior to release 8.6.000.06

Web Services uses CometD to deliver unsolicited notifications to clients for real-time events, such as getting a new call or chat message. At runtime, CometD delivers messages, providing clients with a consistent approach while maintaining support for multiple browsers.

> **Important**
> Web Services and Applications does not currently support web sockets.

For details about CometD, see http://cometd.org/.

For details about how to configure CometD in Web Services, see cometDSettings.

Persistent Data Storage

Genesys Web Services (GWS) optionally stores the following information in SQL database. If these APIs are not used, then database is not required.

- Settings and Settings Group
- Contacts

Sizing

This article describes the sizing requirements for your contact center based on the number of agents using Genesys.

Sizing recommendations for standard deployment

Standard hardware requirement is as follows:

- 4CPU cores, 16GB RAM, 32GB HDD for Standard GWS VM.
- One VM for API (no WWE) + Platform service – can handle 1000 concurrent agents, login rate 1 agent/sec, transactions not limited factor.
- For small non-production lab deployments with less than 10 concurrent agents, databases may be deployed on the same VM or host.

Sizing recommendations based on concurrent agents usage

The following table provides hardware sizing information for each Data Center based on the number of concurrent agents' usage.

Contact Center Size	Hardware Sizing for HA	Service Description

Small (<=1000 concurrent agents)	1+1 GWS VMs (or 1 single GWS VM for non-HA deployments)	• GWS API Service
		  • API processing
		  • Statistics Indexing
		  • Configuration Indexing
		  • WWE Static Content
		• GWS Platform Service

Medium (<=5000 concurrent agents)	1+1 GWS VMs	• GWS API Service
		  • Statistics Indexing
		  • Configuration Indexing
		• GWS Platform Service

	N+1 GWS VMs (1 VM per 1000 concurrent agents)	• GWS API Service

		• API processing • WWE Static Content • GWS Platform Service
	Optional: 1 Load Balancer	• WWE Static Content
Large (>5000 concurrent agents)	1+1 GWS VMs	• GWS API Service • Statistics Indexing
	1+1 GWS VMs	• GWS API Service • Configuration Indexing
	N+1 GWS VMs (1 VM per 1000 concurrent agents)	• GWS API Service • API processing
	K+1 GWS VMs	• GWS Platform Service (half as many VMs as API Service VMs)
	1 Load Balancer (determined by the customer)	• Balances the load between GWS API Service nodes and GWS Platform Service nodes
	1 Load Balancer	• WWE Static Content
Platform Service-only (typically for Outbound Contact Expert without Agent Desktop)	1+1 GWS VMs	• GWS Platform Service

> **Important**
> You can choose to partition a single Load Balancer for both external and internal load balancing.

Redis Sizing Guidelines

For production deployments of GWS 8.6, the requirements of 3 primary and 3 replica nodes for a Redis cluster should be sufficient.
The number of Redis keys is 6 keys per agent (1 key per agent session, 3 keys per current active

session, 1 key per call, 1 key per agent's interactions), with key size up to 4kB
Index size: 50 MB per 10,000 agents
Throughput: Up to 12 operations/sec with Redis under high load for each agent session (depends on specific traffic profile for a deployment)

Planning your deployment

Genesys Web Services 8.6 require the following dependent components:

- GWS Platform Service
- Redis Cluster
- Elasticsearch
- SQL database (MS SQL or Postgres) - only required if Genesys Web Services is storing persistent data.
- Configuration Database or Configuration Database Replica (MS SQL or Postgres)

In the **production environment**, you must deploy the Redis cluster, Elasticsearch cluster, and SQL database (MS SQL or Postgres if used). Also depending on the size of the deployment the GWS Platform Service must be deployed in each of the Virtual Machines (VMs) that runs Web Service or deployed on separate VMs.

In the **development environment**, you can install Redis cluster, Elasticsearch, SQL database, GWS Platform Service and Web Services on the same host.

> **Important**
> The number of Web Services nodes required for your deployment depends on a variety of factors. Contact your Genesys Representative to determine the optimal number of nodes needed for your deployment.

For both production as well as development deployment follow the same basic deployment steps.

Deployment tasks

Complete the following steps to deploy Genesys Web Services 8.6.

> **Important**
> If you are upgrading from one version of Genesys Web Services to another, see the Web Services and Applications Migration Guide.

1. Review the Prerequisites. Make sure all supporting components are installed and configured.
2. Install Web Services and Applications.
3. Initialize Redis.

4. Initialize Postgres or MSSQL.
5. Deploy the web application.
6. Configure Web Services.
7. Customize your required features, as necessary.
8. Configure additional security (optional).
9. Start and test Web Services, Workspace Web Edition, and Gplus Adapter for Salesforce.
10. Follow up with the appropriate configuration or developer information, depending on which components you plan to use:
    - Configure Workspace Web Edition
    - Work with Web Services (takes you to a new guide)
    - Deploy Gplus Adapter for Salesforce
    - Configure Gplus Adapter for Salesforce (takes you to a new guide)

Next step

- Review the Prerequisites

Prerequisites

To work with Web Services and Applications, your system must meet the software and browser requirements established in the Genesys Supported Operating Environment Reference Guide, and meet the following minimum requirements.

> **Important**
> In multi-tenant Configuration Server deployments, Web Services and Applications support only one tenant that contains all configuration data.

Server Requirements

OS requirements

- Red Hat Enterprise Linux, version 8
- Red Hat Enterprise Linux, version 9
- glibc(x86-64) >= 2.17
- libtool-ltdl(x86-64) >= 2.4.6
- libgcc(x86-64) >= 8.5.0

Java requirements

- Genesys Web Services and Applications version 8.6+ requires Java 17. You can download and install Java using Java 17 64bit for Linux. For more information, refer to the Java documentation.

Database requirements

- Elasticsearch 8
- Redis 7.2 (Note: Previous releases of GWS 8.6 required Redis 6)
- PostgreSQL 13/14 or MSSQL 2019 or Oracle 19C (Only required if Web Services and Applications stores persistent data)
- Configuration Database or Configuration Database Replica: PostgreSQL 13/14 or MSSQL 2019 or Oracle 19C

Note: A Platform Service-only deployment does not require Redis, Elasticsearch, or a SQL database for GWS persistent data.

Jetty requirements

- Jetty 12 is now embedded in the Web Services and Applications installation. Note: Previous releases of GWS 8.6 included Jetty 11.

Browser support

> **Important**
> Genesys recommends to use the latest release of each browser. However, Genesys will review and address the issues reported in the N-1 versions of the browsers.

Workspace Web Edition

Workspace Web Edition supports the following browsers:

- Google Chrome
- Firefox
- Microsoft Edge Chromium

Gplus Adapter for Salesforce

- Google Chrome
- Firefox

Client Workstation Requirements

The following table shows the recommended hardware requirements for the client workstations.

Recommended Hardware Requirements - Client Workstation

Processor	Memory	Hard Drive	Graphic Card	Network
Intel Core 2 Duo CPU 2.8 GHz	4 GB	200 MB	DirectX 9.0+	xDSL / LAN

NOTE: 8 GB of RAM is recommended when non-Genesys applications are being run concurrently, or to improve performance.

Next step

* Installing Web Services and Applications

Installing

To install Web Services, first you need to set up the two application objects it uses in the Genesys configuration environment:

- An application of type `Genesys Generic Server` that is called the WS Cluster Application.
- An application of type `Genesys Generic Client` that is called the WS Node Application.

Configuring the Web Services applications

Perform the following procedures to configure the Web Services applications. Select a tab to configure the applications in *either* Configuration Manager or Genesys Administrator.

Configuration Manager

Creating and importing the application templates

The Web Services installation package includes a template for `Genesys Generic Server` (the WS Cluster Application), but you must create a new template for `Genesys Generic Client` (the WS Node Application).

Start

1. To create the `Genesys Generic Client` template, navigate to the **Application Templates** folder. Right-click and select **New > Application Template**.
2. Configure the **General** tab of the template as shown below:
   - Name: WS_Node
   - Type: Genesys Generic Client
   - Version: 8.6
   - State Enabled: Yes
3. Click **OK**.
4. To import the `Genesys Generic Server` template, navigate to the **Application Templates** folder in Configuration Manager. Right-click and select **Import Application Template**.
5. Navigate to the **templates** folder in your installation package.
6. Select the **Web_Services_and_Applications_860** template file.
7. Click **OK**.

End

Creating the WS Cluster Application

Start

1. Navigate to the **Applications** folder. Right-click and select **New > Application**.
2. Select the **Web_Services_and_Applications_860** template and click **OK**.
3. Configure the **General** tab as shown below:
   - Name: WS_Cluster
   - Template: Workspace_Web_Edition_Web_Services_860
   - Component Type: [Unknown]
   - State Enabled: Yes
4. On the **Tenants** tab, click **Add**.
   1. Choose the `Environment` tenant (or any other tenant that has a connection to your Configuration Server).
   2. Click **OK**.
5. On the **Server Info** tab, choose a `Host` object. See Create Host in the *Management Framework Deployment Guide* for more information about Host objects. This automatically adds a corresponding port entry. The port value is ignored by the server and does not need to be modified.
6. On the **Start Info** tab, add a "." to the `Working Directory`, `Command Line`, and `Command Line Arguments` fields. These values are mandatory for all applications and must be entered to save the application object. Web Services does not use these values, so the "." is used as a placeholder.
7. On the **Connections** tab, add the following connections:
   - Configuration Server
   - T-Server/SIP Server (if supporting voice)
   - Interaction Server (if supporting multimedia)
   - Universal Contact Server (if supporting multimedia)
   - Stat Server (if supporting reporting)

   > **Important**
   > Default statistics definitions provided with GWS 8.6 don't fully cover 'blended' scenarios where agents use voice and multimedia/digital channels simultaneously.

8. Click **OK** to save the `WS_Cluster` application.

End

Creating the WS Node Application

Start

1. Navigate to the **Applications** folder. Right-click and select **New > Application<**.
2. Select the WS_Node template and click **OK**.
3. Configure the **General** tab as shown below:
   - Name: WS_Node
   - Template: WS_Node
   - State Enabled: Yes
4. On the **Connections** tab, add the following connections:
   - Cluster application that was configured in the previous procedure.
5. Click **OK** to save the WS_Node application.

End

Genesys Administrator

Creating and importing the application templates

The Web Services installation package includes a template for Genesys Generic Server (the WS Cluster Application), but you must create a new template for Genesys Generic Client (the WS Node Application).

Start

1. To create the Genesys Generic Client template, navigate to **PROVISIONING > Environment> Application Templates**.
2. Select **New...** and configure the properties of the template as shown below:
   - Name: WS_Node
   - Type: Genesys Generic Client
   - Version: 8.6
   - State: Enabled
3. Click **Save & Close**.
4. To import the Genesys Generic Server template, click **Upload Template** in the **Tasks** panel. The **Click 'Add' and choose application template (APD) file to import** dialog opens.
5. Click **Add** and navigate to the **templates** folder in your installation package.
6. Select the **Web_Services_and_Applications_860** template file and click **Open**.
7. Click **Save & Close**.

End

Creating the WS Cluster Application

Start

1. Navigate to **PROVISIONING > Environment > Application** and click **New...**.
2. In the **General** section, configure the properties of the application as shown below:
   - Name: WS_Cluster
   - Template: Web_Services_and_Applications_860
   - State: Enabled
3. Add the following connections:
   - Configuration Server
   - T-Server/SIP Server (if supporting voice)
   - Interaction Server (if supporting multimedia)
   - Universal Contact Server (if supporting multimedia)
   - Stat Server (if supporting reporting)

   > **Important**
   > KPIs and Statistics are reported only for the voice channel. Web Services does not support real-time statistics for mixed media (voice/multimedia) environments. If a mixed media environment is used, voice statistics are not accurate.

4. In the **Server Info** section, select a `Tenant`:
   1. Click **Add**.
   2. Choose the `Environment` tenant (or any other tenant that has a connection to your Configuration Server).
   3. Click **OK**.
5. Choose a `Host` object. See Create Host in the *Management Framework Deployment Guide* for more information about Host objects.
6. Add a default Listening Port:
   1. Click **Add**.
   2. Enter the application's Port. For instance `7000`.
   3. Click **OK**.
7. Add a "." to the `Working Directory`, `Command Line`, and `Command Line Arguments` fields. These values are mandatory for all applications and must be entered to save the application object. Web Services does not use these values, so the "." is used as a placeholder.

End

Creating the WS Node Application

Start

1. Navigate to **PROVISIONING > Environment > Application** and click **New...**.
2. In the **General** section, configure the properties of the application as shown below:
   - Name: WS_Node
   - Template: WS_Node
   - State: Enabled
3. Add the following connections:
   - Cluster application that was configured in the previous procedure.
4. Click **Save & Close**.

End

Next step

- Deploying the web application

Deploying the web application

Prerequisites

Prerequisite for all Web Services and Applications Service VMs

- **RPM (RPM Package Manager):** Used for installing, updating, and removing software packages.

GWS Platform Service has the following direct dependencies that must be installed on each VM where it's planned to be installed.

- **glibc(x86-64)** >= 2.17
- **libtool-ltdl(x86-64)** >= 2.4.6
- **libgcc(x86-64)** >= 8.5.0

Third Party Component Prerequisite

- **OpenJDK 17:** Required for GWS API Service.

Installing GWS Platform Service

1. Install Unix ODBC manager: `yum install unixODBC`
2. Install MSSQL 18 or Oracle 23 ODBC driver according to corresponding driver documentation:
   - Install the Microsoft ODBC 18 driver for SQL Server
   - Oracle Instant Client ODBC Installation
3. Create DSN entry of corresponding DB driver in /etc/odbc.ini
   MSSQL:

   ```
   [MicrosoftODBC-18]
   Driver=ODBC Driver 18 for SQL Server
   ```

   Oracle:

   ```
   [OracleODBC-23]
   Driver=Oracle 23 ODBC driver
   ```

   Note: Oracle ODBC initialization script (odbc_update_ini.sh) allows to automatically create DSN in odbc.ini during installation, if it's done, step 3 should be omitted. For instance, to create DSN during installation, the script should be run with following arguments:

   ```
   ./odbc_update_ini.sh / /usr/lib/oracle/23/client64/lib "Oracle 23 ODBC driver"
   "OracleODBC-23" /etc/odbc.ini
   ```

4. Download the RPM Package to the target installation server.
5. Optional - you may skip this step and go to the next if you want to install GWS Platform Service with default user and group. To install the service with a custom user and group:
    1. Create a configuration file named **gws-service-platform.conf** in the directory **/usr/lib/sysusers.d/**
    2. In the configuration file created in the previous step, add the following lines to specify the username and groupname:

        ```
        u <username>
        g <groupname>
        ```

        For example, to create a user and group named 'genuser', the sample code will be:

        ```
        u genuser
        g genuser
        ```

        This process creates a new user and group named 'genuser' during installation and then assigns the ownership of the folder **/<installation_path>/gws-service-platform** to the newly created user and group.
6. Install the GWS Platform Service. You can either run a basic installation or custom installation, their respective commands follow:
    - If you want to run the basic installation, that is, installing the service in a default location, run the following command:
        ```
        rpm -ivh gws-service-platform-<version>-1.x86_64.rpm
        ```

        Running the above command installs the service in the default location: **/opt/genesys/gws-service-platform**. During installation, a default user and a group named **genesys** are created if it doesn't already exist and the ownership of the folder **/opt/genesys/gws-service-platform** is assigned to the newly created default user and group.
    - If you want to install the service in a custom location, run the following command with the `--prefix` option as shown below:
        ```
        rpm -ivh gws-service-platform-<version>-1.x86_64.rpm --prefix <custom_location>
        ```

        For example, to install the service in the path **/user/share**, the sample command will be:

        ```
        rpm -ivh gws-service-platform-<version>-1.x86_64.rpm --prefix /user/share
        ```

Updating configuration

You can update the existing configuration using the sample configuration file **environment.yaml** available in the RPM package.

To update the configuration,

1. Navigate to the default installation path: **/opt/genesys/gws-service-platform**. Or, the custom path: **/<installation_path>/gws-service-platform**.
2. Modify the configuration using the sample file **environment.yaml** and save.

Updating environment variables

If required, you can modify the default values of environment variables available in the **/usr/lib/**

systemd/system/gws-service-platform.service file to suit your organizational needs.

To modify the environment variables,

1. Find the environment variable in the **gws-service-platform.service** file using the keyword search **Environment=**.
2. Copy the variable(s) you wish to modify and paste them in the file **/usr/lib/systemd/system/gws-service-platform.service.d/gws-service-platform.conf**.
3. Modify the environment variable value as required. While modifying ensure that you follow the same format followed in the **gws-service-platform.service** file, that is, `Environment=<Env_variable_name>=<Env_variable_value>`

> **Important**
>
> It is recommended to keep your modified environment variables separately in **/usr/lib/systemd/system/gws-service-platform.service.d/gws-service-platform.conf** to avoid the risk of values being overwritten during upgrades or reinstallation.

An example **gws-service-platform.service** file with environment variables is as follows:

```
[Service]
Environment=GWS_SERVER_PORT=8092
Environment=GWS_ENV_CONFIG_FILE=./environment.yaml
Environment=GWS_COMMON_LOCATION=/
Environment=GWS_COMMON_AUTH_SECRET= (must match serverSettings.auth.secret in GWS
API Application yaml)
```

4. After modifying the required environment variables, run the following command to save the changes.
   `systemctl daemon-reload`

Note: For Platform Service-only deployments, the remainder of the installation steps are not applicable and should be skipped

Installing Web Services and Applications

Prerequisites check:

Ensure the dependent services such as Elasticsearch, PostgreSQL, Redis, and GWS Platform Service are installed and running. If the dependent services are installed on a different machine, ensure the respective hostnames are resolvable from the computer you're going to deploy the Web Services and Applications service.

1. Download the RPM Package to the target installation server.
2. Optional - you may skip this step and go to the next if you want to install Web Services and Applications Service with default user and group. To install the service with a custom user and group:
   1. Create a configuration file named **gws-api-v2.conf** in the directory **/usr/lib/sysusers.d/**
   2. In the configuration file created in the previous step, add the following lines to specify the username and groupname:

```
u <username>
g <groupname>
```

For example, to create a user and group named 'genuser', the sample code will be:

```
u genuser
g genuser
```

This process creates a new user and group named 'genuser' during installation and assigns the ownership of the folder **/<installation_path>/gws-api-v2** to the newly created user and group.

3. Install the Web Services and Applications Service. You can either run a basic installation or custom installation, their respective commands follow:
   - If you want to run the basic installation, that is, installing the service in a default location, run the following command:
     ```
     rpm -ivh gws-api-v2-<version>-1.noarch.rpm
     ```
     Running the above command installs the service in the default location: **/opt/genesys/gws-api-v2**. During installation, a default user and a group named **genesys** are created if it doesn't already exist and the ownership of the folder **/opt/genesys/gws-api-v2** is assigned to the newly created default user and group.
   - If you want to install the service in a custom location, run the following command with the `--prefix` option as shown below:
     ```
     rpm -ivh gws-api-v2-<version>-1.noarch.rpm --prefix <custom_location>
     ```
     For example, to install the service in the path **/user/share**, the sample command will be:
     ```
     rpm -ivh gws-api-v2-<version>-1.noarch.rpm --prefix /user/share
     ```

Updating configuration

You can update the existing configuration using the sample configuration files available in the RPM package. Review the Web Services and Configuration files section to know about the sample configuration files.

To update the configuration,

1. Navigate to the default installation path: **/opt/genesys/gws-api-v2/config**. Or, the custom path: **/<installation_path>/gws-api-v2/config**.
2. Modify the configuration using the related sample file (for example, **application.yaml**) and save.

Updating environment variables

If required, you can modify the default values of environment variables available in the **/usr/lib/systemd/system/gws-api-v2.service** file to suit your organizational needs.

To modify the environment variables,

1. Find the environment variable in the **gws-api-v2.service** file using the keyword search **Environment=** .

2. Copy the variable(s) you wish to modify and paste them in the file **/usr/lib/systemd/system/gws-api-v2.service.d/gws-api-v2.conf**.
3. Modify the environment variable value as required. While modifying ensure that you follow the same format followed in the **gws-api-v2.service** file, that is, `Environment=<Env_variable_name>=<Env_variable_value>`

> **Important**
>
> It is recommended to keep your modified environment variables separately in **/usr/lib/systemd/system/gws-api-v2.service.d/gws-api-v2.conf** to avoid the risk of values being overwritten during upgrades or reinstallation.

An example **gws-api-v2.service** file with environment variables is as follows:

```
[Service]
Environment="JAVA_OPTIONS=-javaagent:opentelemetry-javaagent.jar
-Dfile.encoding=UTF-8 -Djavax.net.ssl.keyStore=NONE"
Environment="JAVA_TOOL_OPTIONS=-XX:+ExitOnOutOfMemoryError -XX:+UseG1GC
-XX:+PrintFlagsFinal -Xms6g -Xmx6g"
```

4. After modifying the required environment variables, run the following command to save the changes.
   `systemctl daemon-reload`

Next Steps

1. Configuring Web Services
2. Initializing Redis
3. Initializing Postgres
4. Starting and Testing Web Services

Installation package files

Web Services configuration and initialization files (Red Hat Enterprise Linux 8 or 9)

File/folder name	Description
/usr/bin/gws-api-v2	The initialization script for the Web Services service.
/usr/lib/systemd/system/gws-api-v2.service	The default configuration file for the Web Services service.
/usr/lib/systemd/system/gws-api-v2.service.d/gws-api-v2.conf	The custom configuration file for the Web Services service. The settings defined in this file override

File/folder name	Description
	the default settings.

Web Services and Applications files

File/folder name	Description
/<installation_path>/gws-api-v2	The home directory of Web Services and Applications.
/<installation_path>/gws-api-v2/gws-api-v2.jar	The application **.jar** file.

Web Services and Applications Configuration files

File/folder name	Description
/<installation_path>/gws/config	The folder that contains sample/templates of the Web Services and Applications configuration files. After installation, files from this folder must be modified according to the deployment environment: • **/<installation_path>/gws-api-v2/config/application.yaml** • **/<installation_path>/gws-api-v2/config/statistics.yaml** • **/<installation_path>/gws-api-v2/config/logback.xml** • **/<installation_path>/gws-api-v2/config/feature-definitions.json**

GWS Platform Service, service configuration and initialization files (Red Hat Enterprise Linux 8 or 9)

File/folder name	Description
/usr/bin/gws-service-platform	The initialization script for the GWS Platform Service.
/usr/lib/systemd/system/gws-service-platform.service	The default service configuration file for the GWS Platform Service.
/usr/lib/systemd/system/gws-service-platform.service.d/gws-service-platform.conf	The custom configuration file for the GWS Platform Service. The settings defined in this file override the default settings in **/usr/lib/systemd/system/gws-service-platform.service**

GWS Platform Service configuration files

File/folder name	Description
/<installation_path>/gws-service-platform	The home directory of the GWS Platform Service.
/<installation_path>/gws-service-platform/gws-service-platform	The GWS Platform Service application file.
/<installation_path>/gws-service-platform/environment.yaml	The folder that contains configuration sample/templates of the GWS Platform Service. After installation, files from this folder must be modified according to the deployment environment.

Deploying Agent Desktop

For large deployments which require increased stability, the static web-based Agent Desktop UI content can be deployed directly on a load balancer. The same content is part of the API v2 service deployment.

Installing Agent Desktop

1. Download the RPM Package to the target installation server.
2. Install Agent Desktop. You can either run a basic installation or custom installation, their respective commands follow:
   - If you want to run the basic installation, that is, installing the service in a default location, run the following command:
     ```
     rpm -ivh gws-agent-desktop-<version>-1.x86_64.rpm
     ```
     Running the above command installs the service in the default location: **/opt/genesys/gws-agent-desktop**. During installation, a default user and a group named **genesys** are created if it doesn't already exist and the ownership of the folder **/opt/genesys/gws-agent-desktop** is assigned to the newly created default user and group.
   - If you want to install the service in a custom location, run the following command with the `--prefix` option as shown below:
     ```
     rpm -ivh gws-agent-desktop-<version>-1.x86_64.rpm --prefix <custom_location>
     ```
     For example, to install the service in the path **/user/share**, the sample command will be:
     ```
     rpm -ivh gws-agent-desktop-<version>-1.x86_64.rpm --prefix /user/share
     ```

Updating load balancer configuration

Configure your load balancer to use the locally installed UI content. UI content is identified by the URL path: **/ui/ad/***

Initialize Postgres

This article describes the initialization steps for Postgres and the hardware requirements for different deployment setups.

> **Important**
>
> Postgres is required only if Web Services and Applications stores persistent data such as:
>
> - Custom Contacts
> - Custom Settings

To initialize Postgres,

1. Create database using the following script:

```
CREATE DATABASE env
    WITH
    OWNER = env
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;
```

2. Once the database is created, create the schema using the following script:

```
CREATE SCHEMA IF NOT EXISTS gws
    AUTHORIZATION env;
```

> **Important**
>
> Creating database schema is essential for all deployment scenarios, including development environments, single-node, and two-node data centers.

Review the following tables to understand the hardware requirements of Postgres for different environments.

Development environment

Requirements	Description
Postgres Version	13 or 14

Requirements	Description
OS	There is no specific constraints for OS.
Processor Speed	2 cores
Memory	2 GB of RAM
Hard Disk / Storage Disk	512 MB of HDD
Volumes	Additional disk space is required for data or supporting components.
Networking	Localhost or internal network access.
Persistence	Database backup mechanism for data persistence.

Single data center

Requirements	Descriptions
Postgres Version	13 or 14
OS	There is no specific constraints for OS.
Processor Speed	4 cores
Memory	8 GB of RAM
Hard Disk / Storage Disk	10 GB
Volumes	Fast storage subsystem (SSD/NVMe) for database files, separate storage for transaction logs and backups.
Networking	Optimized network configuration for intra-data center communication
Persistence	Periodic database backups for point-in-time recovery.

Two data centers

Requirements	Descriptions
Postgres Version	13 or 14
OS	There is no specific constraints for OS.
Processor Speed	4 cores
Memory	8 GB RAM
Hard Disk / Storage Disk	10 GB
Volumes	Distributed storage solution with redundancy across data centers, fast storage for database files.
Networking	High-speed, low-latency connections between data centers.
Persistence	Streaming Replication or logical replication for

Requirements	Descriptions
	synchronous or asynchronous replication between data centers.

Initialize Microsoft SQL Server

This article describes the initialization steps for Microsoft SQL Server (MS SQL) and the hardware requirements for different deployment setups.

> **Important**
> Microsoft SQL Server is only required if Web Services and Applications stores persistent data such as:
>
> - Custom Contacts
> - Custom Settings

The initialization step for MS SQL Server involves creating a database name. Use the following script to create the database name:

```
CREATE DATABASE [gws_gws]
```

This script creates database name for any deployment environment such as development setup, single-node or two-node data centers.

> **Important**
> For MS SQL Server, database schema creation is not a mandatory step.

Review the following tables to understand the hardware requirements of MS SQL server for different environments.

Development environment

Requirements	Description
MS SQL version	2019 Developer edition
AOAG Required	Optional
Nodes	Minimum 1 Nodes
Memory	At least 4 GB per Node and should be increased as database size increases to ensure optimal performance.
Processor	2 cores

Requirements	Description
Hard Disk / Storage Disk	10 GB
Networking	Localhost or internal network access.
Persistence	Database backup mechanism for data persistence.

Single data center

Requirements	Description
MS SQL version	2019 Enterprise edition (with latest CU / Service Pack)
FailOver Cluster	Windows Server Failover Clustering (WSFC)
AOAG Required	Enabled
Nodes	Minimum 2 Nodes
Memory	At least 8 GB per Node and should be increased as database size increases to ensure optimal performance.
Processor	minimum 2 cores per node
Hard Disk / Storage Disk	4 drives (Data, Log, Backup, TempDB), 100GB per drive
Networking	Optimized network configuration for intra-data center communication.
Persistence	Always On Availability Groups or similar technology for synchronous or asynchronous replication between data centers.

Two data centers

Requirements	Description
MS SQL version	2019 Enterprise edition (with latest CU / Service Pack)
FailOver Cluster	Windows Server Failover Clustering (WSFC)
AOAG Required	Enabled
Nodes	Minimum 4 Nodes
Memory	At least 16 GB per Node and should be increased as database size increases to ensure optimal performance.
Processor	Minimum 2 cores per node
Hard Disk / Storage Disk	4 drives(Data, Log, Backup, TempDB), 150 GB per drive
Networking	High-speed, low-latency connections between data

Requirements	Description
	centers
Persistence	Always On Availability Groups or similar technology for synchronous or asynchronous replication between data centers.

Initialize Oracle

This article describes the initialization steps for Oracle Database 19c and the hardware requirements for different deployment setups.

> **Important**
>
> Oracle Database is only required if Web Services and Applications stores persistent data such as:
>
> - Custom Contacts
> - Custom Settings

Typically, Oracle initialization doesn't require to create a database instance, only a new dedicated account and granting required permissions is necessary. However, if a dedicated database instance for GWS is required by policy, it should be created according to the official Oracle documentation: Creating a Database with the CREATE DATABASE Statement.

To create an account which is a database user through which GWS connects to the Oracle database, the following SQL DDL-statement can be used:

```
CREATE USER gws_dev
  IDENTIFIED BY gws_password
  DEFAULT TABLESPACE gws_env
  ACCOUNT UNLOCK;
```

For more details and options, please see CREATE USER Statement.

The following SQL command grants required privileges to the created account on the previous step (refer to GRANT Statement for more information):

```
GRANT CONNECT,
  CREATE TABLE,
  CREATE VIEW,
  CREATE PROCEDURE,
  UNLIMITED TABLESPACE
  TO gws_dev;
```

Review the following tables to understand the hardware requirements of Oracle Database for different environments.

Development environment

Requirements	Description
Oracle version	Oracle XE
HA mode	none
Memory	At least 4 GB per Node and should be increased as database size increases to ensure optimal performance.
Processor	2 cores
Hard Disk / Storage Disk	10 GB
Networking	Localhost or internal network access.
Persistence	Database backup mechanism for data persistence.

Single data center

Requirements	Description
Oracle version	Oracle 19c
HA mode	RAC
Memory	At least 8 GB per Node and should be increased as database size increases to ensure optimal performance.
Processor	Minimum 2 cores per node
Hard Disk / Storage Disk	4 drives (Data, Log, Backup, TempDB), 100GB per drive
Networking	Optimized network configuration for intra-data center communication.

Two data centers

Requirements	Description
Oracle version	Oracle 19c
HA Mode	DataGuard or MAA
Memory	At least 16 GB per Node and should be increased as database size increases to ensure optimal performance.
Processor	Minimum 2 cores per node
Hard Disk / Storage Disk	4 drives(Data, Log, Backup, TempDB), 150 GB per drive
Networking	High-speed, low-latency connections between data centers

Initialize Redis

Web Services and Applications uses Redis to store transient data about agent sessions and interactions. Redis doesn't need any special initialization steps. However, in this article you can review the following tables to understand the hardware requirements of Redis for different environments.

Development environment

Requirement	Description
Redis Version	7.2 (Note: Previous releases of GWS 8.6 required Redis 6)
Cluster Setup	Optional
Hardware Requirements	Moderate resources, such as 1-2 CPU cores, 1-2 GB RAM.
Networking	Localhost or internal network access.
Persistence	Optional but can be disabled for faster development iterations.
Security	Basic authentication and authorization mechanisms may be implemented.
High Availability	Not critical in development, single-instance setup is acceptable.
Scalability	Scaling considerations not a priority, single-instance setup is sufficient.

Production environment

For production environments, each data center should have the following setup:

Requirements	Description
Redis Version	7.2 (Note: Previous releases of GWS 8.6 required Redis 6)
Cluster Setup	Yes >= 3 Master nodes (Must be an odd number of nodes)
Replicas	1 Per Node
Hardware Requirements	4 cores, 4 or 8 GB RAM.
Networking	Accessible within the data center network, firewall rules configured as per security policies.
Persistence	Recommended for data durability, using either RDB snapshots or AOF logs.

Requirements	Description
Security	Robust authentication and authorization mechanisms in place, SSL/TLS encryption for data in transit.
High Availability	Setup with master-slave replication for failover in case of node failure.
Scalability	Ability to scale horizontally by adding more Redis instances or vertically by upgrading hardware resources.

Elasticsearch

Web Services uses Elasticsearch — an open-source, full-text search engine with a RESTful web interface — to store real-time statistics and facilitate search capabilities. Real-time statistics reflect the current state of the object (User, Queue, Skill).

For new Web Services and Applications 8.6 deployments, you must set up a new cluster of Elasticsearch nodes that is separate from your Web Services nodes. See Configuring Web Services to use an Elasticsearch cluster for details.

Configuring Web Services to use an Elasticsearch cluster

You can configure Web Services to work with an Elasticsearch cluster by completing the steps below.

Prerequisites

- You have deployed and configured a cluster of Elasticsearch nodes. Refer to the Elasticsearch documentation for details. **Note:** Genesys recommends that you use the latest stable 8.x version of Elasticsearch.

Start
Complete the following steps for each Web Services node:

1. Web Services keeps Elasticsearch in sync with Configuration Service and removes statistical data for deleted objects by performing index verification. By default, Web Services runs index verification when its node is started, but you can also configure scheduled index verification by setting **enableScheduledIndexVerification** to `true` in the elasticSearchSettings option. By default, the index verification takes place every 720 minutes (12 hours), but you can change this timing by setting **indexVerificationInterval** in elasticSearchSettings. **Note:** Genesys recommends that you only configure one Web Services node for index verification to avoid excessive requests to Elasticsearch.
2. If your deployment uses statistics, make sure you complete the reporting configuration steps, including setting the nodeId.
3. Set the TransportClientSettings in the **elasticSearchSettings** section. Web Services connect to the Elasticsearch cluster configured under the transport client.
4. Set the enableElasticSearchIndexing option to `true`. Once this option is enabled Web Services writes and reads statistics values to and from Elasticsearch.
5. Set the elasticSearchSettings option to appropriate values for your environment.
6. Set the enableSecurityFeatures option to `false`.

End

Using GWS without Elasticsearch

Web Services and Applications 8.6 can optionally be deployed without Elasticsearch. This disables functionality of select GWS APIs and should only be considered after a technical consultation with Genesys support.

Next Step

- Back to Configuring features

Observability

Genesys Web Services 8.6 provides deployment instructions and scripts which you can use when installing and configuring Observability based on Grafana's open source software solution.

The benefits of using this Observability Solution are:

1. Help with root cause analysis.
2. Help with resilience and reliability (MTTR, uptime, and so on).
3. Help with faster and more informed feedback loops (think: continuous improvement).
4. Provide centralized and searchable storage for logs and metrics for Genesys Web Services 8.6.
5. Enable customers to directly share relevant logs and metrics with Genesys Customer Support to help with troubleshooting during incidents happening in an on-premise environment.

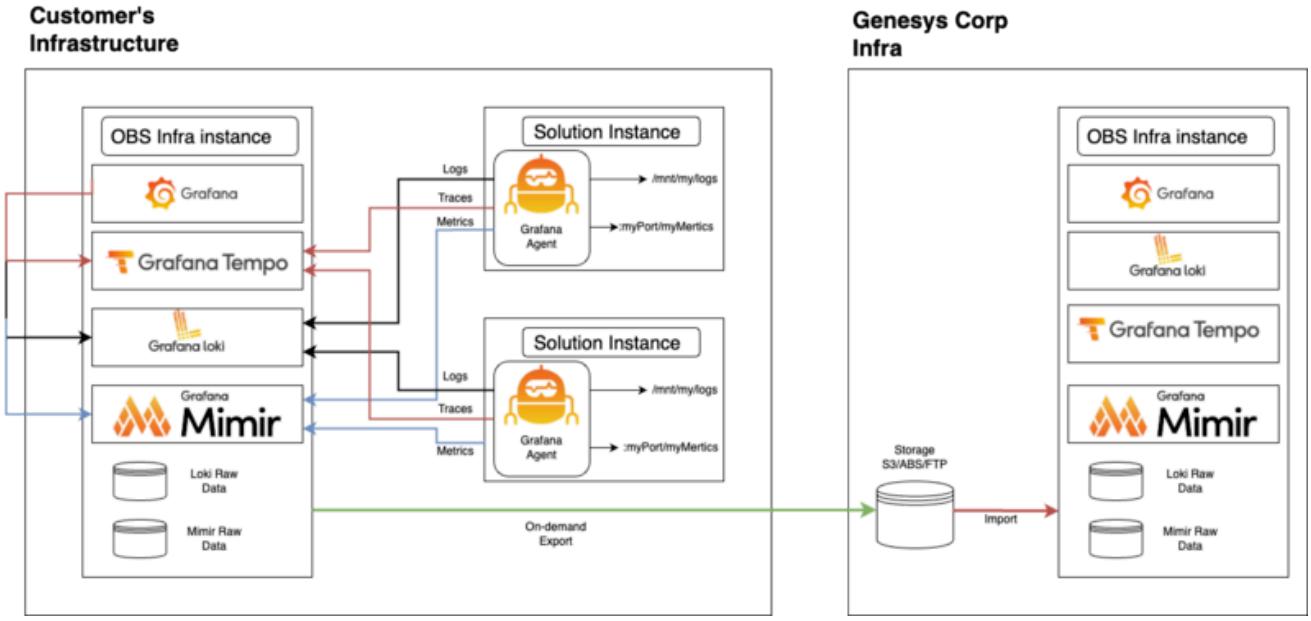
The Observability Solution is founded upon Grafana’s Observability Stack (open source) which consists of the following components:

- **Grafana** - analytics and visualization
- **Mimir** - storage for metrics
- **Loki** - log aggregation backend
- **Tempo** – tracing backend
- **Grafana Agent** - vendor-neutral telemetry collector

Architecture

The Observability Solution architecture consists of:

- OBS Core Stack (Grafana, Loki, Mimir, Tempo) running on a single designated instance.
- Grafana Agents configured on instances where Genesys’ software is running and sends data to the OBS Core Stack.
- On-demand export from OBS Core Stack to Genesys (the nature of "On-demand" is completely controlled by customer).



Observability platform installation

> **Important**
>
> Genesys has provided deployment instructions and scripts which install the Grafana platform with Genesys Web Services 8.6. However, Genesys cannot provide support for any issues encountered with the Grafana platform.

> **Important**
>
> In order to ensure you are using the latest instructions and to gain access to the RPM package referenced below, refer to the Observability installation package included in Genesys Web Services 8.6.
>
> **OBS_instance/README.md**

Installation procedure

Complete the following steps to install Grafana.

1. Get the repository keys.

```
wget -q -O gpg.key https://rpm.grafana.com/gpg.key
```

```
sudo rpm --import gpg.key
```

2. Create **/etc/yum.repos.d/grafana.repo** with the following parameters.

```
[grafana]
name=grafana
baseurl=https://rpm.grafana.com
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://rpm.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

3. Install packages.

```
sudo dnf install grafana-enterprise loki mimir tempo
```

4. Update the loki configuration file **(/etc/loki/config.yml)** with the following parameters:

```
# /etc/loki/config.yml
auth_enabled: false
server:
  http_listen_port: 3100
  grpc_listen_port: 9096
  grpc_server_max_recv_msg_size: 10000000
  grpc_server_max_send_msg_size: 10000000
common:
  instance_addr: 127.0.0.1
  path_prefix: /tmp/loki
  storage:
    filesystem:
      chunks_directory: /tmp/loki/chunks
      rules_directory: /tmp/loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory
query_range:
  results_cache:
    cache:
      embedded_cache:
        enabled: true
        max_size_mb: 100
schema_config:
  configs:
    - from: 2022-11-30
      store: tsdb
      object_store: filesystem
      schema: v12
      index:
        prefix: index_
        period: 24h

ruler:
  alertmanager_url: http://localhost:9093
#### custom part
compactor:
  working_directory: /tmp/loki/retention
  shared_store: filesystem
  compaction_interval: 10m
  retention_enabled: true
  retention_delete_delay: 2m
```

```
  retention_delete_worker_count: 150
limits_config:
  retention_period: 7d
  ingestion_rate_mb: 7
  ingestion_burst_size_mb: 20
```

5. Update the Mimir configuration file **(/etc/mimir/config.yml)** with the following parameters:

```
# /etc/mimir/config.yml

multitenancy_enabled: false
target: all,alertmanager,overrides-exporter
server:
  http_listen_port: 9009
  # Configure the server to allow messages up to 100MB.
  grpc_server_max_recv_msg_size: 104857600
  grpc_server_max_send_msg_size: 104857600
  grpc_server_max_concurrent_streams: 1000
ingester:
  ring:
    # The address to advertise for this ingester.  Will be autodiscovered by
    # looking up address on eth0 or en0; can be specified if this fails.
    instance_addr: 127.0.0.1
    # We want to start immediately and flush on shutdown.
    min_ready_duration: 0s
    final_sleep: 0s
    num_tokens: 512
    # Use an in memory ring store, so we don't need to launch a Consul.
    kvstore:
      store: memberlist
    replication_factor: 1
blocks_storage:
  tsdb:
    dir: /tmp/mimir/tsdb
  filesystem:
    dir: ./data/tsdb
compactor:
  data_dir: /tmp/mimir/compactor
ruler_storage:
  backend: local
  local:
    directory: /tmp/mimir/rules
##### custom part
limits:
  compactor_blocks_retention_period: 2w
  max_global_series_per_user: 0
  ingestion_rate: 1000000
  ingestion_burst_size: 20000000
memberlist:
  join_members: [ localhost ]
```

6. Update the Tempo configuration file **(/etc/tempo/config.yml)** with the following parameters.

```
#/etc/tempo/config.yml
server:
  http_listen_port: 3200
  grpc_listen_port: 9097
query_frontend:
  search:
    duration_slo: 5s
    throughput_bytes_slo: 1.073741824e+09
  trace_by_id:
    duration_slo: 5s
```

```
distributor:
  receivers:
    jaeger:
      protocols:
        thrift_http:
        grpc:
        thrift_binary:
        thrift_compact:
    zipkin:
    otlp:
      protocols:
        http:
        grpc:
    opencensus:

ingester:
  max_block_duration: 5m
  #enable_inet6: false

compactor:
  compaction:
    block_retention: 24h
  #enable_inet6: false

metrics_generator:
  registry:
    external_labels:
      source: tempo
      cluster: docker-compose
    #enable_inet6: false
  storage:
    path: /tmp/tempo/generator/wal
    remote_write:
      - url: http://localhost:9009/api/v1/write
        send_exemplars: true
storage:
  trace:
    backend: local
    wal:
      path: /tmp/tempo/wal
    local:
      path: /tmp/tempo/blocks
overrides:
  defaults:
    global:
      max_bytes_per_trace: 10000000
    metrics_generator:
      processors: [service-graphs, span-metrics] # enables metrics generator
```

7. Create Grafana's default data source file **(/etc/grafana/provisioning/datasources/grafana-provisioning-datasources.yaml)** with the following parameters:

```
# goes to /etc/grafana/provisioning/datasources/grafana-provisioning-datasources.yaml
apiVersion: 1
datasources:
  - name: Mimir
    uid: mimir
    type: prometheus
    access: proxy
    orgId: 1
    url: http://localhost:9009/prometheus
    version: 1
    editable: true
```

```
    jsonData:
      httpHeaderName1: 'X-Scope-OrgID'
      alertmanagerUid: 'alertmanager'
    secureJsonData:
      httpHeaderValue1: 'demo'
    isDefault: true
  - name: Mimir Alertmanager
    uid: alertmanager
    type: alertmanager
    access: proxy
    orgId: 1
    url: http://localhost:9009/
    version: 1
    editable: true
    jsonData:
      httpHeaderName1: 'X-Scope-OrgID'
      implementation: 'cortex'
    secureJsonData:
      httpHeaderValue1: 'demo'
  - name: Loki
    uid: loki
    editable: true
    type: loki
    access: proxy
    url: http://localhost:3100
    jsonData:
      derivedFields:
        - datasourceUid: tempo
          matcherRegex: "traceId\":\"(\\w+)\""
          name: traceId
          url: '$${__value.raw}'
          urlDisplayLabel: ''
        - datasourceUid: tempo
          matcherRegex: "trace_id\":\"(\\w+)\""
          name: trace_id
          url: '$${__value.raw}'
          urlDisplayLabel: ''
      timeout: 125
  - name: Tempo
    uid: tempo
    editable: true
    type: tempo
    access: proxy
    url: http://localhost:3200
```

8. Update the file **/etc/grafana/provisioning/dashboards/sample.yaml** with the following parameters:

```
# /etc/grafana/provisioning/dashboards/sample.yaml
apiVersion: 1
providers:
 - name: 'default'
   orgId: 1
   folder: ''
   folderUid: ''
   type: file
   options:
     path: /var/lib/grafana/dashboards
```

9. Create a folder **/var/lib/grafana/dashboards**.
10. Copy attached dashboards files from the path: **OBS_instance/dashboards** to **/var/lib/grafana/dashboards**.

11. Copy attached recording rules to **/tmp/mimir/rules/anonymous**.
12. Enable and start services.

```
systemctl enable loki
systemctl start loki
systemctl enable mimir
systemctl start mimir
systemctl enable tempo
systemctl start tempo
systemctl enable grafana-server
systemctl start grafana-server
```

13. Check the statuses of the newly deployed services.

```
systemctl status loki
systemctl status mimir
systemctl status tempo
systemctl status grafana-server
```

14. Check the accessibility of Grafana UI using the following URL and update the default password:
    https://your_host_ip:3000/

Grafana Agent installation

The following steps must be repeated for any host which has Genesys Web Services 8.6 deployed.
grafana_agent_static/README.md

1. Copy the **.tar.gz** package to the nodes where the agent needs to be installed.
2. Untar the archive.
3. From the **grafana_agent_static/config** folder, run the **agent_install.sh** script with the following two arguments.
    1. the hostname of Loki/Mimir/Grafana instance
    2. the path to folder with GWS logs location, that is, /logs:

```
ls -l /logs
total 16
drwxrwxr-x 2 genesys genesys 16384 Mar  5 14:20 gws-api-v2
drwxrwxr-x 2 genesys genesys    38 Mar  5 11:28 gws-service-platform
```

4. Script should be executed with root permissions, that is, **sudo ./agent_install.sh <OBS_HOSTNAME_OR_IP> <LOG_FOLDER>**

For example, **sudo ./agent_install.sh d-pat-u1040.us.int.genesyslab.com /mnt/logs/dir_with_logs_from_gws_components/**

Support for Interaction Server Cluster

Web Services supports Interaction Server Cluster deployment for load balancing. For more information on cluster deployments, suggested configurations, and the configuration procedure, see Interaction Server Cluster.

Special considerations

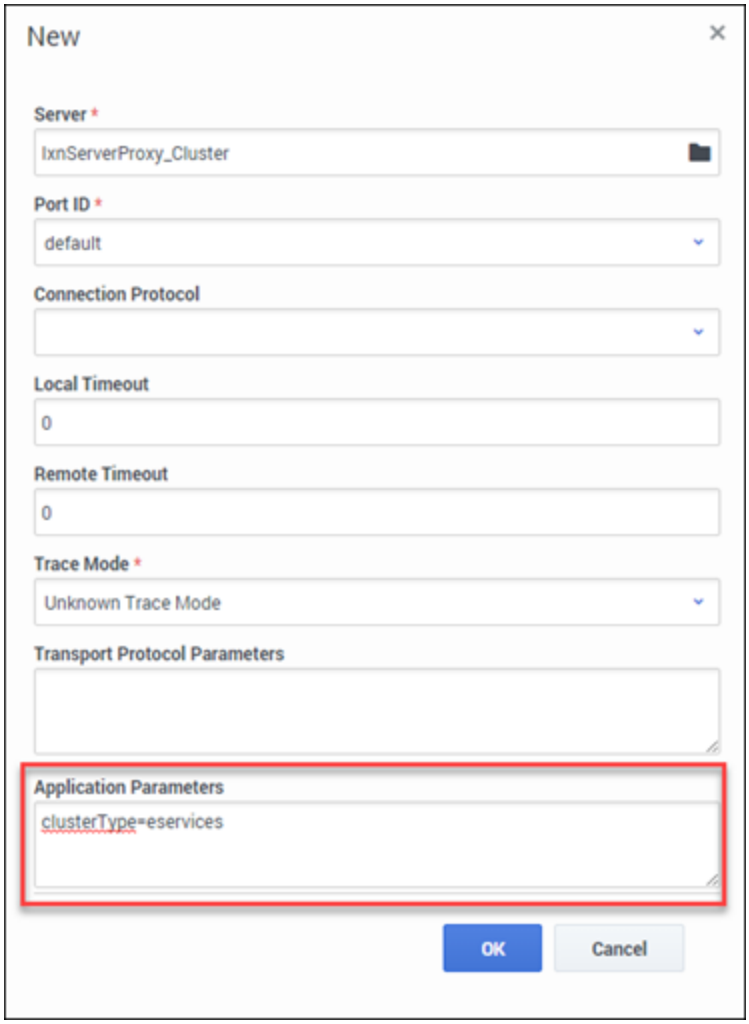
The Cloud Cluster can have one of the following connections:

- Direct connections to Interaction Server Pair
- Connection to an Application Cluster that contains a list of Interaction Server Proxies

The Cloud Cluster cannot support both connection options listed above.

When you configure the connection, the **Application Parameters** field must include the following string:

```
clusterType=eservices
```


New
Server *
IxnServerProxy_Cluster
Port ID *
default
Connection Protocol
Local Timeout
0
Remote Timeout
0
Trace Mode *
Unknown Trace Mode
Transport Protocol Parameters
Application Parameters
clusterType=eservices
OK
Cancel

Support for Universal Contact Server Cluster

Web Services supports a Universal Contact Server Cluster deployment for load balancing. For more information on cluster deployments, suggested configurations, and the configuration procedure, please see Universal Contact Server Cluster and see Creating the UCS 9.1 Cluster Application Object.

Special Considerations

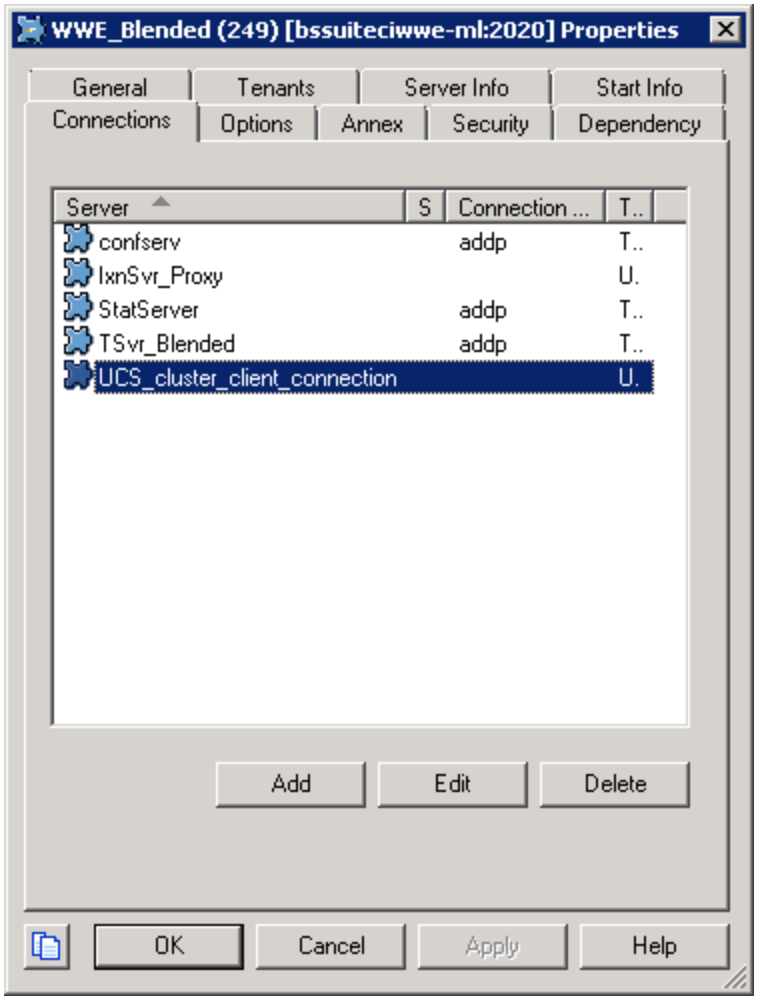
The Cloud Cluster can have one of the following connections:

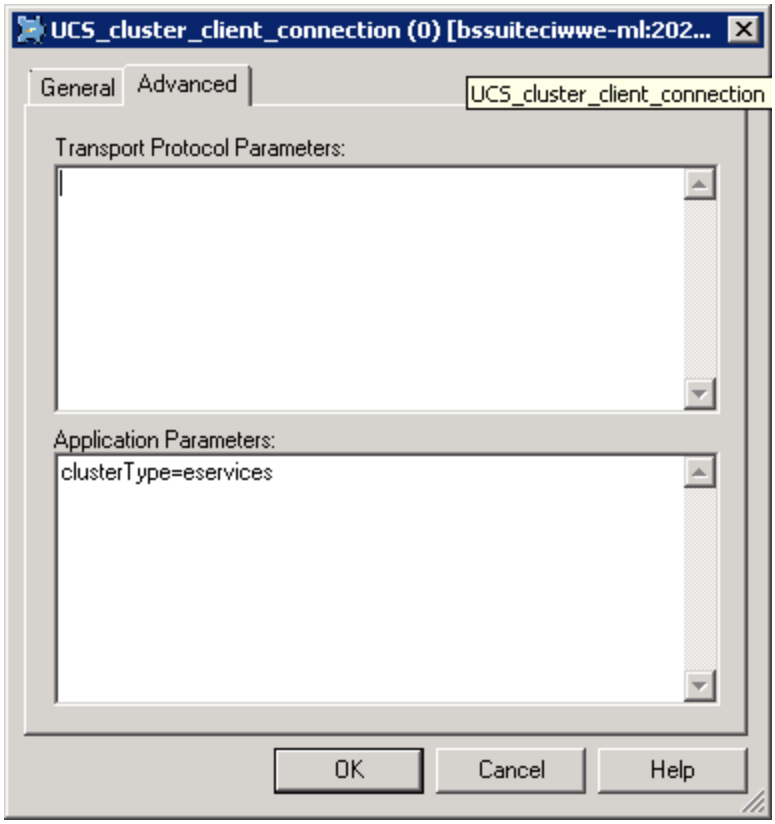
- Direct connections to Universal Contact Server pair
- Connections to an Application Cluster that contains a list of Universal Contact Servers 9.x

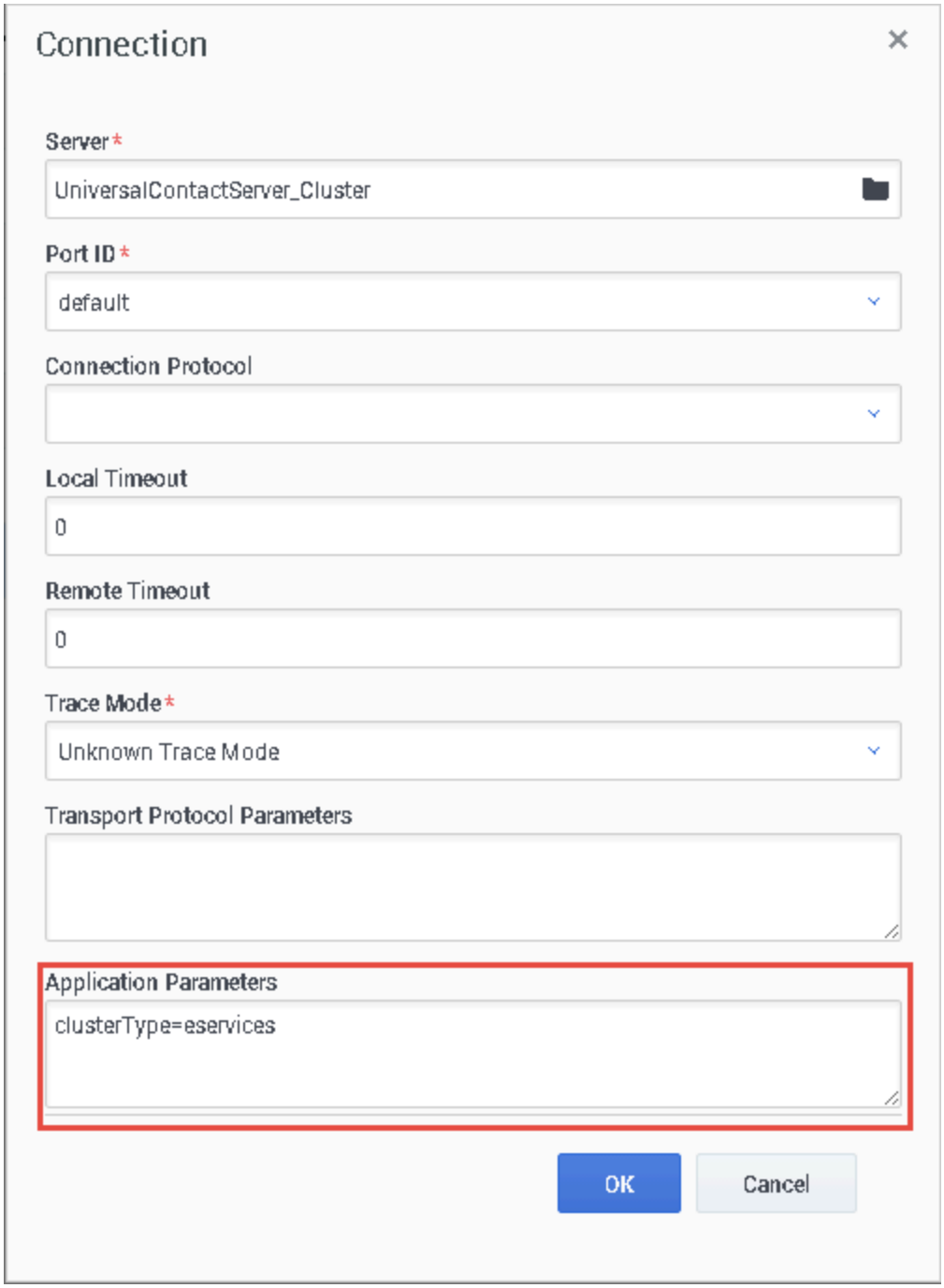
The Cloud Cluster cannot support both connections listed above.

When you configure the connections, the **Applications Parameters** field must include the following string:

```
clusterType=eservices
```


WWE_Blended (249) [bssuiteciwwe-ml:2020] Properties
General
Tenants
Server Info
Start Info
Connections
Options
Annex
Security
Dependency
Server
S
Connection ...
T..
confserv
addp
T..
IxnSvr_Proxy
U.
StatServer
addp
T..
TSvr_Blended
addp
T..
UCS_cluster_client_connection
U.
Add
Edit
Delete
OK
Cancel
Apply
Help


UCS_cluster_client_connection (0) [bssuiteciwwe-ml:202...
General
Advanced
UCS_cluster_client_connection
Transport Protocol Parameters:
Application Parameters:
clusterType=eservices
OK
Cancel
Help


Connection
Server*
UniversalContactServer_Cluster
Port ID*
default
Connection Protocol
Local Timeout
0
Remote Timeout
0
Trace Mode*
Unknown Trace Mode
Transport Protocol Parameters
Application Parameters
clusterType=eservices
OK
Cancel

Configuring GWS API Service Settings

As part of Deploying the web application, you created the **application.yaml** file (or Web Services created it for you). To configure basic Web Services and Applications settings, you need to update the **application.yaml** file on each of your Web Services nodes. In later topics, you'll learn more about modifying this file to configure additional features and security. For now, review the contents below for details about each section in the **application.yaml** configuration file.

> **Important**
>
> As the GWS API Service is not required for OCX ONLY deployments, no configuration is necessary.

Logging settings

The purpose of the **logging** section is to tell Web Services where to find the **logback.xml** file you created (or Web Services created for you) as part of Deploying the web application and where to save logs.

The **application.yaml.sample** file includes the following default **logging** section:

```
logging:
  file: [ToBeChanged: Log file name, will be stored in ${LOG_FILE} which may be used in
logback.xml]
  path: [ToBeChanged: Log folder, will be stored in ${LOG_PATH} which may be used in
logback.xml]
```

See logging for details about all supported configuration settings for this section.

Jetty settings

You use the **jetty** section of the **application.yaml.sample** file to tell Web Services how Jetty should behave. The **application.yaml.sample** file includes the following default **jetty** section:

```
jetty:
  port: 8080
```

See jetty for details about supported configuration settings for this section.

Optional: Postgres/MSSQL Cluster settings

```
# DB Settings for local database
db:
  host: [ToBeChanged: DB_HostName|DB_IP_Address]
  port: [ToBeChanged: DB_Port]
  dbType: [ToBeChanged: “postgres”| “mssql”] <dbType is mandatory for MS-SQL db  and optional
for Postgres, Use dbtype as "mssql" for MS-SQL db>
  dbName: [ToBeChanged: DB_NAME]
  schema: [ToBeChanged: SCHEMA_NAME]
  username: [ToBeChanged: USER_NAME]
  password: [ToBeChanged: USER_PASSWORD]

# DB Settings for connectivity to primary database (for write operations initiated from
secondary Data Center)
db-write-only:
  host: [ToBeChanged: DB_HostName|DB_IP_Address]
  port: [ToBeChanged: DB_Port]
  dbType: [ToBeChanged: “postgres”| “mssql”] <dbType is mandatory for MS-SQL db  and optional
for Postgres, Use dbtype as "mssql" for MS-SQL db>
  dbName: [ToBeChanged: DB_NAME]
  schema: [ToBeChanged: SCHEMA_NAME]
  username: [ToBeChanged: USER_NAME]
  password: [ToBeChanged: USER_PASSWORD]
```

Redis settings

```
# Redis Settings
spring:
  data:
    redis:
      cluster:
        nodes: host1:port1, host2:port2  [ToBeChanged: comma separated list of
REDIS_HOST:REDIS_PORT]
      ssl:
        enabled: ${REDIS.TLS:false}
      password: ${REDIS.PASSWORD:}
      username: <password>
      tls:
        trustStorePath: ${REDIS.TRUSTSTORE.PATH:}
        trustStorePassword: ${REDIS.TRUSTSTORE.PASSWORD:}
        verifyPeer: ${REDIS.VERIFY.PEER:true}
```

or set environment variables: REDIS_TLS REDIS_PASSWORD REDIS_TRUSTSTORE_PATH REDIS_TRUSTSTORE_PASSWORD REDIS_VERIFY_PEER

Platform settings

```
# Platform Settings
    platformSettings:
        platformServiceUrl: http://localhost:8092[ToBeChanged:GWS_SERVICE_PLATFORM_ADDRESS]
```

8092 is the default port but could also be Load Balancer, if used.

Server settings

This section provides the core settings Web Services needs to run your node.

The **application.yaml.sample** file includes the following default **serverSettings** section:

```
serverSettings:

  # OPS account
  opsUserName: [ToBeChanged: <OPS_USER_NAME>]
  opsUserPassword: [ToBeChanged: <OPS_USER_PASSWORD>]

  # Configuration Server credentials
  applicationName: Cloud
  applicationType: CFGGenericClient
  cmeUserName: [MatchFromEnvironment.yaml: username]
  cmePassword: [MatchFromEnvironment.yaml: password]

  # Elastic Search
  enableElasticSearchIndexing: [ToBeChanged: "true"|"false"]
  elasticSearchSettings:
   enableScheduledIndexVerification: false
   enableIndexVerificationAtStartUp: false
   transportClient:
     nodes:
       - host: [ToBeChanged: ES_HostName|ES_IP_Address]
         port: [ToBeChanged: ES_Port]

  # Multi regional supporting
  nodePath: [ToBeChanged: node position in cluster, example: /<REGION>/HOST]
  nodeId: [ToBeChangedOrRemoved: unique value in cluster <NODE_ID>]

  # SSL and CA (optional, can be disabled or removed if not used)
  caCertificate: [ToBeChangedOrRemoved: <CA_CERTIFICATE>]
  jksPassword: [ToBeChangedOrRemoved: <JKS_PASSWORD>]

  # SAML (optional, can be disabled or removed if not used)
  samlSettings:
    encryptionKeyName: [ToBeChangedOrRemoved: <SAML_ENCRYPTION_KEY_NAME>]
    signingKeyName: [ToBeChangedOrRemoved: <SAML_SIGNING_KEY_NAME>]
    identityProviderMetadata: [ToBeChangedOrRemoved: <SAML_IDENTITY_PROVIDER_METADATA>]
    serviceProviderEntityId: [ToBeChangedOrRemoved: <SAML_SERVICE_PROVIDER_ENTITY_ID>]
    encryptionKeyPassword: [ToBeChangedOrRemoved: <SAML_ENCRYPTION_KEY_PASSWORD>]
    signingKeyPassword: [ToBeChangedOrRemoved: <SAML_SIGNING_KEY_PASSWORD>]
    tlsKeyName: [ToBeChangedOrRemoved: <SAML_TLS_KEY_NAME>]
    tlsKeyPassword: [ToBeChangedOrRemoved: <SAML_TLS_KEY_PASSWORD>]
    responseSkewTime: [ToBeChangedOrRemoved: <SAML_RESPONSE_SWEW_TIME>]
    wantAssertionSigned: [ToBeChanged: "true"|"false"]

  # CORS
  crossOriginSettings:
    allowedOrigins: [ToBeChangedOrRemoved: <CROSS_ALLOWED_ORIGINS>]

  # Auth Settings
  auth:
        secret: [Must be configured and must match secret from environment variable]

  # Multimedia Disaster Recovery
  drMonitoringDelay: [ToBeChangedOrRemoved: <DR_MONITORING_DELAY>]
```

Note: The secret length must be at least 32 characters.

Make sure that you update all settings marked as `[ToBeChanged]`. You must also do the following:

- Set the applicationName to the name of the application you created in Configuring the Web Services applications — for example, `WS_Node`.

See serverSettings for details about supported configuration settings for this section.

Partitioned cookie settings

If your browser support Cookies Having Independent Partitioned State (CHIPS) and if you want to opt-in to use CHIPS (adding Partitioned cookie attribute), add the following configuration in the **application.yaml** file.

```
jetty:
 cookies:
   partitioned: true
```

> **Important**
>
> This configuration is mandatory for browsers where third-party cookies are disabled but they should be shared across domains. For more details, see How We're Protecting Your Online Privacy - The Privacy Sandbox

.

On-premises settings

Example on-premises settings:

```
onPremiseSettings:
    countryCode: US
```

The **application.yaml.sample** file doesn't include a default **onPremiseSettings** section, so you'll need to add it yourself.

See onPremiseSettings for details about all supported configuration settings for this section.

Tuning the Web Services host performance

Complete the steps below on each Web Services node to tune the performance of the host environment.

1. To optimize TCP/IP performance, you can run the following commands:

```
sudo sysctl -w net.core.rmem_max=16777216
sudo sysctl -w net.core.wmem_max=16777216
sudo sysctl -w net.ipv4.tcp_rmem="4096 87380 16777216"
sudo sysctl -w net.ipv4.tcp_wmem="4096 16384 16777216"
sudo sysctl -w net.core.somaxconn=4096
sudo sysctl -w net.core.netdev_max_backlog=16384
sudo sysctl -w net.ipv4.tcp_max_syn_backlog=8192
sudo sysctl -w net.ipv4.tcp_syncookies=1
sudo sysctl -w net.ipv4.tcp_congestion_control=cubic
```

2. After providing for some means of starting Jetty, determine the user or group that will start Jetty and increase the file descriptors available to that user or group by adding the following to the **/etc/security/limits.conf** file:

```
<user_name>               hard nofile        100000
<user_name>               soft nofile        100000
```

Where <user_name> is the name of the user or group that is starting Jetty.

SameSite cookies

To handle sameSite cookie attribute, you must configure options for both Jetty and CometD.

If the value of **SameSite** is set to None, Chrome browser also checks if the Secure cookie attribute is present, and if not, then warn user.

To mitigate this issue, make the following edits in application.yaml:

```
...
jetty:
  ...
  cookies:
    ...
    secure: true
    sameSite: None
...
serverSettings:
  ...
  cometDSettings:
    ...
    cookieSecure: true
    cookieSameSite: None
```

> **Important**
> If cookies are configured to be secure, the browser applies them to a secure connection only (https); therefore, these options take effect only if **enableSsl** is set to true.

If the value of **SameSite** is set to Lax or Strict, a secured connection is not required, for example:

```
...
```

```
jetty:
  ...
  cookies:
...
    ...
    httpOnly: true
    secure: false
    sameSite: Lax
...
serverSettings:
  ...
  cometDSettings:
    ...
    cookieHttpOnly: true
    cookieSecure: false
    cookieSameSite: Lax
```

However, it is important to note the following:

- If **SameSite** is set to Lax, the cookie is sent only on same-site requests or by top-level navigation with a safe HTTP method. That is, it will not be sent with cross-domain POST requests or when loading the site in a cross-origin frame, but it will be sent when the user navigates to the site via a standard top-level <a href=...> link.
- If **SameSite** is set to `Strict`, the cookie is never sent in cross-site requests. Even if the user clicks a top-level link on a third-party domain to your site, the browser refuses to send the cookie.

> **Important**
>
> You can choose an insecure connection by specifying a different type of SameSite (Lax or Strict), but this means that it will be impossible to embed Workspace Web Edition in an iframe or use it for any other cross-domain integrations. For example, applications like Genesys CRM Workspace/Adapter will not work with this configuration.

Configuring Statistics indexing

```
Add nodeId: /<REGION>/HOST
enableElasticSearchIndexing: true
```

Configuring Configuration Objects indexing

```
enableElasticSearchIndexing: true
enableScheduledIndexVerification: true
enableIndexVerificationAtStartUp: true
```

Next step

- Configuring features

Configuring GWS Platform Service Settings

Environment Variables

Name	Mandatory	Default Value	Description
GWS_COMMON_AUTH_SECRET	Yes		Specifies the secret that should be used for authentication purpose for inter-service communication. For more details, please refer to "serverSettings.auth.secret" in GWS API Service configuration options and/or the OCX configuration option "secret".
GWS_COMMON_LOCATION	No	/	Specifies the region or datacenter name that service should serve for. For more details, please refer to "serverSetting.nodePath" in GWS API Service configuration options.
GWS_DATABASE_UPDATE_POLLING_TIMEOUT	No	1000	Specifies in milliseconds how often service should check for changes in ConfigDB.
GWS_ENV_CONFIG_FILE	No	./environment.yaml	Specifies a location of environment.yaml file that should be used for service initialization.
GWS_EXPORTER_OTLP_TRACES_ENDPOINT	No	http://localhost:4317	Specifies OpenTelemetry address in URL format that should be used for exporting traces.
GWS_LOGGING_ENABLE_COMPRESSION	No	true	Specifies if compression for stored log files should be applied.
GWS_LOGGING_INTERVAL	No	3600	Specifies in seconds how often log file rotation should be performed.
GWS_LOGGING_MAX_FILE_COUNT	No		Specifies the max

Name	Mandatory	Default Value	Description
			number of log files that should be kept. By default, service doesn't remove any log file.
GWS_LOGGING_MAX_FILE_SIZE	No		Specifies the max log file size when log rotation should be performed in addition to configured interval (GWS_LOGGING_INTERVAL). By default, log rotation is done by timer only.
GWS_LOGGING_MAX_HISTORY	No		Specifies in seconds how long the service's log files should be kept. By default, they are kept forever.
GWS_LOGGING_PATH	No	/var/log/gws-service-platform	If specified, enables writing logs into a file instead of STDOUT/ STDERR (in such case, logs are accessible through JournalD). Specifies the path where logs should be stored.
GWS_LOGGING_FILE	No	gws-service-platform.log	Specifies the log file name.
GWS_LOG_LEVEL	No	INFO	Specifies service logging level. Allowed values: [ERROR, WARN, INFO, DEBUG, TRACE]
GWS_SERVER_HOST	No	0.0.0.0	Specifies the host name or IP address of network interface that should be used by service to open API listening port. Default value "0.0.0.0" that means open listing port on all network interfaces.
GWS_SERVER_PORT	No	8092	Specifies the port number that should be used as API listening port.
GWS_DATABASE_DATA_SOURCE_NAME	No	MSSQL: MicrosoftODBC-18 Oracle: OracleODBC-23	Specifies the custom DSN that should be used by service instead of default.
GWS_DATABASE_USE_TNSNAMES	No	false	When set true, the database parameter

Name	Mandatory	Default Value	Description
			from environment.yaml is treated as the Oracle SID (GWS Platform service supports Oracle SIDs from tnsnames.ora file) and host/port are not used.

environment.yaml file

Note: password field values in environment.yaml can be encrypted. To encrypt, pass a command line argument following --encode parameter. Example:

./gws-service-platform --encode "testPa33W0rd"

FE9EDB4BF51ED570C18C2CC2F530468D

General

```
appName: Cloud # Specifies the application name from ConfigServer that GWS should use
connectionProtocol: addp # Specifies if ADDP protocol should be used along with ConfigServer
connection
localTimeout: [ToBeChanged: timeout] # Specifies a local timeout for ADDP protocol
password: [ToBeChanged: password] # Specifies the account password that GWS should use for
connecting to ConfigServer
remoteTimeout: [ToBeChanged: timeout] # Specifies a remote timeout for ADDP protocol
tenant: Environment # Specifies default environment in ConfigServer (reserved for future
usage)
tlsEnabled: [ToBeChanged: true or false] # Indicates if TLS should be used while establishing
connection to ConfigServer
traceMode: [ToBeChanged: CFGTMBoth/CFGTMLocal/CFGTMRemote/CFGTMNone] # Specifies what mode
should be used for ADDP protocol
username: [ToBeChanged: username] # Specifies the account name that GWS should use for
connecting to ConfigServer
```

ConfigServer

```
locations: [ToBeChanged: /<REGION>] # Specifies the name of datacenter for which
configuration is set.
initDb: false # Indicates if database schema should be provisioned (reserved for future usage)
primaryAddress: [ToBeChanged: ConfigServer/CSProxy address]
primaryPort: [ToBeChanged: ConfigServer/CSProxy port]
readFromDb: true # Indicates if read requests should be handled with data from database
readOnly: true # Indicates if connection to ConfigServer shouldn't be used for write
operations (reserved for future usage)
```

Database

```
database: [ToBeChanged: database] # Specifies the name of ConfigServer's database or
ServiceName for Oracle DB
host: [ToBeChanged: host] # Specifies the address of ConfigServer's database
password: [ToBeChanged: password] # Specifies database account password
poolSize: 3 # Specifies the size of database connections pool
port: [ToBeChanged: port] # Specifies the port of ConfigServer's database
type: [ToBeChanged: "postgres" | "mssql" | "oracle"] # Specifies the ConfigServer's database
type
```

```
username: [ToBeChanged: username] # Specifies database account name
```

Sample File

```
appName: Cloud
configServers:
  - db:
      database: orcldb.test.com
      host: db.dc1.com
      password: gws
      poolSize: 5
      port: 1521
      type: oracle
      username: gws_test
    initDb: false
    locations: "/DC1"
    primaryAddress: masterCS.com
    primaryPort: 8888
    readFromDb: true
    readOnly: false
  - initDb: false
    locations: "/DC1"
    primaryAddress: ProxyCS.dc1.com
    primaryPort: 8888
    readFromDb: false
    readOnly: true
  - db:
      database: orcldb-replica.test.com
      host: db.dc2.com
      password: gws
      poolSize: 5
      port: 1521
      type: oracle
      username: gws_test
    initDb: false
    locations: "/DC2"
    primaryAddress: ProxyCS.dc2.com
    primaryPort: 8889
    readFromDb: false
    readOnly: true
connectionProtocol: addp
localTimeout: 5
password: pass_test
remoteTimeout: 7
tenant: Environment
tlsEnabled: true
traceMode: CFGTMBoth
username: gws_test
```

Multiple Data Center Deployment

Genesys Web Services 8.6 supports deployment with multiple (two or more) data centers. This section describes this type of deployment.

Overview

Genesys Web Services 8.6 supports deployments in two and three data centers.

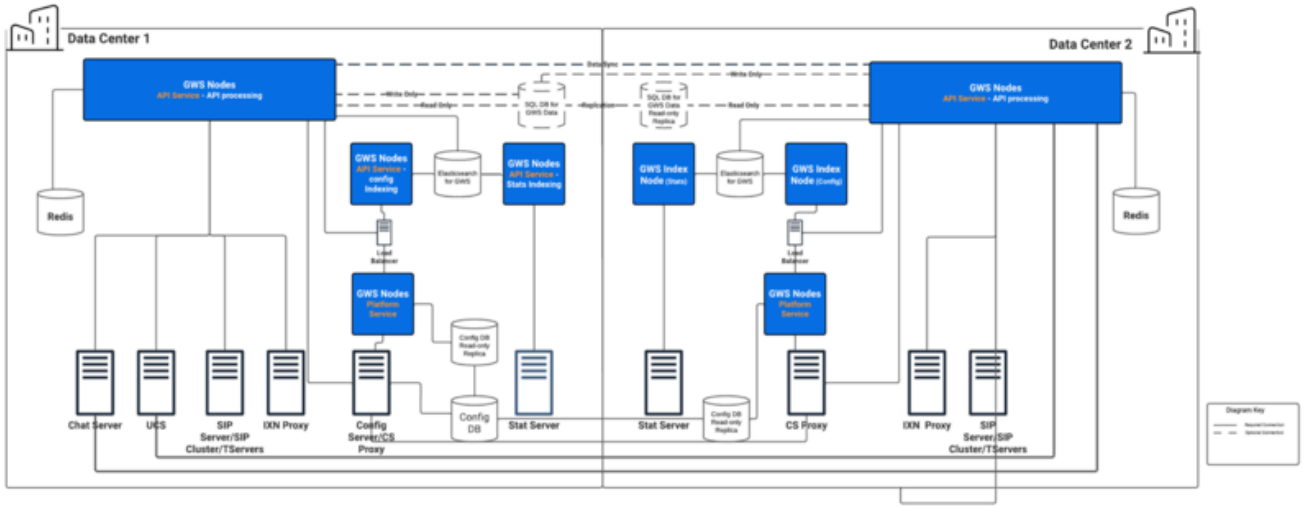
- Typically one data center is designated as the "primary". This is the data center where the primary read/write instances of the Configuration Database and GWS Database (if used/deployed).
- The other data center(s) will each have a deployment that mirrors the primary data center in terms of the GWS Nodes, Redis cluster, and Elasticsearch. Note that Redis and Elasticsearch operate independently in each data center.
- GWS in the primary data center should always be active (except for maintenance or disaster scenarios). GWS in the secondary data center(s) can be active or standby.

Architecture

A typical GWS data center in a multiple data center deployment consists of the following components:

- 2+ GWS API Service nodes
- 2+ GWS Platform Service nodes
- 2 GWS Stat Index nodes running in primary/secondary mode
- 2 GWS Configuration Index nodes running in primary/secondary mode
- 3+ Redis Cluster primary nodes
- 3+ Elasticsearch nodes

The following diagram illustrates the architecture of this sample multiple data center deployment.



Medium deployments may also be deployed across multiple data centers.

Note the following restrictions of this architecture:

- Each data center must have a dedicated list of Genesys servers, such as Configuration Servers, Stat Servers, and T-Servers.
- Each GWS data center must have its own standalone and dedicated Elasticsearch Cluster.
- Each GWS data center must have its own standalone and dedicated Redis Cluster.
- The GWS node identity must be unique across the entire Cluster.

Incoming traffic distribution

GWS relies on a third-party reverse proxy to enable traffic distribution both within and across data centers. The reverse proxy provides session stickiness based on association with sessions on the backend server; this rule is commonly referred to as *Application-Controlled Session Stickiness*. In other words, when a GWS node creates a new session and returns `Set-Cookie` in response, the load-balancer issues its own stickiness cookie. GWS uses a `JSESSIONID` cookie by default, but this can be reconfigured by using the following option in the **application.yaml** file:

```
jetty:
  cookies:
    name: <HTTP Session Cookie Name>
```

Business continuity (smart failover)

Depending on the components that are functional/non-functional in a given data center (primary or secondary), the behavior will be different when the agent logs back into the other data center. All these behaviors are dependent on the agent via their desktop application (WWE or custom application) re-logging into the other data center. This re-login can be handled in two different ways:

1. The agent receives the appropriate error from the application and terminates that instance of the application, start a new instance and re-logs in the application but connected to the GWS cluster in the other data center.
2. The desktop application detects the error situation and then automatically navigates the agent through the necessary step to re-log them into the other data center. In WWE, this process is called smart failover.

The following are the common conditions that could cause an agent to re-log into another data center:

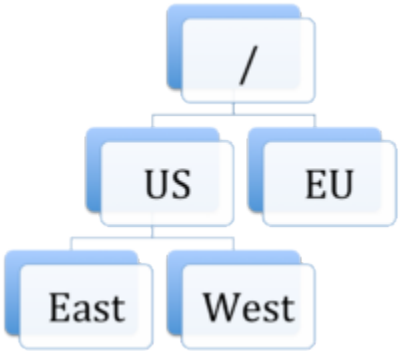
Data Center condition	What capabilities are available when logged into the other data center
GWS/WWE is down in either one of the data centers.	Agent stays in current state. The status of active interactions in a failing data center is as follows: • **Voice** - All existing interactions are active. • **Digital** - All existing interactions are active. The **Contact History** view resets to the Default view.
Primary data center is down (for example, due to a data center network issue).	Agent will go into ready state and is ready for new interactions (all channels). Following is the status of the interactions that were active in the primary data center: • **Voice** - All existing interactions are dropped. • **Digital** - All existing interactions are dropped. The **Contact History** view resets to the Default view.
Secondary data center is down (for example, due to a data center network issue).	Agent will go into ready state and is ready for new voice interactions. Following is the status of the interactions that were active in the primary data center: • **Voice** - All existing interactions are dropped. • **Digital** - All existing interactions are active. The **Contact History** view resets to the Default view.
Voice Service is down in primary or secondary data center.	Agent will go into ready state and is ready for new voice interactions. Following is the status of the interactions that were active in the primary data center: • **Voice** - All existing interactions are dropped. • **Digital** - All existing interactions are active. The **Contact History** view resets to the Default view.
Digital Service is down in the primary region	Agent changes state based on the loss of the digital/social interactions. Following is the status of the interactions that were active in the primary data center: • **Voice** - All existing interactions are active.

	• **Digital** - Existing digital /social interactions are dropped. The **Contact History** view resets to the Default view.

Configuration

This section describes the additional configuration required to set up a multiple data center deployment.

The topology of a GWS Cluster can be considered as a standard directory tree where a leaf node is a GWS data center. The following diagram shows a GWS Cluster with two geographical regions (US and EU), and three GWS data centers (East and West in the US region, and EU as its own data center).



> **Important**
>
> In a Disaster Recovery (DR) environment (for instance, having two separate SIP Servers connected to a WS Cluster Application with different locations specified), it is possible that the **Place** can contain two DNs of the type **Extension**. However, the two DNs must belong to different switches and the API server will be mapped only to one of them during the **StartContactCenterSession** operation. In such cases, the DN visibility depends on the **nodePath** parameter in the **application.yaml** file in the API server and on the `location` attribute of the SIP Server connection in the WS Cluster Application.

Genesys Web Services and Applications

The position of each node inside the GWS Cluster is specified by the mandatory property **nodePath** provided in **application.yaml**. The value of this property is in the standard file path format, and uses the forward slash (/) symbol as a delimiter. This property has the following syntax:

```
nodePath: <path-to-node-in-cluster>/<node-identity>
```

Where:

- <path-to-node-in-cluster> is the path inside the cluster with all logical sub-groups.
- <node-identity> is the unique identity of the node. Genesys recommends that you use the name of the host on which this data center is running for this parameter.

For example:

```
nodePath: /US/West/api-node-1
```

Each server will be aware of servers or nodes in the same path of above. In addition to the configuration options set in the standard deployment procedure, set the following configuration options in **application.yaml** for all GWS nodes to enable the multiple data center functionality:

```
serverSettings:
 nodePath: <path-to-node-in-cluster>/<node-identity>

statistics:
 locationAwareMonitoringDistribution: true
 enableMultipleDataCenterMonitoring: true
```

- Set both **locationAwareMonitoringDistribution** and **enableMultipleDataCenterMonitoring** on all nodes. Without them, the statistics information in a Multiple Data Center environment will not be displayed properly.}}

In addition, set the following options on all Stat nodes:

```
serverSettings:
   elasticSearchSettings:
      enableScheduledIndexVerification: true
      enableIndexVerificationAtStartUp: true
```

GWS API Service node in the secondary Data Center might need to connect to the GWS API Service node in the primary Data Center if users are authenticated against the password stored in the configuration database. This connectivity enables changing the password by a user connected to the secondary Data Center.

The following should be configured in CME or GA:

```
Section “master-configuration-data-center” in Cloud Cluster application options should
contain following options:
location: <location of the primary region>
redirectUri: <URI of primary region LB>
```

If the SQL DB is used for persistent data storage, it must be configured to replicate from the primary Data Center to the secondary Data Center database using the native replication capabilities of the database.

Additionally, the GWS API Service nodes in the secondary Data Center must be configured with a connection to the database in the primary Data Center, see https://docs.genesys.com/Documentation/GWS/8.6.0DRAFT/Dep/ConfigurationPremise.

Limitations

The cross-site monitoring is only supported via Team Communicator, using the WWE option teamlead.monitoring-cross-site-based-on-activity-enabled.
The `My Agents` view is not supported for cross-site operations.

Configuring and enabling Web Services features

You can configure the following Web Services features:

- Reporting with Statistics
- Elasticsearch
- Chat and email conference and transfer through a queue
- Contact availability
- Agent Group availability (for Voice only)

Some features are enabled by default. Other features, such as eService features, are disabled and you can enable the features that are applicable for your deployment. For more information, see Enable features in the Feature Definition file.

> **Important**
> Feature configuration is not required for OCX ONLY deployments.

Next step

- Configuring security

Reporting with Statistics

You can configure Web Services to use, store, and expose statistical data for agents, skills, and queues. This data is used for reporting in Workspace Web Edition and you can also access it through the API. Complete the following steps to set up reporting in Web Services:

1. Configure Elasticsearch to support real-time statistics.
2. Enable reporting.

Enabling reporting

Start

1. Open the **application.yaml** file.
2. Configure the Web Services node by setting the nodeId option to a unique identifier, such as the host name or IP address of the node.
3. Review the statistics configuration options for details about setting the connection to Stat Server. Adjust the default settings for these options as required for your deployment.
4. Save your changes and close the file.
5. Confirm that the **statistics.yaml** file is in present your Web Services home folder. This file defines which statistics Web Services requests from Stat Server.

End

Next step

- Back to Configuring features

Consultation, conference, and transfer through a queue for chat

To ensure consultations, conferences, and transfers through queues are reported correctly for Chat, you must update the **Interaction Subtype** Business Attribute.

Genesys Administrator

Updating the **Interaction Subtype** Business Attribute

Start

1. Navigate to **PROVISIONING > Routing/eServices > Business Attributes**, select **Interaction Subtype**, and click **Edit...**.
2. Select the **Attributes Values** tab and click **New...**.
3. Enter the following:
   - Name: `InternalConferenceInvite`
   - Display Name: `Internal Conference Invite`
4. Click **Save & Close**.

End

Configuration Manager

Updating the **Interaction Subtype** Business Attribute

Start

1. Navigate to **Business Attributes > Interaction Subtype > Attribute Values**.
2. Right-click and select **New > Business Attribute Value**.
3. In the **General** tab, enter the following:
   - Name: `InternalConferenceInvite`
   - Display Name: `Internal Conference Invite`
4. Click **OK**.

End

Next step

- Back to Configuring features

Contact availability

Your Web Services and Applications solution must meet the following requirements to enable contact availability for contact resources of type User in the Contacts API:

- Your environment must include a connection to Stat Server.
- You have enabled statistics monitoring.
- Your **statistics.yaml** file contains the following definitions:

```
---
#internal stats
name: CurrentTargetState
statisticDefinitionEx:
    category: CurrentTargetState
    mainMask: "*"
    subject: DNStatus
    dynamicTimeProfile: "0:00"
    intervalType: GrowingWindow
objectType: AGENT
notificationMode: IMMEDIATE
notificationFrequency: 0
---
name: CurrentAgentState
notificationFrequency: 0
notificationMode: IMMEDIATE
objectType: AGENT
statisticDefinitionEx:
    category: CurrentState
    mainMask: "*"
    subject: DNAction
```

- You have enabled multimedia channel states monitoring (optional).
- The contact must have a device assigned and be logged in; otherwise, Web Services does not include the availability subresource.

Agent Group Availability (for Voice)

Your Web Services and Applications solution must meet the following requirements to enable Agents to view Agent Group availability for Voice in Team Communicator:

- Your environment must include a connection to Stat Server.
- You have enabled statistics reporting.
- Your **statistics.yaml** file contains the following definitions:

```
---
name: TransferAvailability_CurrentReadyAgents
notificationFrequency: 10
notificationMode: IMMEDIATE
objectType: VIRTUAL_AGENT_GROUP
statisticDefinitionEx:
     dynamicFilter: "MediaType=voice"
     category: CurrentNumber
     mainMask: WaitForNextCall
     subject: DNStatus
---
name: TransferAvailability_CurrentReadyAgents
notificationFrequency: 10
notificationMode: IMMEDIATE
objectType: AGENT_GROUP
statisticDefinitionEx:
     dynamicFilter: "MediaType=voice"
     category: CurrentNumber
     mainMask: WaitForNextCall
     subject: DNStatus
---
```

Enabling features in the Feature Definitions file

Some features are enabled by default. Other features, such as eService features, are disabled and you can enable the features that are applicable for your deployment.

The Feature Definitions file contains a list of features that are available for your contact center. By default, all data access APIs and all voice-related functionality is enabled. Therefore, for most voice-only deployments, you do not need to make any changes to the Feature Definitions file.

Procedure

1. Locate the **feature-definitions.json** file, which is located in the **config** folder and then open the file in a text editor.
2. For each feature that you want to enable, set the **autoAssignOnContactCenterCreate** flag to true.
3. Save the file.
   In the following example, the Facebook API is enabled and GWS will attempt to connect to the interaction server

   ```
   { "id": "api-multimedia-facebook", "displayName": "Multimedia Facebook API",
   "description": "API for Multimedia Facebook", "autoAssignOnContactCenterCreate": true }
   ```

Sample Feature Definitions file

```
[
  {
    "id": "api-provisioning-read",
    "displayName": "API Provisioning Read",
    "description": "General provisioning read",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-provisioning-write",
    "displayName": "API Provisioning Write",
    "description": "General provisioning write",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-voice",
    "displayName": "Voice API",
    "description": "API for Voice",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-voice-predictive-calls",
    "displayName": "Voice API - Predictive calls",
```

```
    "description": "Enables predictive calls for a contact center",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-voice-outbound",
    "displayName": "Voice API Outbound",
    "description": "API for Outbound",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-supervisor-agent-control",
    "displayName": "API Supervisor Agent Control",
    "description": "API for Supervisors to Control Agent State",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-supervisor-monitoring",
    "displayName": "API Supervisor Monitoring",
    "description": "API for Supervisors to Monitor Agents",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-multimedia-chat",
    "displayName": "Multimedia Chat API",
    "description": "API for Multimedia Chat",
    "autoAssignOnContactCenterCreate": false
  },
  {
    "id": "api-multimedia-email",
    "displayName": "Multimedia Email API",
    "description": "API for Multimedia Email",
    "autoAssignOnContactCenterCreate": false
  },
  {
    "id": "api-multimedia-facebook",
    "displayName": "Multimedia Facebook API",
    "description": "API for Multimedia Facebook",
    "autoAssignOnContactCenterCreate": false
  },
  {
    "id": "api-multimedia-twitter",
    "displayName": "Multimedia Twitter API",
    "description": "API for Multimedia Twitter",
    "autoAssignOnContactCenterCreate": false
  },
  {
    "id": "api-multimedia-workitem",
    "displayName": "Multimedia Workitem API",
    "description": "API for Multimedia Workitem",
    "autoAssignOnContactCenterCreate": false
  },
  {
    "id": "api-user-account-management-email",
    "displayName": "User Account Management via Email",
    "description": "API for account management via email",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-devices-webrtc",
    "displayName": "WebRTC Support",
    "description": "API for WebRTC provisioning",
    "autoAssignOnContactCenterCreate": true
  },
```

```
  {
    "id": "api-ucs-voice",
    "displayName": "Support UCS for voice",
    "description": "For support contact center in voice",
    "autoAssignOnContactCenterCreate": false
  },
  {
    "id": "api-voice-instant-messaging",
    "displayName": "API Voice Instant Messaging",
    "description": "API for Internal Agent-to-Agent Chat",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-platform-configuration-read",
    "displayName": "Platform Configuration API - read",
    "description": "Low-level configuration API",
    "autoAssignOnContactCenterCreate": true
  },
  {
    "id": "api-platform-configuration-write",
    "displayName": "Platform Configuration API - write",
    "description": "Low-level configuration API",
    "autoAssignOnContactCenterCreate": true
  }
]
```

Configuring security

Web Services adheres to the standards described in the Open Web Application Security Project (OWASP) Top 10 — see the OWASP website for details about the Top 10 — and has adopted several methods of ensuring security, for example:

- Errors are logged locally to prevent information leakage through API requests.
- User sessions have a timeout option.
- Cross Site Request Forgery Protection

Web Services includes additional security configurations that you can use with your installation:

- Transport Layer Security (TLS)
- Security Assertion Markup Language (SAML) authentication
- Cross-Site Request Forgery (CSRF) protection
- Cross-Origin Resource Sharing (CORS) filter

For details about how Web Services handles authentication, see Web Services authentication flow.

Next step

- Starting and testing Web Services and Applications

Transport Layer Security (TLS)

Configuring TLS between Web Services and Configuration Server

Web Services can use a secured Transport Layer Security (TLS) connection mechanism to connect to Configuration Server. When configured, Web Services connects to a secure port on Configuration Server, verifies the server's authority, and encrypts/decrypts network traffic. You can configure secured connections to Configuration Server in the following ways:

- Minimal configuration
- Validate the certificate against the CA

Prerequisites

Before configuring Web Services, make sure the Configuration Server secure port is configured as described in Introduction to Genesys Transport Layer Security in the *Genesys Security Deployment Guide* and that all certificates for server host and the certificate authority are configured and available.

Minimal configuration

Web Services does not check the server's certificate against the Certificate Authority, but all traffic is encrypted. To configure Web Services with minimal configuration, all you need to do is configure a connection to a secured port on Configuration Server. You can do this using **either** of the following methods:

- For the initial connection to Configuration Server, set the **tlsEnabled** option to `true` in the **environment.yaml** file. This creates a secured connection to Configuration Server the first time Web Services starts.
- For an environment that is already configured with Configuration Manager synchronization enabled, you can make changes with Configuration Manager as described in the Genesys Security Deployment Guide.

Validate the certificate against the CA

In order to support the client-side certificate check, Web Services needs the public key for the Certificate Authority (CA). Web Services supports the PEM and JKS key storage formats, but recommends using JKS.

Complete the steps below to validate the certificate against the CA.

> **Important**
> The steps described in this procedure are meant to be an example for developers and should not be used in production. For a production environment, you should follow your own company's security policies for creating and signing certificates.

Start

1. If you plan to use a JKS file, you can generate if from a PEM file by importing the PEM certificate, as shown here:

   ```
   keytool -importcert -file ca_cert.pem -keystore ca_cert.jks
   ```

2. Once you have the **ca_cert.jks** file, place it in a location available from your Web Services host, such as:
   - A local folder on the Web Services host
   - A network share
3. Configure the following options in the serverSettings section of the **application.yaml** file:
   - For a PEM file, set caCertificate to the location of the file. For example:

     ```
     caCertificate: /opt/ca_cert.pem
     ```

   - For a JKS file, set caCertificate to the location of the file and set jksPassword to the password for the key storage. For example:

     ```
     caCertificate: /opt/ca_cert.jks
     jksPassword: pa$$word
     ```

End

For TLS for all other servers, it uses the configuration data from Configuration Server.

Next Step

- Back to Configuring security

SAML authentication

Web Services supports Security Assertion Markup Language (SAML) for single sign-on (SSO) authentication to the Agent Desktop and custom integrations.

Configuring SAML

To enable SAML, make the following configuration changes in the serverSettings section of the **application.yaml** file on each of your Web Services nodes:

Start

1. Set the following options in the SSL and CA section:
   - caCertificate — should point to a JKS key storage that includes the SAML encryption key. See Generating security keys for details.
   - jksPassword — should be the password for the **caCertificate** key storage.
2. Set the following option in the SAML section:
   - samlSettings — the following properties are mandatory:
     - encryptionKeyName
     - signingKeyName
     - identityProviderMetadata
3. Save the changes to the file. Your configuration should look something like this:

```
# SSL and CA
caCertificate: /Users/samluser/Documents/Keys/keystore.jks
jksPassword: password

# SAML
samlSettings:
    serviceProviderEntityId: genesys.staging.GWS
    encryptionKeyName: client
    signingKeyName: client
    identityProviderMetadata: /Users/samluser/Documents/Metadata/idp-metadata.xml
```

4. To activate SAML authentication, append the browser URL for Workspace Web Edition with `?authType=saml`.
5. To enable extended SAML logging, add the following string to **logback.xml** file: `<logger name="org.springframework.security.saml2" level="%LEVEL%"/>`, where valid values for LEVEL are INFO (preferred) or DEBUG.

End

Generating security keys

You can use the keytool utility that comes with the Java SDK to generate a JKS key store. Use the following command:

```
keytool -genkey -keystore <path_to_jks_file> -alias <key_name> -keypass <key_password>
-storepass <store_password> -dname <distinguished_name>
```

If you already have a JKS key store, you can add a key to it by executing the command above with the same file name and the new key name and key password. For example:

```
keytool -genkey -keystore /opt/keystore.jks -alias encryption_key -keypass genesys -storepass
genesys -dname "CN=GWS, OU=R&D, O=Genesys, L=Daly City, S=California, C=US"
```

SAML API mappings between GWS 8.5 and GWS 8.6

The SAML API endpoints in GWS 8.5 must be mapped to its equivalent SAML 2.0 API endpoints in GWS 8.6.

SAML (GWS 8.5)	SAML 2.0 (GWS 8.6)
/saml/login	/saml2/authenticate/gws
/saml/SSO	/login/saml2/sso/gws
/saml/logout	/saml2/logout
/saml/SingleLogout	/logout/saml2/slo
/saml/metadata	/saml2/service-provider-metadata/gws

Backward compatibility

To ensure backward compatibility, configure a load balancer such as nginx to map legacy SAML endpoints to the new SAML 2.0 APIs when routing requests to GWS 8.6.

Example configuration:

```
location /saml/login {
  proxy_pass https://gws-host:8443/saml2/authenticate/gws;
  proxy_set_header Host $host;
  proxy_set_header X-Real-IP $remote_addr;
}
```

Next step

- Back to Configuring security

CSRF protection

Web Services provides protection against Cross Site Request Forgery (CSRF) attacks. For general information and background on CSRF, see the OWASP CSRF Prevention Cheat Sheet.

To set up Cross Site Request Forgery protection, set the following options in the serverSettings section of the **application.yaml** file on each of your Web Services nodes:

- enableCsrfProtection — determines whether CSRF protection is enabled on the Web Services node.
- crossOriginSettings — specifies the configuration for cross-origin resource sharing in Web Services. Make sure this option has the *exposedHeaders* setting with a value that includes `X-CSRF-HEADER,X-CSRF-TOKEN`.

For example, your configuration might look like this:

```
enableCsrfProtection: true
crossOriginSettings:
    corsFilterCacheTimeToLive: 120
    allowedOrigins: http://*.genesys.com, http://*.genesyslab.com
    allowedMethods: GET,POST,PUT,DELETE,OPTIONS
    allowedHeaders: "X-Requested-With,Content-Type,Accept,
Origin,Cookie,authorization,ssid,surl,ContactCenterId"
    allowCredentials: true
    exposedHeaders: "X-CSRF-HEADER,X-CSRF-TOKEN"
```

For more information about CSRF protection in the Web Services API, see Cross Site Request Forgery Protection.

Next step

- Back to Configuring security

CORS filter

Web Services supports Cross-Origin Resource Sharing (CORS) filter, which allows applications to request resources from another domain. For general information and background on CORS, see Cross-Origin Resource Sharing.

> **Important**
> CORS must be enabled for the screen recording options to be available in the Speechminer Web UI when the using Microsoft Internet Explorer web browser.

To set up Cross-Origin Resource Sharing, make sure you set the crossOriginSettings option in the serverSettings section of the **application.yaml** file on each of your Web Services nodes. It specifies the configuration for cross-origin resource sharing in Web Services. Make sure this option has the **exposedHeaders** setting with a value that includes `X-CSRF-HEADER,X-CSRF-TOKEN`.

For example, your configuration might look like this:

```
crossOriginSettings:
    corsFilterCacheTimeToLive: 120
    allowedOrigins: http://*.genesys.com, http://*.genesyslab.com
    allowedMethods: GET,POST,PUT,DELETE,OPTIONS
    allowedHeaders: "X-Requested-With,Content-
Type,Accept,Origin,Cookie,authorization,ssid,surl,ContactCenterId,X-CSRF-TOKEN"
    allowCredentials: true
    exposedHeaders: "X-CSRF-HEADER,X-CSRF-TOKEN"
```

For more information about CORS in the Web Services API, see Cross-Origin Resource Sharing.

Next step

- Back to Configuring security

Web Services authentication flow

Web Services provides authentication in the following sequence:

1. Salesforce Authentication

- Enters here if a request contains two specific headers (Salesforce Session ID and Salesforce Identity URL).
- If successful, the user is authenticated and execution flow proceeds to the authorization stage.
- If authentication headers are not present or authentication fails, execution flow proceeds to the next step.

2. Configuration Server Authentication

- Enters here if a request contains basic authentication header and Configuration Server authentication is enabled for this contact center.
- If successful, user is authenticated and execution flow proceeds to the authorization stage.
- If authentication headers are not present, Configuration Server authentication is disabled, or authentication fails, execution flow proceeds to the next step.

3. Web Services Authentication

- Enters here if a request contains basic authentication header.
- If successful, user is authenticated and execution flow proceeds to the authorization stage.
- If authentication headers are not present or authentication fails, execution flow proceeds to the next step.

4. Security Assertion Markup Language (SAML) Authentication

- Enters here if SAML is enabled and configured.
- An attempt is made to authenticate user through the standard SAML authentication flows.
- If successful, the user is authenticated and execution flow proceeds to the authorization stage.
- If not successful, the user receives an anonymous authentication, which means this users is only given access to unprotected endpoints.

Next step

- Back to Configuring security

Password encryption

For added security, consider encrypting your passwords in the **application.yaml** file.

The following table identifies which passwords can be encrypted and where you can find them in the **application.yaml** file:

File section	Settings
jetty > ssl	• keyStorePassword • keyManagerPassword • trustStorePassword
serverSettings	• opsUserPassword • cmePassword • jksPassword • webDAVPassword
serverSettings > samlSettings	• encryptionKeyPassword • signingKeyPassword • tlsKeyPassword
serverSettings > accountManagement > smtpServer	• password
serverSettings > elasticSearchSettings > transportClient	• password • keyStorePassword • keyPassword
db	• password
spring > data > redis	• password

Procedure: Encrypting passwords

Start

1. Run the GWS application with the **--encrypt** parameter followed by the password you need to encrypt. For example:

   ```
    $ java -jar gws.jar --encrypt ops
   CRYPT:an03xPrxLAu9p==
   ```

   The GWS application only encrypts and prints the password. The server won't actually start.

2. Copy the printed encrypted password and paste into the **application.yaml** file. For example:

   ```
   opsUserName: ops
    opsUserPassword: CRYPT:an03xPrxLAu9p==
   ```

   The server only decrypts passwords that start with the **CRYPT:** prefix. Passwords without the **CRYPT:** prefix are considered plain text and remain unmodified.

End

Secure Cookies

Web Services uses the **secure** flag option when sending a new cookie to the user within an HTTP Response. The purpose of the **secure** flag is to prevent cookies from being observed by unauthorized parties due to the transmission of a the cookie in clear text.

Enabling the **secure** flag

Set the **cookies** option in the **jetty** section of the **application.yaml** file on your Web Services nodes. For details, see Configuring Web Services.

```
cookies:
   httpOnly: true
   secure: true
```

Sample Cookie Header when **secure** flag is not set

```
Set-Cookie: MyCookieName=The value of my cookie; path=/; HttpOnly
```

Sample Cookie Header when **secure** flag is set

```
Set-Cookie: MyCookieName=The value of my cookie; path=/; HttpOnly; secure
```

When the cookie is declared as secure in the cookies configuration option, the browser will prevent the transmission of a cookie over an unencrypted channel.

HTTPS for Elasticsearch

The HTTPS communication secures the interaction between Elasticsearch and various clients such as web browsers, Postman, and Spring Boot client applications. This requires configuration in both Server and Client.

Server configuration

In Genesys Web Services (GWS) 8.6, you can use your custom Java KeyStore (JKS) file to establish the HTTPS connection between GWS and Elasticsearch.

To establish the HTTPS connection,

1. Copy the **jksStorage.jks** file to the Elasticsearch configuration path: **/usr/share/elasticsearch/config/jksStorage.jks**
2. Enable SSL by configuring the JKS file by using the following settings in the Elasticsearch configuration.

```
xpack.security.enabled: true
xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.keystore.type: JKS
xpack.security.http.ssl.keystore.path: jksStorage.jks
xpack.security.http.ssl.keystore.password: *******
```

For more details, refer to the Elasticsearch official documentation site.

> **Important**
>
> Once SSL is enabled on the server side, clients must use the same certificate(s) from the JKS file for encrypted connections.

Client (GWS 8.6) configuration

The client side configuration involves configuring the certificate in the client's trustStore. You can do this in two ways:

1. Custom trustStore configuration
2. System default configuration

Custom trustStore configuration

In this method, GWS 8.6 serves as the client.

To enable custom trustStore configuration in GWS 8.6,

1. Copy the JKS file to the client (GWS) machine.
2. Add the following configuration in the **application.yaml** file of the GWS 8.6 application.

```
serverSettings:
  caCertificate: /path/to/jksStorage.jks
  jksPassword: *******

  elasticSearchSettings:
     transportClient:
        nodes:
          - host: 127.0.0.1
            port: 9200
        username: elastic
        password: password
        useTls: true # Enable this for ES https connection
```

Peer verification configuration

You can verify the peer's certificate of other SSL/TLS connections to ElasticSearch for enhanced security by adding the `verifyPeer` option. Add this option in the **elasticSearchSettings** section in the **application.yaml** file of the GWS 8.6 application. By default, this option is set to `true`. For example,

```
elasticSearchSettings:
  verifyPeer: true/false
```

System default configuration

If a custom configuration is not found (that is, if **serverSettings.caCertificate** is not configured), then the system default configuration is used for the client side configuration.

The JDK ships with a limited number of trusted root certificates in the **<java-home>/lib/security/cacerts** file. It is your responsibility to maintain (that is, add/remove) the certificates contained in the default truststore. Depending on the certificate setup of the ElasticSearch server, additional root certificate(s) must be added. For more details, refer the JSSE Reference guide.

From the Elasticsearch server setup in your environment, extract the certificate details from the **.jks** file, and append it to the **<java-home>/lib/security/cacerts** file. The example JKS file name used in this article is **jksStorage.jks**.

To update the **cacerts** file,

1. Export the certificate from the **.jks** file by running the following command:
   ```
   keytool -exportcert -alias your_alias_name -file ca_cert.pem -keystore
    jksStorage.jks
   ```
   If prompted to enter a password, use the password associated with your JKS file.
2. Import the certificate into the Java cacerts file.
   ```
   keytool -importcert -alias your_alias_name -file ca_cert.pem -keystore cacerts
   ```

If prompted to enter a password, use the cacerts password, which is `changeit`.

Enabling anonymous access

When **xpack.security.enabled** is set to `true`, login credentials are required to access Elasticsearch. If you want to set up Elasticsearch without login credentials, you can enable anonymous access.

To enable anonymous access, add the following settings to the Elasticsearch configuration.

```
xpack.security.authc.anonymous.roles: superuser
xpack.security.authc.anonymous.authz_exception: true
```

For details on built-in roles, refer to the Elasticsearch Built-in Roles Documentation.

Elasticsearch connection configuration

Elasticsearch supports the following connections depending on the configured settings:

- HTTP without authentication
- HTTP with authentication
- HTTPS with authentication
- HTTPS without authentication

Settings	HTTP without Authentication	HTTP with Authentication	HTTPS with Authentication	HTTPS without Authentication
xpack.security.enabled	false	true	true	true
xpack.security.http.ssl.enabled	false	false	true	true
ELASTIC_PASSWORD	NA	password	password	password
xpack.security.http.ssl.keystore.type	NA	NA	JKS	JKS
xpack.security.http.ssl.keystore.path	NA	NA	jksStorage.jks	jksStorage.jks
xpack.security.http.ssl.keystore.password	NA	NA	genesys	genesys
xpack.security.authc.anonymous.authz_exception	NA	NA	NA	true
xpack.security.authc.anonymous.roles	NA	NA	NA	superuser

mTLS configuration

To configure a specific keystore/truststore path along with its key alias and password for using mutual TLS (mTLS), add the following option:

```
elasticSearchSettings:
  transportClient:
    keyStorePath: /path/to/jksStorage.jks
    keyStorePassword: password
    keyAlias: key_alias
    keyPassword: password
```

Next Step

- Back to Configuring security

HTTPS for Platform Service

The following configuration are required at the Platform Service for accepting a HTTPS connection.

Server-side configuration for GWS Platform Service

To enable HTTPS at the server-side,

1. Configure the following TLS-specific settings.

Name	Mandatory	Default Value	Description
GWS_SERVER_TLS_REQUIRED	No	`false`	Determines if TLS is required. Change to `true` if a certificate is also provided. **Note**: Once TLS is enabled, Platform Service accepts only HTTPS requests.
GWS_SERVER_TLS_PROVIDER	No	`rustls`	Specifies the TLS provider. Possible values: `rustls`, `native`.
GWS_SERVER_TLS_HANDSHAKE_TIMEOUT	Yes (if TLS is enabled)	`10000` (ms)	Specifies the timeout (in milliseconds) for TLS handshake. Default is `10000` ms (10 seconds).
GWS_SERVER_TLS_CERT_PATH	Yes (if TLS is enabled)	**-**	Specifies the path to the TLS certificate.
GWS_SERVER_TLS_KEY_PATH	Yes (if TLS is enabled)	**-**	Specifies the path to the TLS key.
GWS_AUTH_TOKEN_IDLE_TIME	No	**-**	Specifies the idle time (in seconds) before an auth token is considered inactive. If absent, all tokens remain active until expiration.
GWS_SERVER_TLS_CERT_PASSWORD	No	""	Specifies the password for the TLS certificate, if applicable.

2. After you enable TLS for the Platform Service, ensure to update the health check URL in the `get_version()` function within `/usr/bin/gws-service-platform` to use HTTPS. For example,
`RESPONSE=$(curl -XGET -skLS https://${GWS_HOST}:${GWS_PORT}${GWS_HEALTH_CHECK_URL} 2>&1)`

3. Update monitoring services, such as Grafana, to send requests to the Platform Service using the HTTPS endpoint after enabling TLS.

Client-side (GWS 8.6) configuration

To establish a HTTPS connection with Platform Service, configure the Platform Service URL in the **application.yaml** file as shown in the following settings:

```
serverSettings:
  platformSettings:
    platformServiceUrl: https://<ip>:<port>
```

Also, make sure that configured CA (Certification Authority) certificates are added to the JKS (Java KeyStore) file.

```
serverSettings:
  caCertificate: /opt/ca_cert.jks #location of the file
  jksPassword: pa$$word #password for the keystore
```

HTTPS for Redis

TLS configuration for Redis

The following settings are required to enable TLS for Redis.

Parameter name	Description
password	Specifies the password used for Redis server authentication.
username	Specifies the username used for Redis server authentication.
verifyPeer	Determines whether to verify the peer's certificate of SSL/TLS connections to ElasticSearch for enhanced security. By default, this option is set to `true`.
truststorePath	Specifies the path to the truststore file, which will contain the Redis server certificate.
truststorePassword	Specifies the password to the truststore file.

Example setting:

```
# Redis Settings
spring:
  data:
    redis:
      ssl:
        enabled: ${REDIS.TLS:false}
      password: ${REDIS.PASSWORD:}
      username: <password>
      tls:
        trustStorePath: ${REDIS.TRUSTSTORE.PATH:}
        trustStorePassword: ${REDIS.TRUSTSTORE.PASSWORD:}
        verifyPeer: ${REDIS.VERIFY.PEER:true}
```

Alternatively, you can set the following environment variables:

- REDIS_TLS
- REDIS_PASSWORD
- REDIS_TRUSTSTORE_PATH
- REDIS_TRUSTSTORE_PASSWORD
- REDIS_VERIFY_PEER

Mutual TLS configuration

The following configuration is required for enabling Mutual TLS (mTLS) connection between GWS

application and Redis server.

Parameter name	Description
keystorePath	Specifies the path to the keystore file that contains the host certificate, which is sent to the Redis Server.
keystorePassword	Specifies the password to the keystore file.
KeyAlias	Specifies the alias (or name) under which the key is stored in the keystore.

Example setting:

```
spring:
  data:
    redis:
      tls:
        keystorePath:/path/to/jksStorage.jks
        keystorePassword: password
        keyAlias: instance
```

Starting and testing

After you install and configure Web Services, you should start the nodes in the following order:

1. Start the statNode, if deployed.
2. Start the remaining nodes.

Starting Web Services nodes

Starting GWS Platform Service

Start the GWS Platform Service by running the following command: `systemctl start gws-service-platform`

Review the status of the service by running the following command: `systemctl status gws-service-platform`

Starting GWS API Service

Start the GWS API service, if deployed, by running the following command: `systemctl start gws-api-v2`

Review the status of the service by running the following command: `systemctl status gws-api-v2`

Testing Web Services

Complete the steps below to verify each Web Services node is up and running.

1. Type the following URL into a web browser:
   `http://`*`ws_host`*`:`*`ws_port`*`/api/v2/diagnostics/version`
   - *`ws_host`* — The host name or IP address for the Web Services node.
   - *`ws_port`* — The port for the Web Services node.

   For example, the URL might be `http://192.0.2.20:8080/api/v2/diagnostics/version`

   If the request is successful, the version is printed in the browser:

   ```
   {"statusCode":0,"build-info":{"name":"cloud-
   web","time":1706590922.271000000,"version":"8.6.000.00.000","group":"com.genesys.gws","artifact":"cloud-
   web"}}
   ```

Testing Workspace Web Edition

Complete the following steps for each Web Services node to confirm that Workspace Web Edition is working.

1. Launch Workspace Web Edition by navigating to `http://ws_host:ws_port/ui/ad/v1/index.html` in a web browser.
   - *ws_host* — The host name or IP address for the Web Services node.
   - *ws_port* — The port for the Web Services node.

   For example, the URL might be `http://192.0.2.20:8080/ui/ad/v1/index.html`
2. Enter the credentials for a WWE agent, see Setting Up Agents On The System. Note that the user must be of the agent type.
3. After clicking **Log In**, Workspace Web Edition displays the agent desktop view and the agent is ready to work.

Testing Gplus Adapter for Salesforce

Complete the following steps for each Web Services node to confirm that Gplus Adapter for Salesforce is working.

1. Launch Gplus Adapter for Salesforce by navigating to `http://ws_host:ws_port/ui/crm-adapter/index.html` in a web browser.
   - *ws_host* — The host name or IP address for the Web Services node.
   - *ws_port* — The port for the Web Services node.

   For example, the URL might be `http://192.0.2.20:8080/ui/crm-adapter/index.html`
2. Enter the credentials for any of the previously created agents. Note that the user must be of the agent type.
3. After clicking **Log In**, Gplus Adapter for Salesforce displays in the web browser. For details about how to deploy the adapter in Salesforce, see Gplus Adapter for Salesforce in this guide.

Web Services configuration options

You can set the configuration options below in the corresponding sections of the **application.yaml** file on your Web Services nodes. For details, see Configuring Web Services.

logging

Settings in this section are listed under "logging".

config

Default Value: `logback.xml`
Valid Values: A valid path
Mandatory: No
Specifies the path to the **logback.xml** file. You created this file (or Web Services created it for you) as part of Deploying the web application.

file

Default Value: `cloud.log`
Valid Values: A valid file name
Mandatory: No
Specifies the name of the log file. This value is stored in *${LOG_FILE}* which may be used in **logback.xml**.

path

Default Value: `./`
Valid Values: A valid path
Mandatory: No
Specifies the path to the log file. This value is stored in *${LOG_PATH}* which may be used in **logback.xml**.

jetty

Settings in this section are listed under "jetty".

cookies

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
httpOnly	No	true	If true, it sets an HTTP-only flag for session cookies.
secure	No	false	If true, it sets the secure cookie flag for session cookies.
sameSite	Yes	None	Specifies what should be returned as SameSite cookie attribute value in response for Jetty's session cookie. Valid values are `None`, `Lax`, or `Strict`.

Mandatory: No
Specifies how Jetty should handle cookies. For example:

```
cookies:
    httpOnly: true
    secure: true
    sameSite: None
```

These options only take effect if **enableSsl** is set to `true`.

host

Default Value: `0.0.0.0`
Valid Values: A host name or IP address
Mandatory: No
Specifies the host name or IP address of the Jetty host.

port

Default Value: `8090`
Valid Values: A valid port
Mandatory: No
Specifies the port of the Jetty host.

idleTimeout

Default Value: `30000`
Valid Values: An integer greater than 0
Mandatory: No
Specifies the maximum idle time, in milliseconds, for a connection.

soLingerTime

Default Value: `-1`
Valid Values: An integer greater than 0, or -1 to disable

Mandatory: No
Specifies the socket linger time.

sessionMaxInactiveInterval

Default Value: `1800`
Valid Values: An integer greater than 0
Mandatory: No
Specifies the period, in seconds, after which a session is deemed idle and saved to session memory.

enableWorkerName

Default Value: `true`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether to add the **WorkerName** parameter into the JSESSIONID cookie.

enableRequestLog

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables request logging. If you set the value to `true`, you must also set values for the requestLog option.

requestHeaderSize

Default Value: 8192
Valid Values: Any positive integer value greater than the default value

Specifies the allowed request header size for the Jetty servlet container.

responseHeaderSize

Default Value: 8192
Valid Values: Any positive integer value greater than the default value

Specifies the allowed response header size for the Jetty servlet container.

requestLog

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
filename	No	yyyy_mm_dd.cloud-request.log	Specifies the log file name format.
filenameDateFormat	No	yyyy_MM_dd	Specifies the log file name date format.

Name	Mandatory	Default Value	Description
logTimeZone	No	GMT	Specifies the timestamp time zone used in the log.
retainDays	No	90	Specifies the time interval, in days, for which Jetty should retain logs.
append	No	true	Specifies whether Jetty appends to the request log file or starts a new file.
extended	No	true	Specifies whether Jetty logs extended data.
logCookies	No	true	Specifies whether Jetty logs request cookies.
logLatency	No	true	Specifies whether Jetty logs the request latency.
preferProxiedForAddress	No	true	Specifies whether Jetty logs IP address or the IP address from the X-Forwarded-For request header.

Mandatory: No
Specifies how Jetty should handle request logging. For example:

```
enableRequestLog: true
  requestLog:
    filename: yyyy_mm_dd.cloud-request.log
    filenameDateFormat: yyyy_MM_dd
    logTimeZone: GMT
    retainDays: 90
    append: true
    extended: true
    logCookies: false
    logLatency: true
    preferProxiedForAddress: true
```

These options only take effect if enableRequestLog is set to `true`.

enableSsl

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables Secure Sockets Layer support. If you set the value to `true`, you must also set values for the ssl option.

ssl

Default Value: None

Valid Values:

Name	Mandatory	Default Value	Description
port	No	443	The SSL port. This option is the equivalent of the Jetty "https.port" variable.
securePort	No	8443	The port to which integral or confidential security constraints are redirected. This option is the equivalent of the Jetty "jetty.secure.port" variable.
idleTimeout	No	30000	The maximum idle time, in milliseconds, for a connection.
soLingerTime	No	-1	The socket linger time. A value of -1 disables this option.
keyStorePath	No	None	The keystore path.
keyStorePassword	No	None	The keystore password.
keyManagerPassword	No	None	The key manager password.
keyStoreProvider	No	None	The keystore provider.
keyStoreType	No	JKS	The key store type.
trustStorePath	No	None	The truststore path.
trustStorePassword	No	None	The truststore password.
trustStoreProvider	No	None	The truststore provider.
trustStoreType	No	JKS	The truststore type.
needClientAuth	No	None	Set this option to true if SSL needs client authentication.
wantClientAuth	No	None	Set this option to true if SSL wants client authentication.
certAlias	Yes	None	The alias of the SSL certificate for the connector.
validateCerts	No	None	Set this option to true if the SSL certificate has to be validated.
validatePeerCerts	No	None	Set this option to true if SSL certificates of the peer have to be validated.
trustAll	No	None	Set this option to true if

Name	Mandatory	Default Value	Description
			all certificates should be trusted if there is no keySstore or truststore.
renegotiationAllowed	No	None	Set this option to true if TLS renegotiation is allowed.
excludeCipherSuites	No	None	Specifies the array of cipher suite names to exclude from enabled cipher suites.
includeCipherSuites	No	None	Specifies the array of cipher suite names to include in enabled cipher suites.
endpointIdentificationAlgorithm	No	None	Specifies the endpoint identification algorithm. Set this option to "HTTPS" to enable hostname verification.
includeProtocols	No	None	The array of protocol names (protocol versions) to include for use on this engine.
excludeProtocols	No	None	The array of protocol names (protocol versions) to exclude from use on this engine.
enableHsts	No	false	If set to `true`, Genesys Web Services (GWS) provides support for HTTP Strict Transport Security (HSTS) protocol. HSTS prevents Man-in-the-Middle attacks that can occur in unsecure (HTTP) browser sessions. The following string is inserted in the response header: `Strict-Transport-Security: max-age=31536000 ; includeSubDomains` This string tells the browser to not accept any untrusted, expired, or revoked TLS certificates from the domain.
enableNonSecureToSecureRedirect	No	false	Redirects HTTP requests to HTTPS.

Name	Mandatory	Default Value	Description

			When enabled, the following string is sent in the response header:
			
			`HTTP/1.1 302 Found`
			`Location: https://...`

Mandatory: No
Specifies how Jetty should handle support for Secure Sockets Layer. For example:

```
enableSsl: true
 ssl:
   port: 443
   securePort: 8443
   idleTimeout: 30000
   soLingerTime: -1
```

These options only take effect if **enableSsl** is set to `true`.

cookies

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
httpOnly	No	true	If true, it sets an HTTP-only flag for session cookies.
secure	No	false	If true, it sets the secure cookie flag for session cookies.

Mandatory: No
Specifies how Jetty should handle cookies. For example:

```
cookies:
   httpOnly: true
   secure: true
```

These options only take effect if **enableSsl** is set to `true`.

sessionCookieName

Default Value: `JSESSIONID`
Valid Values: Any string which can be used as a cookie name as per RFC 6265
Mandatory: No
Defines the name of the session cookie used by Web Services.

enableXXSSProtection

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables XSS Protection and the following header is added to the response.

```
X-XSS-Protection: 1
```

enableXXSSProtectionBlockMode

Default Value: `true`
Valid Values: `true`, `false`
Mandatory: No
When enabled, blocks the page from being rendered by sending the following header in the response:

```
X-XSS-Protection: 1; mode=block
```

enableXContentTypeOptions

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
When enabled, content sniffing is disabled by sending the following header in the response:

```
X-Content-Type-Options: nosniff
```

xFrameOptions

The X-Frame-Options HTTP response header can be used to indicate whether a browser should be allowed to render a page in a <frame>, <iframe>, or <object>. Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.

Default Value: None
Valid Values: DENY, SAMEORIGIN, ALLOW-FROM
Mandatory: No
When enabled, the following header is added to the response.

```
X-Frame-Options: DENY
```

xFrameOptionsAllowFromUri

Default Value: None
Valid Values: A valid URI
Mandatory: No
When enabled, **xFrameOptions** is set to ALLOW-FROM, and only iframes from the specified URI are allowed. The following header is added to the response.

```
X-Frame-Options: ALLOW-FROM https://example.com/
```

This header is not supported by all browsers. For the list of browser types and versions that support the X-Frame-Options header, please refer to Clickjacking Defense Cheat Sheet.

xFrameOptionsExcludedUris

Default Value: None
Valid Values: List of URIs
Mandatory: No
Responses for requests from the specified URIs will not contain the X-Frame-Options header. The URIs must be relative and start with a slash (/) symbol. They must not contain the hostname and the port information. This is a workaround for browsers that do not support xFrameOptions.

> **Important**
>
> If you are using a browser that does not fully support xFrameOptions, it is highly recommended to list the URIs in the **xFrameOptionsExcludedUris** option to exclude these pages from the Clickjacking prevention:

```
jetty:
  ...
  xFrameOptionsExcludedUris:
    - /ui/crm-adapter/index.html
    - /ui/crm-workspace/index.html
    - /ui/dashboard/index.html
    - /ui/ad/v1/disaster-recovery.html
```

Postgres/MSSQL Cluster Settings

host

Default Value: None
Valid Values: A comma-separated list of IP addresses or host names
Mandatory: Yes
Specifies the Postgress or MS-SQL node IPs or host names.

port

Default Value: None
Valid Values: A valid port
Mandatory: Yes
Specifies the port for DB to listen for clients.

dbType

Default Value: postgres
Valid Values: postgres | mssql
Mandatory: Yes
Specifies the type of the DB.

dbName

Default Value: gws
Valid Values: A valid db name.
Mandatory: Yes
Specifies the name of the DB.

schema

Default Value: gws
Valid Values: A valid schema name.
Mandatory: Yes
Specifies the schema of the DB. Mandatory for Postgres only.

username

Default Value: None
Valid Values: Any alphanumeric value that can include special characters.
Mandatory: Yes
The username that the Web Services server uses to connect to DB.

password

Default Value: None
Valid Values: Any alphanumeric value, including special characters
Mandatory: Yes
The password that the Web Services server uses to connect to DB.

Redis Settings

redis

nodes

Default Value: null
Valid Values: List of Redis nodes.
Mandatory: Yes
Specifies the list of Redis nodes the Web Service server should connect to.

max-redirects

Default Value: 5
Valid Values: A positive integer.
Mandatory: No
Specifies the maximum number of redirections allowed when a client is interacting with the Redis

Cluster.

lettuce

Lettuce is a high-performance Redis client library for Java, offering asynchronous, non-blocking I/O operations and support for Redis cluster and Sentinel deployments. The following parameters can be configured for **Lettuce**.

period

Default Value: 60
Valid Values: A positive integer.
Mandatory: No
Specifies the period (in seconds) between the Cluster topology refresh job runs.

adaptive

Default Value: `false`
Valid Values: `true, false`
Mandatory: Yes
Specifies whether 'refresh triggers' initiate topology view updates based on events happened during Redis Cluster operations.

serverSettings

Settings in this section are listed under "serverSettings".

URLs

externalApiUrlV2

Default Value: None
Valid Values: A public schema-based URL ending with `/api/v2`.
Mandatory: Yes
Specifies the prefix used for resources in the public API. In a development environment, the host and port should be set to the host name or IP address of the Web Services node. In a production environment, the host and port should be set to the host name or IP address of the load balancer in a production environment. For example, `https://192.0.2.20/api/v2`.

internalApiUrlV2

Default Value: None
Valid Values: A public schema-based URL ending with `/internal-api`.
Mandatory: Yes
Specifies the prefix used for internal resources. In a development environment, the host and port should be set to the host name or IP address of the Web Services node. In a production environment, the host and port should be set to the host name or IP address of the load balancer in a production environment. For example, `http://192.0.2.20/internal-api`.

Paths

pathPrefix

Default Value:
Valid Values: A valid prefix
Mandatory: No
Specifies a prefix that Web Services adds to the relative URIs it includes in responses. For example, if you set **pathPrefix** to `/api/v2` and make the following request:

```
GET http://localhost:8080/api/v2/devices
```

Web Services returns the following response:

```
{
   "statusCode":0,
   "paths":[
      "/api/v2/devices/971ed91d-82bf-490b-94d2-02d240165764",
      "/api/v2/devices/a3f9e854-54d8-4260-bea3-d6e450ee7df0"
   ],
   "uris":[
      "http://localhost:8080/api/v2/devices/7c7ab1f7-e596-41bc-9ff4-4a12c489865f",
      "http://localhost:8080/api/v2/devices/a3f9e854-54d8-4260-bea3-d6e450ee7df0"
   ]
}
```

Notice that `paths` includes relative URIs with the `/api/v2` prefix.

General

chatServerRejoinAttempts

Default Value: `5`
Valid Values: `Any positive integer`
Mandatory: No
Specifies the number of join attempts to a chat session.

chatServerRejoinDelay

Default Value: `10000`
Valid Values: `Any positive integer`
Mandatory: No
Specifies the delay, in milliseconds, between join attempts to a chat session.

updateInteractionForSetContact

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether Web Services `RequestUpdateInteraction` instead of `RequestAssignInteractionToContact` when the `SetContact` request is called. Using `RequestUpdateInteraction` prevents UCS from updating the `ThreadId` interaction attribute. In some scenarios, updates to this attribute can cause an interaction to become orphaned in the UCS

database.

sendParticipantsUpdatedBeforeStatusChange

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Specifies the order in which GWS sends `ParticipantsUpdated` and `StatusChange` notifications. To make sure that after reconnecting to Chat Server GWS send the `ParticipantsUpdated` notification first and then the `StatusChange` notification, set the value of this option to `true`.

disableCreatorAppIdUpdates

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Preserves the original `CreatorAppId` of an interaction in UCS. Set the value of this option to `true` to make sure that the `CreatorAppId` is not changed after reconnecting to Chat Server.

enableIntermediateParticipantNicknameFix

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Specifies how the nickname of a participant in the chat message of the application CometD notification is displayed. By default (the `false` value of the option), the latest nickname of a participant is displayed for all messages in the chat transcript. When this option is set to `true`, messages of the participant who changed the nickname are displayed with the actual nickname at the time when messages were sent.

enableNotificationOnPullChat

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables sending of a notification with interaction properties while pulling chat interaction from a workbin. By default, the same notification is sent to other media types.

Add the following notification to the application.yaml file:

```
...
serverSettings:
  ...
  enableNotificationOnPullChat: true
```

enableJoinOnPullChat

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables joining to the chat session of the chat interaction that is being pulled from a workbin. Following notification is a chat transcript similar to the one which is sent for the Accept operation.

Add the following notification to the application.yaml file:

```
...
serverSettings:
  ...
  enableJoinOnPullChat: true
```

enableSaveEmailReplyUserDataToUCS

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables GWS to save the UserData of the parent email interaction to the child email interaction for Reply and ReplyAll operations. When set to `true`, GWS sends the AttachedData filled with the UserData of the parent interaction to UCS and UCS can render the corresponding field codes in Standard Responses.

enableUcsOrphanedScrollCleanup

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
When enabled, Web Services cleans up the orphan "Scrolls" from Universal Contact Server (UCS) that consume memory. When Web Services gets a list of interactions from UCS that has more than 1000 results, it creates a "Scroll" that must be released by Web Services when it is no longer needed.

enableFindOrCreateCallSearchInMemory

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
When enabled, Web Services searches for a call in memory, if the call is not found in Redis.

enableSyncConnectionToIxnServer

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
When enabled, Web Services waits for a configured amount of time for the Interaction Server connection to open before trying to send requests. You can use the **syncConnectionToIxnServerTimeout** option to adjust the wait time.

syncConnectionToIxnServerTimeout

Default Value: `3000`
Valid Values: positive integer
Mandatory: No
Defines the wait time for opening the connection to Interaction Server when the **enableSyncConnectionToIxnServer** option is enabled.

enableNotReadyOnConferenceInviteExpiration

Default Value: `false`

Valid Values: `true`, `false`
Mandatory: No
When enabled, Web Services changes an agent's state when the agent does not answer an invite to Consult or Conference by a queue (routing-based). You can configure the status, which will be set using the **statusNameOnConferenceInviteExpiration** option.

statusNameOnConferenceInviteExpiration

Default Value: `NotReady`
Valid Values: name of the agent's status
Mandatory: No
Defines the status which will be set when the agent does not answer an invite to Consult or Conference by a queue (routing-based), and the **enableNotReadyOnConferenceInviteExpiration** is enabled.

enableEmailCaseInsensitive

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables enforcing case-insensitive comparison of email addresses to remove the sender's address from recipients lists ("To", "Cc", "Bcc") while replying all in an email interaction.

enableInteractionRequestPull

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
When enabled, Web Services gets the current interaction from Interaction Server if an agent is disconnected from one Web Services node and is reconnected to another. This feature is only applicable if the agent reconnects using the same Workspace Web Edition instance. It does not apply to any active consultations or supervisors associated with the interaction. Previously, it was possible to have more than one agent in the same chat session when an agent switched connections to a new node while handling an interaction.

temporaryAuthenticationTokenTTL

Default Value: 300
Valid Values: An integer greater than 0
Mandatory: No
Specifies the time to live, in seconds, for the temporary authentication token.

enableCsrfProtection

Default Value:
Valid Values: `true`, `false`
Mandatory: No
Enables cross site request forgery protection. If you set the value to `true`, make sure you use the default values for **exposedHeaders** in the crossOriginSettings option. If you have already updated the **exposedHeaders**, just make sure the values include the defaults.

enableOpenIDConnect

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether Web Services uses OAuth 2.0 authentication.

enableStaleSessionsMonitoring

Default Value: `true`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether Web Services should run its StaleSessionsMonitor process to periodically poll Redis for expired CometD sessions. This process releases any devices that might not have been released as part of EndContactCenterSession if CometD session information was lost.

requestTakeSnapshotTimeout

Default Value: `10000`
Valid Values: positive integer
Mandatory: No
Defines the timeout for the **GetContent** operation. This API request leads to sending **RequestTakeSnapshot** to Interaction Server. In some cases, when there are lots of interactions in a queue, this request could take a long time.

staleSessionsMonitorSettings

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
monitoringInterval	No	60	Specifies, in seconds, how often Web Services scans for expired CometD sessions.
expiredSessionAge	No	180	Specifies the age, in seconds, at which Web Services considers the CometD session to be expired.

Mandatory: No
Specifies the configuration for monitoring expired CometD sessions. For example:

```
...
staleSessionsMonitorSettings:
    monitoringInterval: 60
    expiredSessionAge: 180
```

includeMessageType

Default Value: `false`

Valid Values: `true, false`
Mandatory: No
Specifies whether to include the original message type of a chat message in the MessageLogUpdated CometD notification when you make a SendMessage request with the Chat API.

enableInteractionPropertiesForStandardResponse

Default Value: `false`
Valid Values: `true, false`
Mandatory: No
Specifies whether Web Services uses the interaction properties from Interaction Server to render standard responses instead of using the interaction attributes from Universal Contact Server.

enableSpecificTwoStepTransferForAvayaSwitch

Default Value: false
Valid Values: true, false
Mandatory: No
When set to `true`, Web Services enables agents to complete a call transfer to a consultation target while the consultation call is on hold.

> **Important**
> This option is used for Avaya switch environments only.

enableStatusForOfflineChatOnRecovery

Default Value: false
Valid Values: true, false
Mandatory: No
When set to `true`, if an offline chat interaction is restored for an agent on a different node, the interaction has the 'LeftChat' status along with the list of corresponding capabilities.
To ensure that interactions are recovered as well as having the status set, Genesys recommends that when using this feature, you also set the value of the **enableSyncConnectionToIxnServer** to `enabled`.

enablePutOnHoldInWorkbin

Default Value: false
Valid Values: true, false
Mandatory: No
Specifies whether or not chat interactions can be put on hold in a workbin and then reopened later to continue the chat.

enableChatSynchronization

Default Value: false
Valid Values: true, false
Mandatory: No

Enable this option so that Web Services and Applications will display all chat messages for the current chat session. Previously, few messages could be lost if they were sent closely after an agent joined the chat session.

Timeouts

activationFailFastPeriod

Default Value: `10000`
Valid Values: Any integer greater than 0
Mandatory: Yes
Determines fast-fail period length after a failure. Any attempt to reconnect to a Genesys server within the time specified by the **activationFailFastPeriod** option results in an immediate failure. To understand how this option works, consider this scenario:

- Agent A logs in to a GWS node. Note that connection to tserver is not active yet. GWS attempts to connect to tserver and the connection attempt fails.
- Agent B attempts to login within the **activationFailFastPeriod**. GWS does not attempt to connect to tserver. The agent is authenticated, but is not logged into the voice channel.
- After **activationFailFastPeriod** expires, Agent C attempts to login. GWS connects to TServer if the authentication is successful. Note that if the connection attempt fails, the activationFailFastPeriod is restarted again.

activationTimeout

Default Value: `12000`
Valid Values: An integer greater than 0
Mandatory: No
Specifies the timeout, in milliseconds, for connecting to any Genesys server (except Configuration Server). This may include several individual attempts if the initial attempt to connect is unsuccessful.

> **Important**
> The activation timeout for Configuration Server is specified with the `configServerActivationTimeout` option.

chatServerConnectionTimeout

Default Value: `7000`
Valid Values: Any positive integer
Mandatory: No
Specifies the timeout, in milliseconds, after which the attempt to connect to the Chat Server fails.

chatServerReconnectTimeout

Default Value: 10000
Valid Values: Any positive integer
Mandatory: No

Specifies the delay, in milliseconds, between attempts to connect to the Chat Server.

configServerActivationTimeout

Default Value: 35000
Valid Values: An integer greater than 0
Mandatory: No
Specifies the timeout, in milliseconds, for connecting to Configuration Server. This may include several individual attempts if the initial attempt to connect is unsuccessful.

configServerConnectionTimeout

Default Value: 15000
Valid Values: An integer greater than 0
Mandatory: No
Specifies the timeout, in milliseconds, for an individual connection attempt to Configuration Server.

connectionTimeout

Default Value: 4000
Valid Values: An integer greater than 0
Mandatory: No
Specifies the timeout, in milliseconds, for an individual connection attempt to any Genesys server (except Configuration Server).

> **Important**
> The connection timeout for Configuration Server is specified with the `configServerConnectionTimeout` option.

inactiveUserTimeout

Default Value: 60
Valid Values: An integer greater than 0
Mandatory: No
Specifies the interval, in seconds, at which the inactive user cleanup process is run by the server. This process is run to invalidate HTTP sessions for users who have been deleted or whose user roles have changed.

reconnectAttempts

Default Value: 1
Valid Values: An integer greater than 0
Mandatory: Yes
Specifies the number of attempts Web Services makes to connect to any Genesys server before attempting to connect to the backup.

reconnectTimeout

Default Value: 10000
Valid Values: An integer greater than 0
Mandatory: Yes
Specifies the timeout, in milliseconds, between the reconnect attempts.

agentSessionCleanUpTimeout

Default Value: 60000 (ms)
Valid Values: Any positive integer.
Mandatory: No
Specify the timeout in milliseconds before the agent session cleanup procedure is initiated.

platformConfigurationReadTimeout

Default value: 10000
Valid values: Any positive integer.
Specify the timeout (in milliseconds) for platform configuration read requests. If an invalid or a negative value is provided, a warning message will be logged and the default value would be applied.

OPS account

opsUserName

Default Value: None
Valid Values: Any alphanumeric value that can include special characters
Mandatory: Yes
Specifies the name of the Web Services super user. Web Services creates this user at startup.

opsUserPassword

Default Value: None
Valid Values: Any alphanumeric value, including special characters
Mandatory: Yes
Specifies the password for the Web Services super user. Web Services creates this user at startup.

Configuration Server credentials

applicationName

Default Value: None
Valid Values: A valid application name
Mandatory: Yes
The name of the Web Services node application object in Configuration Server. For example, `WS_Node`.

applicationType

Default Value: None
Valid Values: A valid application type

Mandatory: Yes
The type of the Web Services node application object in Configuration Server. This value should be `CFGGenericClient`.

cmeUserName

Default Value: None
Valid Values: A valid Configuration Server user
Mandatory: Yes
The username that the Web Services server uses to connect to Configuration Server.

> **Important**
>
> Genesys recommends that you use the provided "default" account in Configuration Server. It is possible to use a different account, but you must take care in configuring the user's account permissions. Outside of a lab setting, this is best done in consultation with Genesys.

cmePassword

Default Value: None
Valid Values: A valid password
Mandatory: Yes
The password for the Configuration Server user Web Services uses to connect to Configuration Server.

addParticipantToConferenceNotReplace

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No Adds a new participant to the conference instead of replacing in some complex cross-site scenarios.

Statistics

locationAwareMonitoringDistribution

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No

Enables you to configure additional connections to different StatServers. GWS chooses the one that is "visible" from the GWS node (based on the location specified in the connection).

Set this option to `true` on all nodes only when deploying multiple data centers.

> **Important**
>
> - This option applies only to multi-data center environments.
> - The statistics collected in a multi-data center environment will not be displayed properly without both this option and the **enableMultipleDataCenterMonitoring** option set to `true`.

enableMultipleDataCenterMonitoring

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No

Enables statistics collection for each data center in a multiple data center configuration.

Set this option to `true` on all nodes only when deploying multiple data centers.

> **Important**
>
> - This option applies only to multi-data center environments.
> - The statistics collected in a multi-data center environment will not be displayed properly without both this option and the **locationAwareMonitoringDistribution** option set to `true`.

statConnectionTimeout

Default Value: `5000`
Valid Values: A positive integer greater than 0
Mandatory: No
Specifies the connection timeout, in milliseconds, for connecting to Stat Server.

statReconnectAttempts

Default Value: 1
Valid Values: A positive integer
Mandatory: No
Specifies the number of reconnect attempts before switching to the backup Stat Server, if the connection to the primary Stat Server is lost.

statReconnectTimeout

Default Value: `10000`
Valid Values: An integer greater than 0

Mandatory: No
Specifies the timeout, in milliseconds, before reconnecting to Stat Server.

statOpenTimeout

Default Value: `60000`
Valid Values: An integer greater than 0
Mandatory: No
Specifies the timeout, in milliseconds, between when a request is sent to Stat Server to open a statistic and when Web Services server determines the statistic has not been opened. If the timeout expires, the Web Services server discards the request and sends a new one.

statisticsMonitorMultimediaChannelStates

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether to monitor user states on non-voice channels.

reportingSyncInterval

Default Value: 30
Valid Values: An integer greater than 0
Mandatory: No
Specifies the interval, in seconds, for the reporting services to poll the database for information about the activities of other monitoring nodes and for the current state of the contact center configuration. If you set this option to a larger value, it decreases the load on the database, but also increases the timeout for detecting nodes that are down and objects that are added or updated in contact centers.

> **Important**
> The value of this option specifies the rate of scheduling, not the delay between when the previous polling finishes and the new polling starts.

enableElasticSearchIndexing

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether the configuration information and statistics should be indexed to Elasticsearch. If set to `true`, you must set the crClusterName option.

statisticsOpenRetryInterval

Default Value: `60` (1 hour)
Valid Values: An integer greater than 0
Mandatory: No
Specifies the interval, in minutes, for trying to reopen failed statistics. For example, if a statistic cannot be opened on Stat Server, it is marked as failed and then the server attempts to reopen the

stat once every hour (the default value of **statisticsOpenRetryInterval**).

Multi regional supporting

nodePath

Default Value: None
Valid Values: A location and node ID, separated by a "/" — for example, `/US/node1`
Mandatory: Yes
Specifies the location and ID of the Web Services node within the deployment topology. This value must be unique across the deployment. For example, a value of `/US/node1` means that the node is located in the US region and has an ID of "node1". The node ID can be the hostname, the IP address, or any other unique identifier.

nodeId

Default Value: None
Valid Values: Any unique identifier, such as the node host name or IP
Mandatory: No
Specifies the unique identifier for the Web Services node. Each node in a cluster must have a unique `nodeId`.

SSL and CA

caCertificate

Default Value: None
Valid Values: Path to a signed certificate
Mandatory: No
Specifies the path to a certificate signed by a Certificate Authority. The file must be in the `.pem` or `.jks` format (if `.jks`, you can also set jksPassword). The certificate can be used if the WS_Cluster application uses Transport Layer Security (TLS) to connect to Genesys servers. This option is also mandatory to enable SAML authentication.

jksPassword

Default Value: None
Valid Values: Password for the key storage
Mandatory: No
Specifies the password for the key storage set in caCertificate, when the certificate is in `.jks` format. This option is mandatory to enable SAML authentication.

SAML

samlSettings

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
serviceProviderEntityId	No	If omitted, Web Services uses the value of the externalApiUrlV2 option.	Specifies the service provider entity ID to be used in the metadata.
encryptionKeyName	Yes	None	Specifies the Security Assertion Markup Language (SAML) encryption key name. This key has to be present in the JKS key storage specified in the caCertificate option.
encryptionKeyPassword	No	If omitted, Web Services uses the value of the jksPassword option.	Specifies the password used to extract the SAML encryption key from JKS storage.
signMetadata	No	true	Specifies whether generated metadata is signed with an XML signature that uses the certificate with the alias of **signingKeyName**.
signingKeyName	Yes	None	Specifies the SAML signing key name. This key has to be present in the JKS key storage specified in the caCertificate option.
signingKeyPassword	No	If omitted, Web Services uses the value of the jksPassword option.	Specifies the password used to extract the SAML signing key from JKS storage.
tlsKeyName	No	None	Specifies the TLS key name. This key has to be present in the JKS key storage specified in the caCertificate option.
tlsKeyPassword	No	If omitted, Web Services uses the value of the jksPassword option.	Specifies the password used to extract the SAML TLS key from JKS storage.
responseSkewTime	No	60	Specifies the maximum difference, in seconds, between the local time and time of the assertion creation which still allows message to be processed. Determines the maximum difference between clocks of the IDP and SP servers.

Name	Mandatory	Default Value	Description
defaultBinding	No	SSO_ARTIFACT	Specifies the default SAML binding Web Services uses. The valid values are: SSO_POST, SSO_PAOS, SSO_ARTIFACT, HOKSSO_POST, or HOKSSO_ARTIFACT.
requestSigned	No	true	Specifies whether this service signs authentication requests.
wantAssertionSigned	No	true	Specifies whether this service requires signed assertions.
identityProviderMetadata	Yes	None	Specifies the path or URL for the identity provider XML metadata file. You can set this option to the path to a physical location or, if the metadata file is exposed by the remote server over HTTP, you can specify the URL (in this case, Web Services applies a 5-second default request timeout).
useExternalUserId	No	false	Specifies whether Web Services should use the external user ID, a property of the CfgPerson object in the Configuration Database , for SAML authentication. If false, Web Services uses the username. If true, Web Service uses the external user ID.

Mandatory: No
Specifies the configuration for Security Assertion Markup Language (SAML) authentication for Web Services. For example:

```
...
samlSettings:
    serviceProviderEntityId: 10.10.15.60
    encryptionKeyName: client
    signingKeyName: client
    identityProviderMetadata: http://ipd.company.host/saml/metadata/idp-metadata.xml
    responseSkewTime: 120
    defaultBinding: SSO_POST
```

CORS

crossOriginSettings

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
allowedOrigins	No	None	Specifies a comma-separated list of allowed origins supported by this Web Services node. For example, http://*.genesys.com, http://*.genesyslab.com
allowedMethods	No	GET,POST,PUT,DELETE,OPTIONS	Specifies a comma-separated list of HTTP methods supported by the server.
allowedHeaders	No	X-Requested-With,Content-Type,Accept,Origin,Cookie,authorization,ssid,surl,ContactCenterId	Specifies whether to include the Access-Control-Allow-Headers header as part of the response to a pre-flight request. This specifies which header field names can be used during the actual request.
allowCredentials	No	true	Specifies the value of the Access-Control-Allow-Credentials header. This should typically be left at the default value.
corsFilterCacheTimeToLive	No	120	Specifies the delay after the contact center allowDomain updating takes effect.
exposedHeaders	No	X-CSRF-HEADER,X-CSRF-TOKEN	Specifies which custom headers are allowed in cross-origin HTTP responses. This should typically be left at the default value. If you do modify the value and you enable the enableCsrfProtection option, make sure the value for **exposedHeaders** includes `X-CSRF-HEADER,X-CSRF-TOKEN`.

Mandatory: No
Specifies the configuration for cross-origin resource sharing in Web Services. For example:

```
...
crossOriginSettings:
    corsFilterCacheTimeToLive: 120
    allowedOrigins: http://*.genesys.com, http://*.genesyslab.com
    allowedMethods: GET,POST,PUT,DELETE,OPTIONS
    allowedHeaders: "X-Requested-With,Content-Type,Accept,
Origin,Cookie,authorization,ssid,surl,ContactCenterId"
    allowCredentials: true
    exposedHeaders: "X-CSRF-HEADER,X-CSRF-TOKEN"
```

Elasticsearch

elasticSearchSettings

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
enableScheduledIndexVerification	No	false	Enables scheduled index verification on the Web Services node. This means that the node goes through all objects handled by Elasticsearch and makes sure they still exist in Configuration Server and vice versa.
indexVerificationInterval	No	15 minutes	Specifies an interval, in minutes, between index verifications for this Web Services node. Important In GWS 8.5, the default interval was 720 minutes.
enableIndexVerificationAtStartUp	No	true	Enables index verification at start-up on this node. This means that, at start-up, the node goes through all objects handled by Elasticsearch and makes sure they still exist in Configuration Server and vice versa.
retriesOnConflict	No	3	Controls how many times to retry if there is a version conflict when updating a document.

Name	Mandatory	Default Value	Description
transportClient	Yes	Values specified in TransportClientSettings	TransportClientSettings in the next table.

TransportClientSettings

Name	Mandatory	Default Value	Description
nodes	Yes	null	Specifies the list of Elasticsearch nodes the transport client should connect to.
useSniff	no	false	Specifies if the transport client should us sniffing functionality and perform auto-discovery of Elasticsearch nodes in the cluster.
ignoreClusterName	no	false	Specifies if Web Services should ignore the name of the cluster when connecting to the cluster.
pingTimeout	no	5000	Specifies, in milliseconds, the ping timeout for Elasticsearch nodes.
nodesSamplerInterval	no	5000	Specifies, in milliseconds, how often Web Services should sample/ping the Elasticsearch nodes listed and connected.

Mandatory: No
Specifies the configuration for Elasticsearch in Web Services. For example:

```
...
elasticSearchSettings:
    clientNode: true
    enableScheduledIndexVerification: true
    indexVerificationInterval: 60
    retriesOnConflict: 2
    useTransportClient: true
    transportClient:
        nodes:
             - {host: 127.0.0.1, port: 9300}
        useSniff: true
        ignoreClusterName: true
        pingTimeout: 10000
        nodesSamplerInterval: 10000
```

newIndexSettings

Name	Mandatory	Default Value	Description
numberZZZofZZZshards	No	5	Specifies the number of primary shards that an index should have. Sharding is the process of splitting an index's data into smaller parts.
refreshZZZinterval	No	1	Specifies how often to perform a refresh operation, which makes recent changes to the index visible to search.
numberZZZofZZZreplicas	no	1	Specifies the number of replica shards for each primary shard in an index.
maxZZZresultZZZwindow	no	10000	Specifies the maximum number of results that can be returned for a single query from an Elasticsearch index.

Mandatory: No
Specifies the configuration for new index settings in Web Services for Elasticsearch. **Note:** The default value for these settings are automatically derived from the GWS 8.6 deployment when left empty. However, you can also override the default values as given in the following example setting.

```
newIndexSettings:
  index:
    numberZZZofZZZshards: 5
    refreshZZZinterval: 1s
    numberZZZofZZZreplicas: 3
    maxZZZresultZZZwindow: 20000
```

enableSecurityFeatures

Specifies the configuration for security features such as SSL. For HTTP connection with Elasticsearch, use the following settings:

```
xpack.security.enabled: false
xpack.security.enrollment.enabled: false
```

To enable HTTPS connection, refer the HTTPS for Elasticsearch article.

Caching

cachingSettings

Default Value: None

Valid Values:

Name	Mandatory	Default Value	Description
agentGroupsTtl	No	300	The time to live (TTL), in seconds, for contact-center Agent Group information in cache.
agentLoginsTtl	No	300	The time to live (TTL), in seconds, for contact-center Agent logins in cache.
agentStatesTTL	No	300	The time to live (TTL), in seconds, for contact-center Agent states in cache.
applicationsTTL	No	300	The time to live (TTL), in seconds, for contact-center applications in cache.
businessUnitSettingsTTL	No	300	The time to live (TTL), in seconds, for contact-center Business Unit settings in cache.
businessUnitsWithSubresourcesTTL	No	300	The time to live (TTL), in seconds, when cache is re-read and updated from Configuration Server.
cleanupPeriod	No	1800	The interval, in seconds, between caches cleanup (eviction of expired elements).
contactCenterFeaturesTTL	No	300	The time to live (TTL), in seconds, for contact-center feature IDs in cache.
contactCenterSettingsTTL	No	300	The time to live (TTL), in seconds, for contact-center custom settings in cache.
contactCentersTtl	No	300	The time to live (TTL), in seconds, for contact-center details in cache.
defaultSystemWideRefreshTimeout	No	300	Specifies the default timeout in seconds for system-wide refresh.
deviceTtl	No	300	The time to live (TTL), in seconds, for contact-center devices in cache.
enableBlockingStrategyForBusinessUnitsWithSubresourcesCache	No	false	Specifies whether business units with

Name	Mandatory	Default Value	Description
			subresources cache should utilize blocking approach or not.
enableBlockingStrategyForContactCenterSettingsCache	No	false	Specifies whether ContactCenter settings cache should utilize blocking approach or not.
enableBusinessUnitsWithSubresourcesCaching	No	false	Specifies whether business units with subresources should be served via cache or via direct Configuration Server reads.
enableSystemWideCaching	No	true	Specifies if caching is used system-wide (`true`) or only during statistics evaluation (`false`).
enumeratorsTtl	No	300	The time to live (TTL), in seconds, for enumerators details in cache.
enumeratorValuesTtl	No	300	The time to live (TTL), in seconds, for enumerators values in cache.
groupsTtl	No	300	The time to live (TTL), in seconds, for contact-center groups information in cache.
placeTtl	No	300	The time to live (TTL), in seconds, for places information in cache.
rpMonitoringTTL	No	60	The time to live (TTL), in seconds, for Routing Point monitoring information in cache.
scriptTtl	No	300	The time to live (TTL), in seconds, for Script details in cache.
socialMediaChannelTtl	No	300	The time to live (TTL), in seconds, for Social Media channel information in cache.
socialMediaSettingTTL	No	1800	The time to live (TTL), in seconds, for Social Media Settings in cache.
skillsTTL	No	300	The time to live (TTL), in seconds, for contact-

Name	Mandatory	Default Value	Description
			center skills in cache.
systemWideCachingSize	No	1000	Specifies the maximum size of cache memory available across the system/application.
transactionsTTL	No	300	The time to live (TTL), in seconds, for contact-center transactions in cache.
usersTtl	No	300	The time to live (TTL), in seconds, for contact-center users information in cache.
virtualAgentGroupsTTL	No	300	The time to live (TTL), in seconds, for contact-center virtual agent groups in cache.
virtualAgentGroupSettingsTTL	No	300	The time to live (TTL), in seconds, for contact-center virtual agent groups settings in cache.
voiceContextCaching	No	true	Specifies whether to use in-memory cached context for processing voice events. Using caching reduces the processing time, but you can expect delays in configuration information propagation.
voiceContextReadSliceSize	No	500	Specifies the size of portions or slices used for reading voice context data within a system/application.
voiceContextRefreshInterval	No	60	The interval, in seconds, between refreshes of the context caches for voice event processing. The service reads configuration information from the database and then refreshes the corresponding caches.

Mandatory: No
Specifies how Web Services should handle various caching scenarios. For example:

. . .

```
cachingSettings:
   enableSystemWideCaching: true
   agentStatesTTL: 30
   contactCenterFeaturesTTL: 30
   contactCenterSettingsTTL: 30
   voiceContextCaching: true
   voiceContextRefreshInterval: 60
```

StandardResponse Caching

Overview

Default cache setting is one hour. However, the tree is not cached, only request to UCS is cached.

```
Default timeToLiveSeconds = 3600; maxEntriesLocalHeap = 10000.
```

Example

```
Request to GetRootCategories – it will be cached for 1 hour.
Request to GetCategory(x) – it will be cached for 1 hour.
You make any changes to Category x – you will not receive it for 1 hour.
You make a change to Category y.
You make a request to GetCategory(y) – you will receive the latest from the server and it
will be cached for 1 hour.
```

Cache Managing

You must provide the following settings in the **application.yaml** file to change default parameters for caching:

```
serverSettings:
 cachingSettings:
   dedicatedCacheSettings:
      - cacheName: ContactServerCategoriesCache
        timeToLiveSeconds: <TTL>
        maxEntriesLocalHeap: <Heap size>
      - cacheName: ContactServerStandardResponsesCache
        timeToLiveSeconds: <TTL>
        maxEntriesLocalHeap: <Heap size
```

DoS Filter

enableDosFilter

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Enables the denial of service filter. If you set the value to `true`, you must also set values for the dosFilterSettings option.

dosFilterSettings

Default Value: None

Valid Values:

Name	Mandatory	Default Value	Description
maxRequestsPerSec	No	25	Specifies the maximum number of requests from a connection per second. Requests that exceed this are first delayed, then throttled.
delayMs	No	100	Specifies the delay, in milliseconds, imposed on all requests over the rate limit, before they are considered at all. Valid values: • -1 = reject request • 0 = no delay • Any other number = delay in milliseconds
maxWaitMs	No	50	Specifies the length of time, in milliseconds, to blocking wait for the throttle semaphore.
throttledRequests	No	5	Specifies the number of requests over the rate limit that are able to be considered at once.
throttleMs	No	30000	Specifies the length of time, in milliseconds, to asynchronously wait for semaphore.
maxRequestMs	No	30000	Specifies the length of time, in milliseconds, to allow the request to run.
maxIdleTrackerMs	No	30000	Specifies the length of time, in milliseconds, to keep track of request rates for a connection, before deciding that the user has gone away, and discarding the connection.
insertHeaders	No	true	If true, DoSFilter headers are inserted into the response.
trackSessions	No	true	If true, the usage rate is tracked by session if a session exists.
remotePort	No	false	If true and session

Name	Mandatory	Default Value	Description
			tracking is not used, then the rate is tracked by IP + port (effectively connection).
ipWhitelist	No	""	A comma-separated list of IP addresses that is not rate limited.

Mandatory: No
Specifies how Web Services should handle denial of service. For example:

```
...
enableDosFilter: true
dosFilterSettings:
   maxRequestsPerSec: 30
   ipWhitelist: 192.168.0.1,192.168.0.2
```

These options only take effect if enableDosFilter is set to `true`.

Account management

accountManagement

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description

forgotPasswordEmailTemplate	No	None	The template to use for an email that is sent to a user who forgets his or her password.
			
			`forgotPasswordEmailTemplate:`
			`  from: <from address>`
			`  subject: <Subject line>`
			`  body: <Email body>`

accountCreatedEmailTemplate	No	None	The template to use for a email that is sent to a user who creates a new account.
			
			`accountCreatedEmailTemplate:`
			`  from: <from address>`
			`  subject: <Subject`

Name	Mandatory	Default Value	Description
			line> body: <Email body>
smtpServer	No	None	The SMTP server configuration information. smtpServer: host: <smtp host name> port: <smtp port> userName: <user name for the account> password: <password in plain text> timeout: <optional SMTP timeout>

Mandatory: No
Specifies the configuration for the email notification and email server used when creating a new user. This option accepts the email server's login configuration, as well as email templates for resetting and creating user passwords. For example:

```
accountManagement:
     forgotPasswordEmailTemplate:
             from: <from address>
             subject: <Subject line>
             body: <Email body>
     accountCreatedEmailTemplate:
             from: <from address>
             subject: <Subject line>
             body: <Email body>
     smtpServer:
             host: <smtp host name>
             port: <smtp port>
             userName: <user name for the account>
             password: <password in plain text>
             timeout: <optional SMTP timeout>
```

CometD

cometDSettings

Default Value: None
Valid Values:

Name	Mandatory	Default Value	Description
cometdSessionExpirationTimeout	No	60	Specifies the timeout for the CometD session to expire on disconnect. It might take an additional minute for the session to be closed after it expires. If you set this option to -1, the session never expires. An agent can login again before the end of this timeout to disable session expiration.
closeHttpSessionOnCometDExpiration	No	true	Enables or disables HTTP session invalidation when CometD times out.
cookieHttpOnly	No	true	If true, it sets an HTTP-only flag for CometD's session cookies.
cookieSecure	No	false	If true, it sets the secure cookie flag for CometD's session cookies.
cookieSameSite	Yes	None	Specifies what should be returned as SameSite cookie attribute value in response for CometD's session cookie. Valid values are: None, Lax, or Strict.
maxSessionsPerBrowser	No	1	The maximum number of sessions (tabs/frames) allowed to long poll from the same browser; a negative value allows unlimited sessions.
multiSessionInterval	No	2000	Specifies the period of time, in milliseconds, for the client normal polling period, in case the server detects more sessions (tabs/frames) connected from the same browser than allowed by the `maxSessionsPerBrowser` parameter. A non-positive value means that additional sessions will be disconnected.

Mandatory: No
Specifies the configuration for the CometD-specific transport server embedded into the Web Services application. For example:

```
cometDSettings:
    cometdSessionExpirationTimeout: 60
    closeHttpSessionOnCometDExpiration: true
    maxSessionsPerBrowser: 2
    multiSessionInterval: 4000
```

And specifies how CometD's session cookie should be handled. For example:

```
cometDSettings:
    cookieHttpOnly: true
    cookieSecure: true
    cookieSameSite: None
```

Cookie options take effect only if **enableSsl** is set to true.

Log header

enableLogHeader

Default Value: `true`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether Web Services includes a header in its main log file. This header contains key information about the Web Services installation, including the version, start time, libraries, and any applicable settings from the **applications.yaml** file.

Routing Point Monitoring

enableRPMonitoring

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
Specifies whether Web Services will support call supervision for monitoring routing points.

Switch Prefixes

enableDialingPlanAvayaSwitchPrefixProcessing

Default Value: `false`
Valid Values: `true`, `false`
Mandatory: No
When enabled, Web Services will compare participant numbers with and without prefixes on outbound calls to avoid rendering redundant participants.

dialingPlanAvayaSwitchPrefix

Default Value: 9

Valid Values: Any positive integer
Mandatory: No
Specifies the outbound call prefix used on Avaya switches.

SIP Cluster Support

useEmployeeIdAsAgentLoginForSIPCluster

Default Value: false
Valid Values: true, false
Mandatory: No
Specifies if Web Services has to use the employee ID of the agent as the login ID.

> **Important**
> This option is used in SIP Cluster environments only.

onPremiseSettings

Settings in this section are listed under "onPremiseSettings".

countryCode

Default Value: None
Valid Values: A two-letter country code
Mandatory: No
The premise contact center's country code. For example, US.

Gplus Adapter for Salesforce

The Gplus Adapter for Salesforce is an integrated solution that enables Salesforce users to handle contact center interactions seamlessly within Salesforce. Beginning with version GWS 8.6 and later, the Gplus Adapter for Salesforce is based on Workspace Web Edition (WWE) only. It's included as part of the Web Services and Applications installation package.

The adapter provides a single integrated agent desktop experience that offers rich data integration for screen pops and dispositioning. It presents complete customer information at a glance to more effectively serve customers in Salesforce Console or Lightning modes. The adapter leverages the Web Services API and the Salesforce Open CTI API.

Gplus Adapter for Salesforce provides the full Workspace Web Edition interface and is available in Salesforce Console and Salesforce Lightning. It provides Salesforce-specific features such as updating activity history, screen pop, and click-to-dial, along with the full Workspace Web Edition user interface and the following features:

- Voice
- Chat
- Email
- Outbound Preview
- Voice and Chat Supervision (monitoring, coaching, barge-in)

> **Important**
>
> The Gplus Adapter URL in Salesforce Call Center follows this format: `https://<your company name>.genesyscloud.com/ui/crm-workspace/index.html`

Deployment tasks

Complete the following tasks to install and configure the adapter in your Genesys environment and in Salesforce.

1. Install and configure Web Services.
2. Make sure you set up SSL for Jetty — the adapter won't work without it. To set up SSL, configure the SSL section of the **application.yaml** file as follows:

```
enableSsl: true
ssl:
  port: 8043
  keyStorePath: bec.jks
  keyStorePassword: OBF:1g3p1kqt1xtl19q51ni31nlv19q91xtx1ku11fzt
  keyManagerPassword: OBF:1g3p1kqt1xtl19q51ni31nlv19q91xtx1ku11fzt
  trustStorePath: bec.jks
```

```
trustStorePassword: OBF:1g3p1kqt1xtl19q51ni31nlv19q91xtx1ku11fzt
```

For more information about configuring SSL, see Configure SSL.

If you want the adapter to use single sign-on, make sure you Configure SAML in Web Services. (You'll also need to add a special parameter to the **CTI Adapter URL** field in Step 4 of Configuring the adapter in Salesforce.)

> **Important**
>
> The Gplus Adapter URL in Salesforce Call Center follows this format: `https://<your company name>.genesyscloud.com/ui/crm-workspace/index.html`. If you're enabling single sign-on in the adapter, add the parameter `authType=saml` to the **CTI Adapter URL**. For example: `https://198.51.100.23:8090/ui/crm-workspace/index.html?crm=salesforce&authType=saml`

3. Install and configure the adapter in Salesforce.
4. Refer to the Web Services and Applications Configuration Guide for information about how to configure Workspace Web Edition and configure the adapter in your Genesys environment. The following Workspace Web Edition features are supported in the adapter:
   - Voice
   - Chat
   - Email
   - Outbound Preview
   - Voice and Chat Supervision (monitoring, coaching, barge-in)

   Test and confirm that your Workspace Web Edition configuration is valid and your required features are enabled.
5. Refer to the Gplus Adapters User Guide for information on how to work with the adapter.

Installing and configuring the adapter in Salesforce

Complete the procedures on this page to install and configure the Gplus Adapter in your Salesforce environment.

If you want to enable Gplus Adapter in Salesforce Lightning after you install and configure the adapter in Salesforce, go here.

The adapter installation procedure involves the following steps:

1. Creating a Gplus Adapter URL.
2. Configuring Gplus Adapter for Salesforce.
3. Adding users to Call Center.
4. Configuring the Utility bar.
5. Configuring the whitelist domain for your Salesforce Console.
6. Configuring screen pops in Salesforce.
7. Accessing Gplus Adapter for Salesforce.

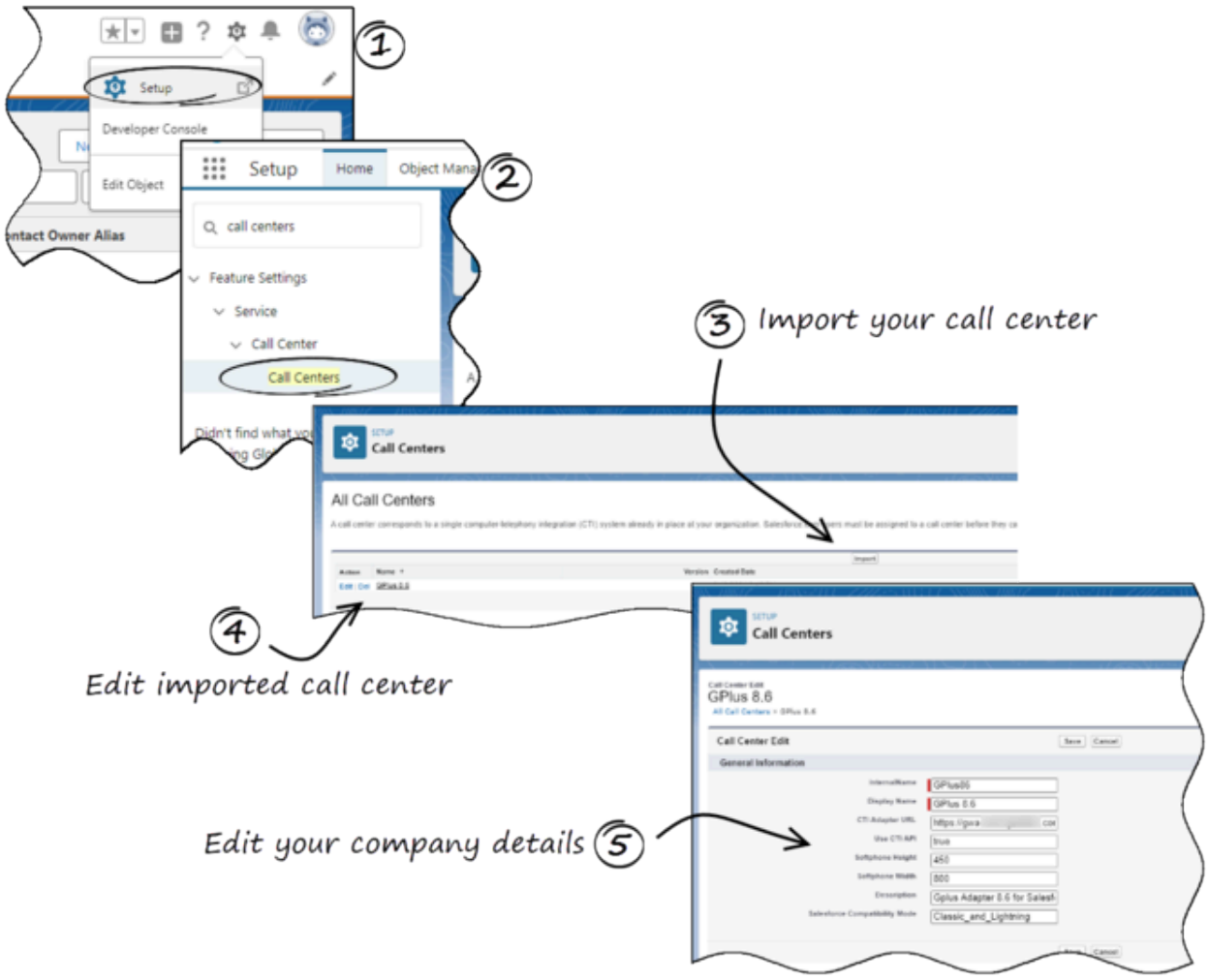
Creating a Gplus Adapter URL

From the Agent Workspace URL, copy the domain name and substitute it in the following URL:

```
https://<domain-name>/ui/crm-workspace/index.html?crm=salesforce
```

Configuring Gplus Adapter for Salesforce

Follow these steps:

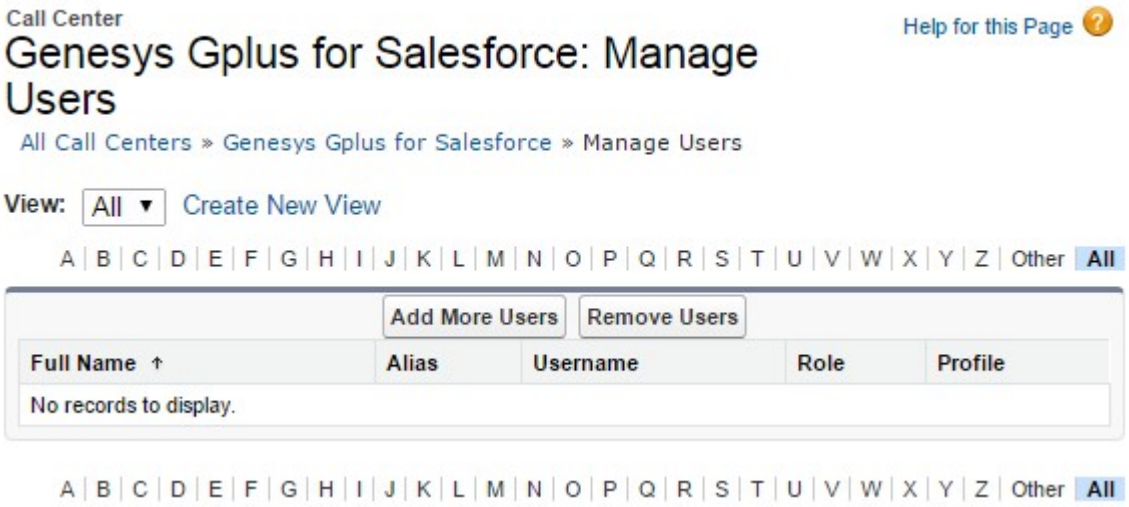


1. Click the gear icon in the top right corner and then click **Setup**.
2. Using the **Quick Find** field, search for and access the **Call Centers setup** page.
3. From the **Call Center** settings page, using the **Import** functionality, import the crm-workspace-callcenter.xml file from your computer. If you have not already downloaded the file, download it from here.
4. From the **All Call Centers** list, click the **Call Center** you just imported. For example, Gplus86.
5. From the **Call Center Edit** page, in the **CTI Adapter URL** field, specify the Adapter URL that you constructed in the **Creating a Gplus Adapter URL** section. For example, https://www.genesysgplustest.com/ui/crm-workspace/index.html?crm=salesforce
6. Set **Salesforce Compatibility mode** to **Classic_and_Lightning**.
7. Save the changes.

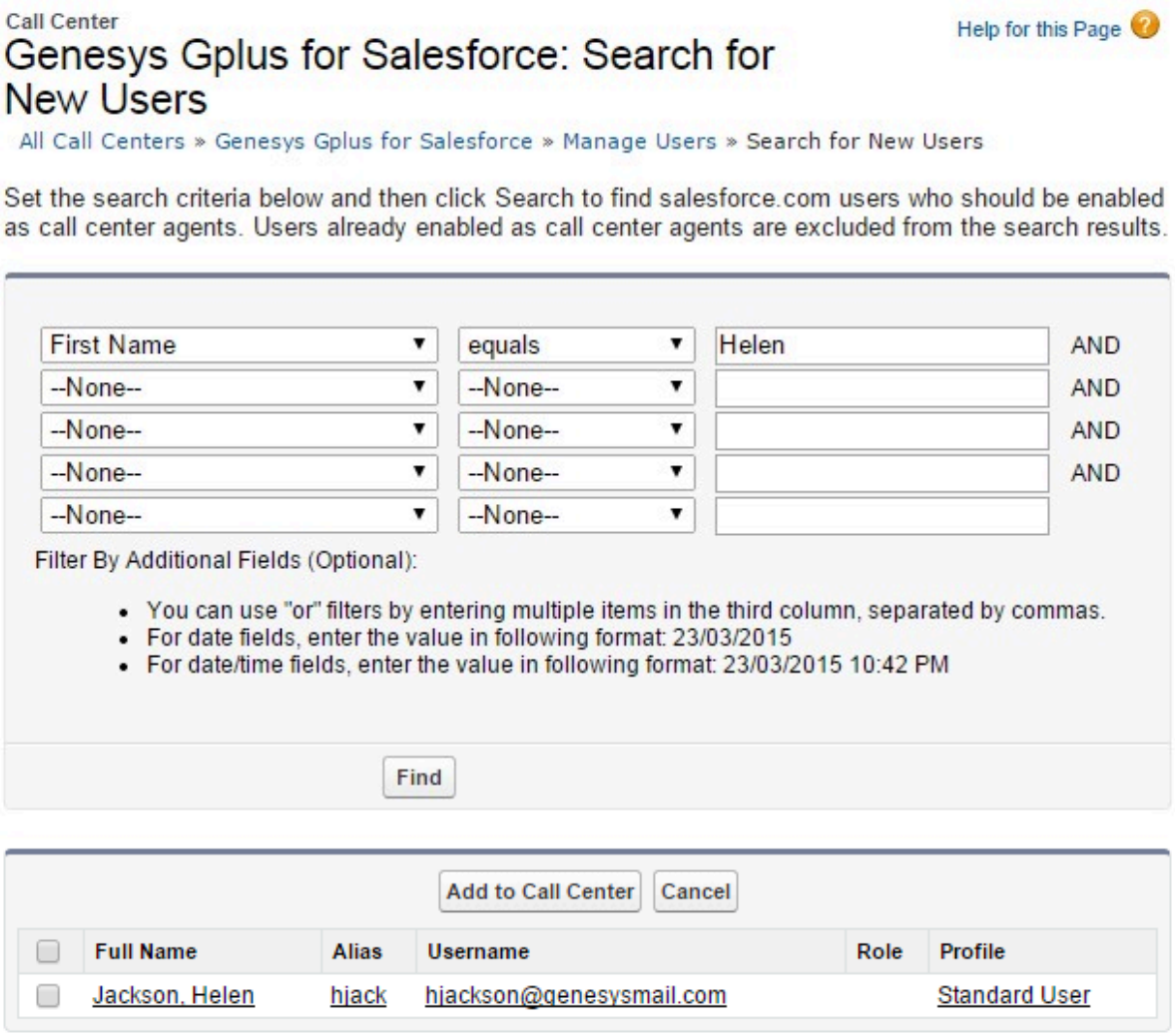
Adding users to Call Center

The next step after you setup your Adapter is to add users to your call center. You must add at least one user to your call center.

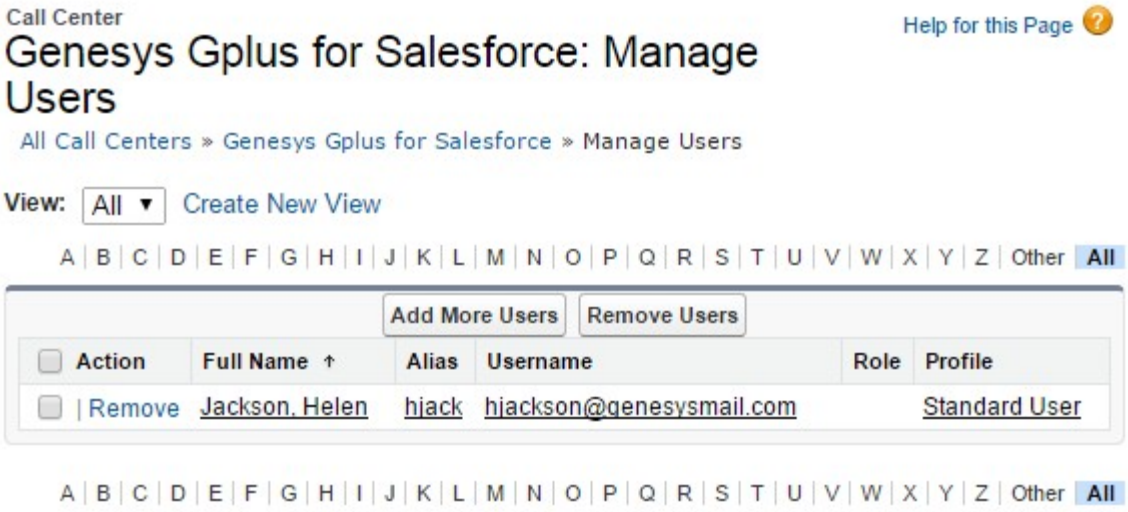
1. In the **Call Centers setup** page, click **Manage Call Center Users**.
2. Click **Add More Users**.



3. On the **Search for New Users** page, you can enter search criteria to find users. Select the ones you want to add to the Gplus Adapter for Salesforce.
4. Click **Add to Call Center**.

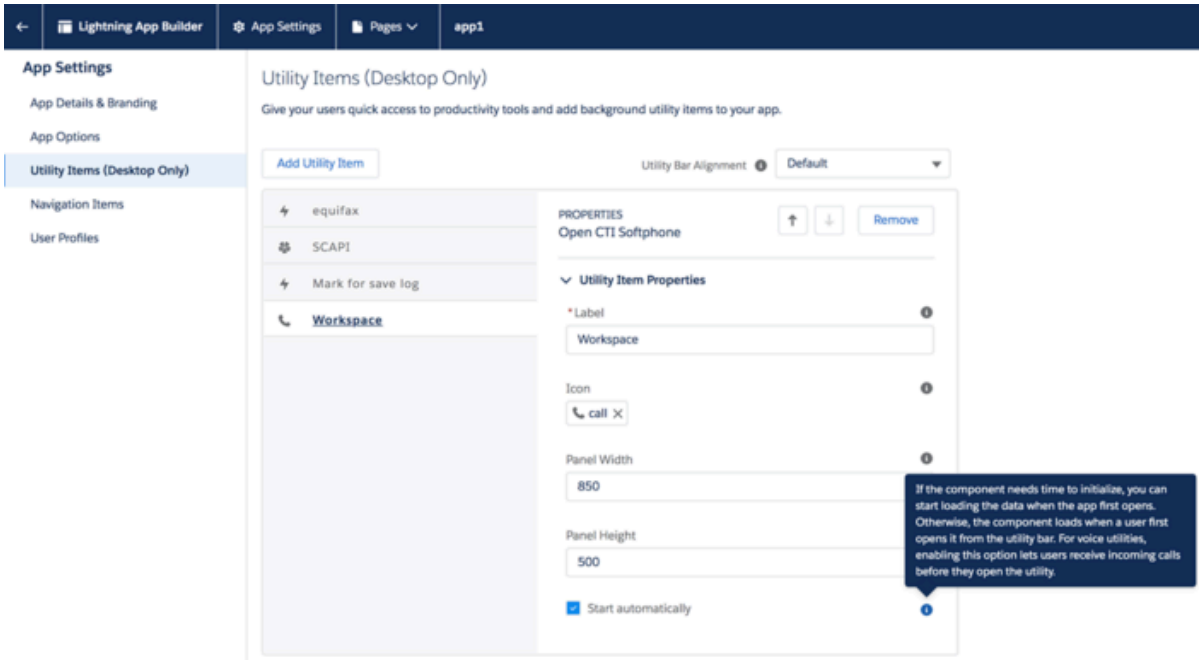


5. Your selected users are added to the list. You can remove a user on this page at any time.



Configuring the Utility bar

1. In the setup page, using the **Quick Find** field, search for and access the **App Manager settings** page.
2. Create a new application by clicking **New Lightning App**. Follow the steps in the Lightning App creation procedure in Salesforce documentation.
3. When creating a new Lightning app, follow the additional steps below to configure the utility items for your Gplus Adapter for Salesforce.
    - In the **App Options** step, ensure that you select **Console Navigation**. Gplus Adapter for Salesforce does not support Standard Navigation.
    - In the **Utility Items** step, add **Open CTI Softphone** by clicking **Add Utility Item**.
    - For the soft phone, you can modify the display properties such as **Label**, **Panel Width**, and **Panel Height**.
    - Select **Start automatically** to start loading the data when the Gplus Adapter first opens to ensure that it initializes immediately; otherwise, Gplus Adapter does not load until an agent first opens it from the **Utility Bar**. If Gplus Adapter for Salesforce is not initialized immediately, then agents do not receive voice calls until they first open the application.



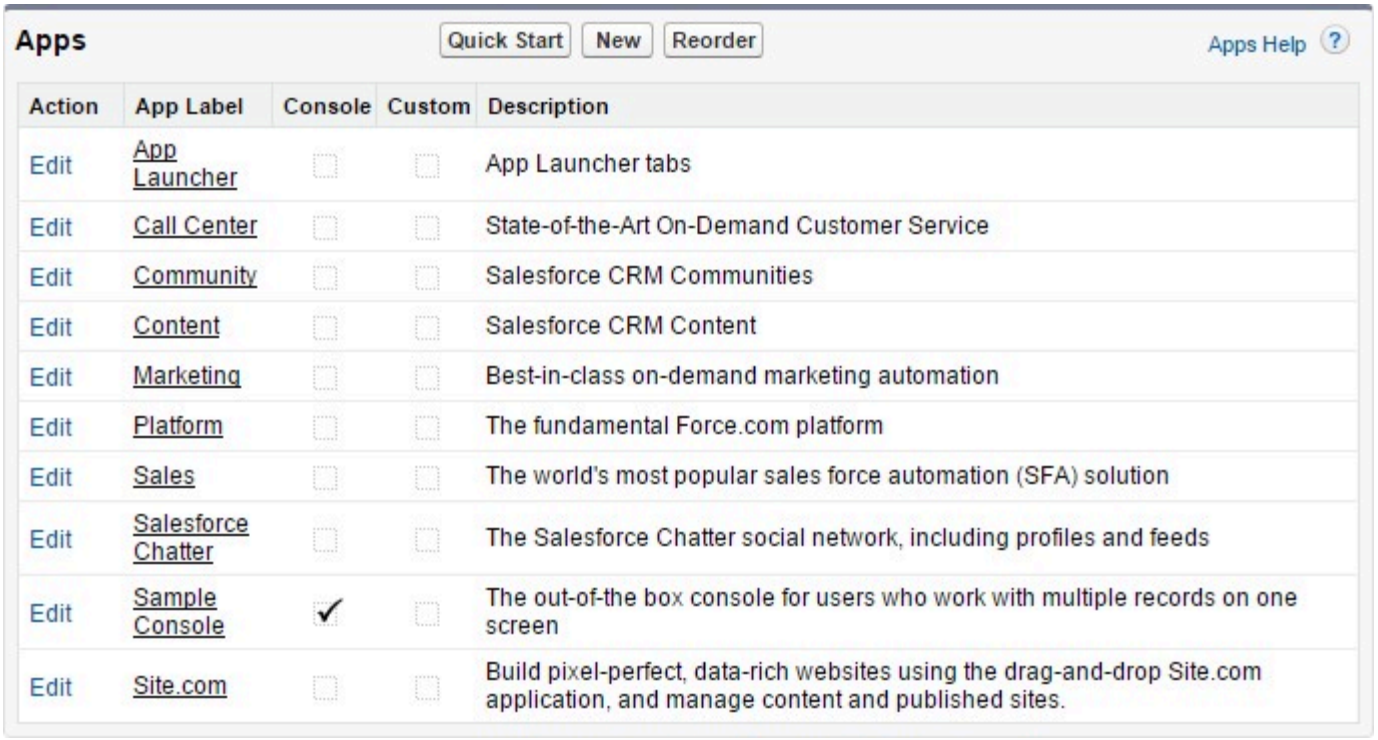
4. Specify **Navigation Items** and **User Profiles** for the new Lightning app by referring to Salesforce documentation.
5. Save the application.

Configuring the whitelist domain for your Salesforce Console

Complete this procedure to add the Genesys domain to the whitelist domains for your Salesforce Console. You need to complete this procedure to allow your users to access the adapter in Salesforce Console in a separate browser window.

Start

1. If you haven't already, login to Salesforce and go to **App Setup > Create > Apps** and select your console app — "Sample Console" in the image below:



2. Click **Edit**. In **Whitelist Domains**, add the host and port for your installation of Web Services. For example: `198.51.100.23:8090`
3. Click **Save**.

End

Configuring screen pops in Salesforce

When an agent receives an external call, the adapter can initiate a screen pop that causes Salesforce to show an appropriate record for the caller. To set up this functionality in Salesforce, login and go to **Setup > Customize > Call Center > SoftPhone Layouts** to create a SoftPhone Layout. Check out the Salesforce documentation for details about configuration.

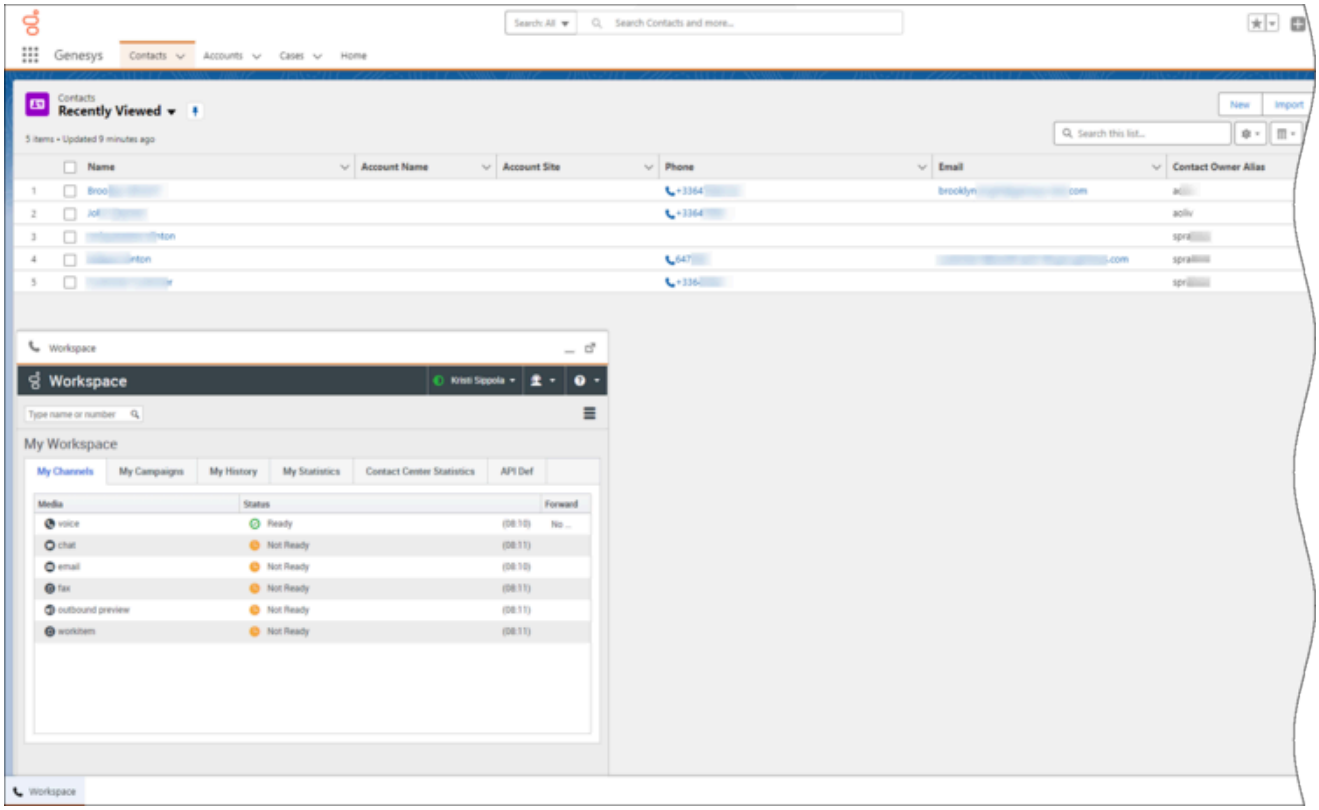
In general, there are a couple of things to consider when you set up a SoftPhone Layout for the adapter:

- The Gplus Adapter for Salesforce ignores the SoftPhone Layout settings that control call-related fields. Instead, the adapter gets this information from toast and case data you configure in the Genesys environment.
- Make sure you configure the **Screen Pop Settings** in the "CTI 2.0 or Higher Settings" section. These settings control whether the screen pop opens in a new window, tab, or Visualforce page.

See Screen pop for more information about configuring screen pops in your Genesys environment.

Accessing Gplus Adapter for Salesforce

You can access Gplus Adapter for Salesforce in Console or Lightning mode by clicking the phone icon in the bottom-left corner.



Enabling Lightning Experience

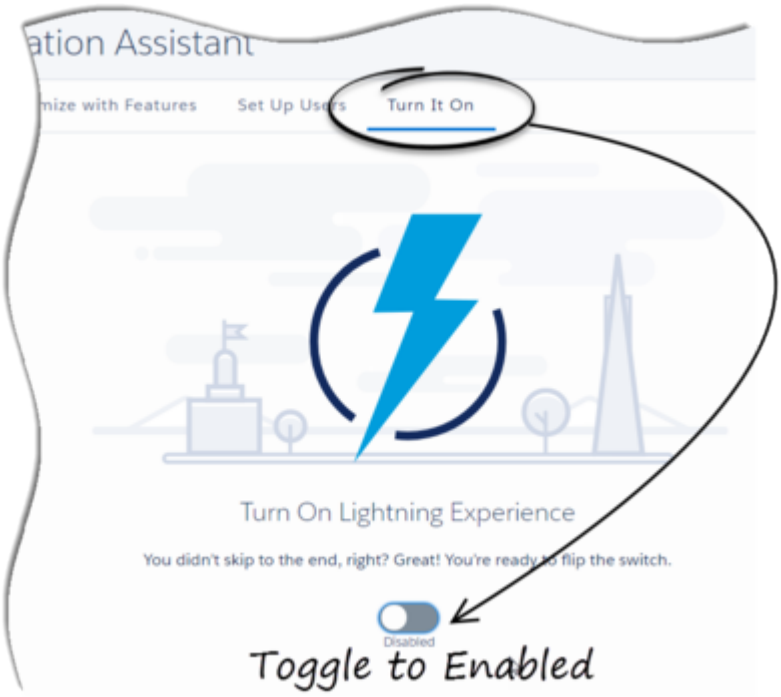
If you're using the Gplus Adapter for Salesforce in Console already, complete the procedures on this page to enable, set up, and access Lightning in your Salesforce environment.

> **Important**
> The Gplus Adapter URL in Salesforce Call Center follows this format: `https://<your company name>.genesyscloud.com/ui/crm-workspace/index.html?crm=lightning`

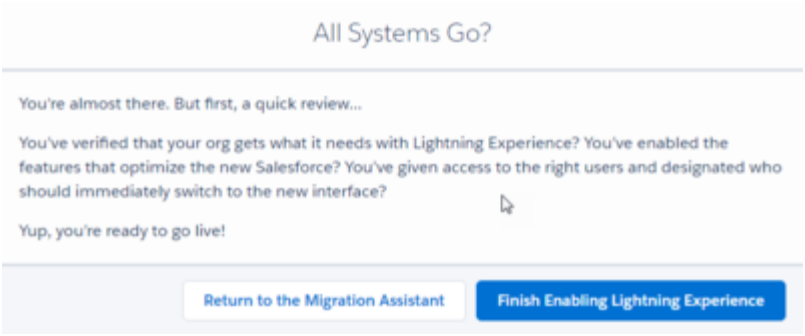
Enabling Lightning in Salesforce

Start

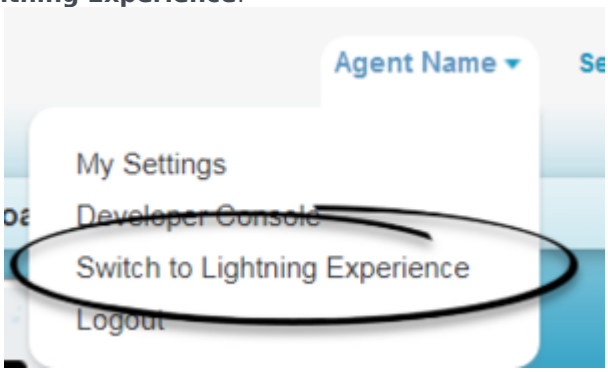
1. Log into the Salesforce environment.
2. From the **Setup** page, select **Lightning Experience** in the left-hand navigation bar.
3. In the **Lighting Experience** window, select **Turn It On**.
4. Move the toggle to the **Enabled** state.



5. A modal will pop up; click the **Finish Enabling Lightning Experience** button in the modal.



6. In the dropdown labeled with the agent's name at the top of the Salesforce Console, click **Switch to Lightning Experience**.



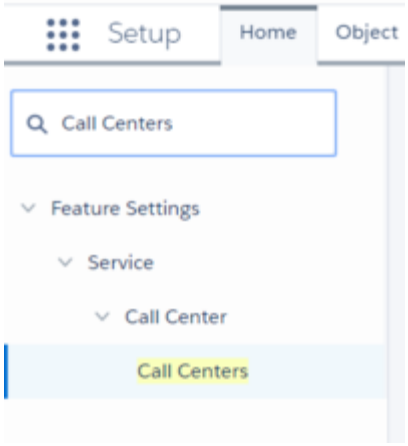
End

Setting Up The Adapter In Lightning

Prerequisite

Download the **lightning-callcenter.xml** file on your computer by right-clicking the link lightning-callcenter.xml and selecting the **Save link as...** option on the popup menu.

1. Go to the **Setup** page by clicking on the gear icon in the top right corner and clicking **Setup**.
2. Using the quick find field, search for and access the **Call Centers** settings page.

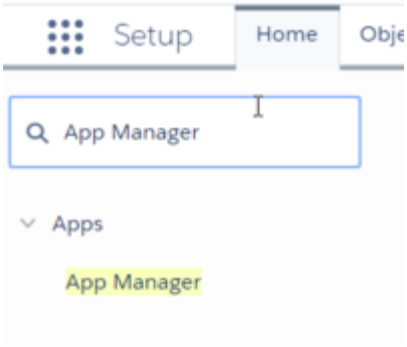


3. Click **Import**.
4. Click **Choose File**.
5. Select the **lightning-callcenter.xml** file downloaded on your computer. If you have not already downloaded the file, right-click the link lightning-callcenter.xml and select the **Save link as...** option to download.
6. Click **Import**.
7. From **All Call Centers** list, click the call center you just imported. For example, **GPlusLightning**.
8. Click **Edit**.
9. In the **CTI Adapter URL** field, replace 'GWSHOST' and 'GWSPORT' with the host and port details of the adapter in your environment. For example, an updated URL will look like this: `https://bec135-gws.live.genesys.com/ui/crm-workspace/index.html?crm=lightning`
   Note: If you are deploying the adapter with Single-Sign-On (SSO) capability, ensure that you add the `&authType=saml` parameter at the end of the CTI Adapter URL. For example, an updated URL with SSO capability will look like this: `https://bec135-gws.live.genesys.com/ui/crm-workspace/index.html?crm=lightning&authType=saml`
10. Click **Save**.
11. Click **Manage Call Center Users**.
12. Click **Add More Users**.
13. Search the interface to find the users you want to add to the Lightning adapter.

> **Important**
> A user cannot be added to both the Lightning and non-Lightning adapters

.

14. Select the users you want to add and click **Add to Call Center**.
15. Using the quick find field, access the **App Manager** settings page.



16. In the apps list, click the **Show more actions** drop down on the far right side of the adapter app you wish to use.
    Note: If you do not see any apps in the list, you can create one by clicking **New Lightning App**.
17. Click **Edit**.
18. Click **Utility Bar**.
19. From the Utility Bar window, click **Add** and select "Open CTI Softphone".
20. Change the **Label** field to "Workspace".
21. Click **Done**.

Accessing the Adapter

Start

1. In the top-left corner, click the **App Launcher** icon:
   
2. Select the app that you created when setting up the adapter.
3. Click **Workspace** from the bar at the bottom-left to open the adapter.
4. Log in to the Adapter.

End

Monitoring

If you have installed the Observability Solution following Installing Observability, you would have access to dashboards designed specifically for Genesys Web Services (GWS) 8.6.

This article provides detailed screenshots of dashboards for different metrics.

Dashboards

Node Exporter full

The following screenshot displays the infrastructure/OS-level dashboard which describes system level metrics, for example, CPU, memory, and network.

GWS overview

The following screenshot displays the GWS 8.6 Application Overview dashboard which describes GWS 8.6 specific performance and application metrics.

GWS instances

The following screenshot displays the detailed GWS 8.6 Application dashboard which contains an in depth GWS 8.6 telemetry based on both logs and metrics.

Note: Additional optional dashboards are available for Datastores such as Redis and ElasticSearch and they are provided in the installation package.

Explore mode

By clicking the **Explore Mode** button, users can leverage the power of both log and metrics query languages (LogQL and PromQL) to create their own custom telemetry requests.

Data Export and Import Process

This is a feature included in the installation package which allows you to transmit telemetry data to Genesys on need basis in case Genesys support requires it.

Note that the Export and Import feature does not redact sensitive data. To use this feature, follow the instructions given as part of the Observability installation package here: **OBS_instance/import_export_scripts/README.md**

Troubleshooting

This page provides solutions to common problems in Web Services and Applications.

The following log for GWS API Service is saved to the **./log** directory on the Web Services node, unless configured differently as part of the deployment.

- **gws-api-v2.log** — Stores WARN level messages about Web Services.

To modify the log message levels, you can edit the **logback.xml** file and change the level to DEBUG or TRACE (instead of WARN):

```
<logger name="com.genesyslab" level="DEBUG" />
```

The following log for GWS Platform Service is saved to the **/var/log/gws-service-platform** directory on the Web Services node, unless configured differently as part of the deployment. To modify the log message levels, you can change the **GWS_LOG_LEVEL** environment variable to the following allowed values: `ERROR, WARN, INFO, DEBUG, TRACE .`

For Golden Signals to help with troubleshooting Web Services and Applications, see Monitoring section.

Caching

Common troubleshooting steps

- **Cache invalidation**: The Cache invalidation operation cleans all caches where Configuration data is stored. However, it doesn't clean voice or mutlimedia context but only Configuration data.

Method	POST
Request mapping	.../api/v2/ops/cache
Permissions	Cloud Administrator
Required attributes	operationName (should be "ResetCache")

Examples

```
POST .../api/v2/ops/cache

{
   "operationName": "ResetCache"
}
```

Appendix: How-to Create SSL Certificate

Prerequisites

- Create the root pair (rootCA key & rootCA cert).
- Prepare the `mkdir /root/ca` directory.
- Create the directory structure:

```
# cd /root/ca
 # mkdir certs crl newcerts private
 # chmod 700 private
 # touch index.txt
 # echo 1000 > serial
```

- Copy the root CA configuration (openssl.cnf) to /root/ca/openssl.cnf
- Create the root key:

```
# cd /root/ca
 # openssl genrsa -aes256 -out private/<rootCA>.key.pem 4096
```

- Enter pass phrase for `<rootCA>.key.pem`: <Enter password>
- Verifying - Enter pass phrase for `<rootCA>.key.pem`: <Enter password>

```
# chmod 400 private/<rootCA>.key.pem
```

Create Root Certificate

- Use the `<rootCA>.key.pem` root key to create the `<rootCA>.cert.pem` root certificate.

```
# cd /root/ca
 # openssl req -config openssl.cnf -key private/<rootCA>.key.pem -new -x509 -days 7300
 -sha256 -extensions v3_ca -out certs/<rootCA>.cert.pem
```

- Enter the pass phrase for `<rootCA>.key.pem`: <password for “rootCA.key.pem”>

```
You are about to be asked to enter information that will be incorporated
 into your certificate request.
 -----
 Country Name (2 letter code) [XX]: <Enter country code>
 State or Province Name []: <Enter state or province>
 Locality Name []: <Enter city>
 Organization Name []: <Enter company name>
 Organizational Unit Name []: <Enter company OU>
 Common Name []: <Enter some value>
 Email Address []: <Enter admin mail account>
```

```
# chmod 444 certs/<rootCA>.cert.pem
```

Verify Root Certificate

```
# cd /<rootCA>.cert.pem
```

The output shows:

- The `Signature Algorithm` used
- The dates of certificate `Validity`
- The `Public-Key` bit length
- The `Issuer`, which is the entity that signed the certificate
- The `Subject`, which refers to the certificate itself

The `Issuer` and `Subject` are identical as the certificate is self-signed. Note that all root certificates are self-signed.

```
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=GB, ST=England,
            O=Alice Ltd, OU=Alice Ltd Certificate Authority,
            CN=Alice Ltd Root CA
    Validity
        Not Before: Apr 11 12:22:58 2015 GMT
        Not After : Apr  6 12:22:58 2035 GMT
    Subject: C=GB, ST=England,
             O=Alice Ltd, OU=Alice Ltd Certificate Authority,
             CN=Alice Ltd Root CA
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (4096 bit)
```

The output also shows the `X509v3 extensions`. We applied the `v3_ca` extension, so the options from `[ v3_ca ]` should be reflected in the output.

```
X509v3 extensions:
    X509v3 Subject Key Identifier:
        38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31
    X509v3 Authority Key Identifier:
        keyid:38:58:29:2F:6B:57:79:4F:39:FD:32:35:60:74:92:60:6E:E8:2A:31

    X509v3 Basic Constraints: critical
        CA:TRUE
    X509v3 Key Usage: critical
        Digital Signature, Certificate Sign, CRL Sign
```

GWS Key and Certificate Generation

- Make a directory for GWS files:

  ```
  # cd /root/ca # mkdir gwsCerts
  ```

- Create a key:

  ```
  # cd /root/ca
  ```

```
# openssl genrsa -aes256 -out gwsCerts/<gwsKey>.key.pem 2048
# chmod 400 gwsCerts/<gwsKey>.key.pem
```

- Create a certificate (CSR):
  Requirement
  the Common Name must be a fully qualified domain name.

  Copy san.cnf and v3.ext to /root/ca and modify the following parameters in these files

```
commonName = <Enter FQDN of your GWS host>
DNS.1   = commonName
DNS.2   = *.<part of FQDN>
```

```
# cd /root/ca
# openssl req -out gwsCerts/<gwsCSR>.csr -newkey rsa:2048 -nodes -keyout
gwsCerts/<gwsKey>.key.pem -config san.cnf
```

- **Enter pass phrase for <gwsKey>.key.pem**
  <password for gws Key>

  You are about to be asked to enter information that will be incorporated into your certificate request.

```
Country Name (2 letter code) [XX]: <Enter country code>
State or Province Name []: <Enter state>
Locality Name []: <Enter city>
Organization Name []: <Enter company>
Organizational Unit Name []: <Enter company OU>
Common Name []: <Enter FQDN of your GWS host>
Email Address []: <Enter email address>
```

- Sign the GWS CSR file:
  Use rootCA authority to sign up GWS csr file.

```
# cd /root/ca
# openssl x509 -req -sha256 -days 367 -in gwsCerts/<gwsCSR>.csr -CA <full path to
'rootCA.cert.pem'> -CAkey <full path to 'rootCA.key.pem'> -CAcreateserial -out
gwsCerts/<gwsSignedCert>.pem -extfile v3.ext -extensions v3_req
# chmod 444 gwsCerts/<gwsSignedCert>.cert.pem
```

Example:

```
openssl x509 -req -sha256 -days 367 -in gwsCerts/gwsCSR.csr -CA /root/ca/certs/
rootCA.cert.pem -CAkey /root/ca/private/rootCA.key.pem -CAcreateserial -out gwsCerts/
gwsSignedCert.pem -extfile v3.ext -extensions v3_req
# chmod 444 gwsCerts/gwsSignedCert.pem
```

- Verify the certificate:

```
# openssl x509 -noout -text -in gwsCerts/<gwsSignedCert>.pem
Check for x509v3 extensions (SAN & v3 extensions).
```

Converting Procedures

- Convert the existing cert to a PKCS12 using OpenSSL.

> **Important**
> A password is required.

```
# cd /root/ca
 # openssl pkcs12 -export -in <gwsSignedCert>.pem -inkey <gwsKey>.key.pem -out
 <keystore.p12> -name <certAlias> -CAfile <full path to 'rootCA.cert.pem'> -caname rootCA
```

Example:

```
# openssl pkcs12 -export -in /root/ca/gwsCerts/gwsSignedCert.pem -inkey /root/ca/gwsCerts/
 gwsKey.key.pem -out keystore.p12 -name firstcert -CAfile /root/ca/certs/rootCA.cert.pem
 -caname rootCA
```

- Convert the PKCS12 to a Java Keystore File.

```
# cd /root/ca
 # keytool -importkeystore -deststorepass <new_keystore_pass> -destkeypass <new_key_pass>
 -destkeystore <gwsKeystore.jks> -srckeystore <keystore.p12> -srcstoretype PKCS12
 -srcstorepass <pass_used_in_p12_keystore> -alias <alias_used_in_p12_keystore>
```

Example:

```
keytool -importkeystore -deststorepass password -destkeypass password -destkeystore
 gwsKeystore.jks -srckeystore keystore.p12 -srcstoretype PKCS12 -srcstorepass password
 -alias firstcert
 * System will automatically tell to change the format:<source lang="text">
 keytool -importkeystore -srckeystore gwsKeystore.jks -destkeystore gwsKeystore.jks
 -deststoretype pkcs12
```

Import Missing Certs and Create Truststore

- Import rootCa certificate to gwsKeystore.jks:
- Use keytool -importcert to import the rootCa certificate into each node keystore:

```
# cd /root/ca
 # keytool -importcert -keystore <gwsKeystore>.jks -alias rootCA -file <path to
 'rootCA.cert.pem'> -noprompt -keypass <keystore password> -storepass <password>
```

Example:

```
# keytool -importcert -keystore gwsKeystore.jks -alias rootCA -file /root/ca/certs/
 rootCA.cert.pem -noprompt -keypass password -storepass password
```

- Create a server truststore:

```
# cd /root/ca
 # keytool -importcert -keystore <gwsTruststore>.jks -alias rootCA -file
 <rootCA>.cert.pem -noprompt -keypass <key password> -storepass <password>
```

Example:

```
# keytool -importcert -keystore gwsTruststore.jks -alias rootCA -file /root/ca/certs/
 rootCA.cert.pem -noprompt -keypass password -storepass password
```

GWS Configuration (application.yaml)

- Configuration example (**jetty** section):

```
enableSsl: true
  ssl:
    port: 8443
    securePort: 443
    idleTimeout: 30000
    soLingerTime: -1
    trustAll: true
    keyStorePath: /root/ca/<gwsKeystore>.jks
    keyStorePassword: <keystore password>
    keyStoreType: JKS
    trustStorePath: /root/ca/<gwsTruststore>.jks
    trustStorePassword: <truststore password>
```

Example:

```
# caCertificate: /root/ca/myKeystore.jks
 # jksPassword: Manila@1234
 port: 8443
 securePort: 443
 idleTimeout: 30000
 soLingerTime: -1
 trustAll: true
 keyStorePath: /root/ca/gwsKeystore.jks
 keyStorePassword: password
 keyStoreType: JKS
 trustStorePath: /root/ca/gwsTruststore.jks
 trustStorePassword: password
```

On Client Desktop

- Add your host (hostname should be specified as FQDN) in <%system_drive%>\Windows\System32\drivers\etc\hosts.
  Example in the file

```
192.168.100.26      gws-centos7.genesys.com
```

- Convert <rootCA>.cert.pem to PFX format:

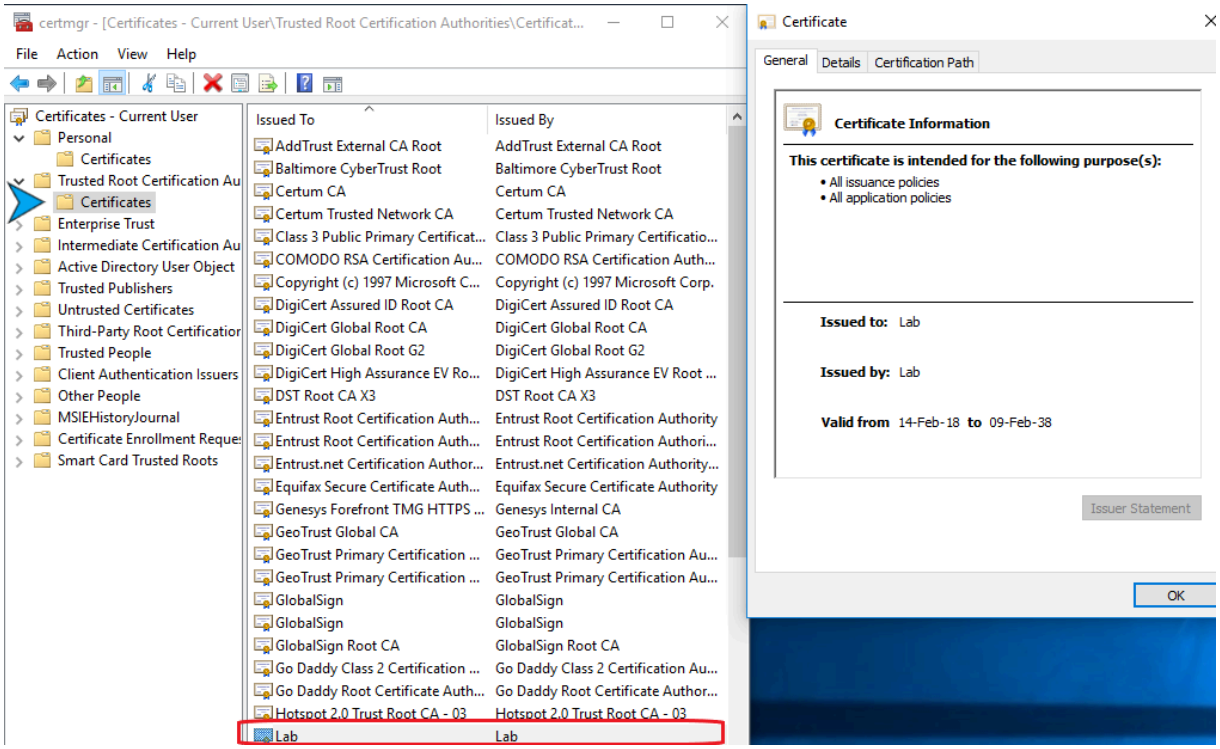
```
# cd /root/ca
 # openssl pkcs12 -inkey <rootCA>.key.pem -in <rootCA>.cert.pem -export -out <rootCA>.pfx
```

Example:

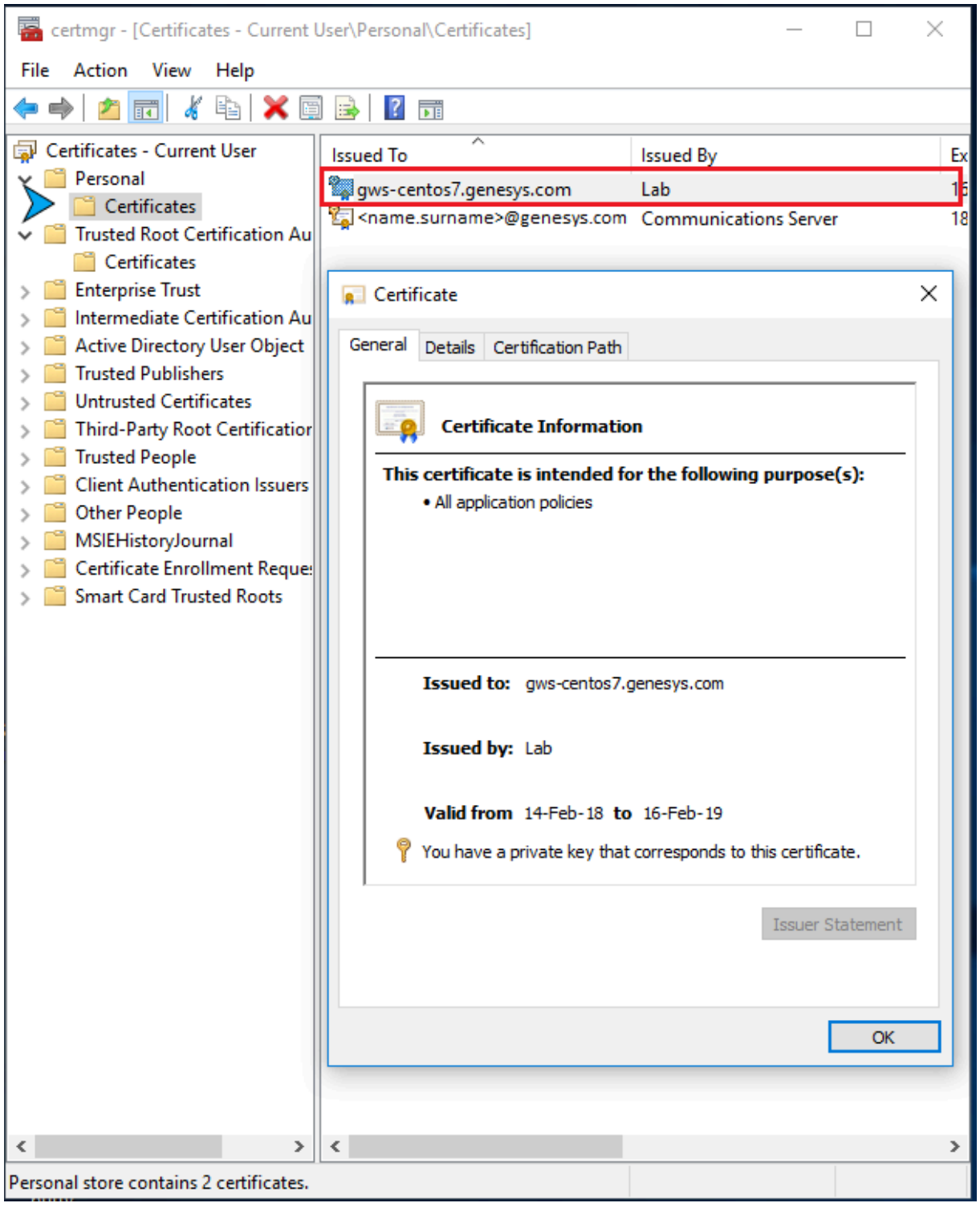
```
openssl pkcs12 -inkey /root/ca/private/rootCA.key.pem -in /root/ca/certs/rootCA.cert.pem
 -export -out rootCA.pfx
```

- Copy <rootCA>.pfx and <keystore>.p12 to Windows host.
- Import the <keystore>.p12 file by double-clicking on it. Use default configuration and specify the password.
- Import the <rootCA>.pfx file, make sure to select “Place all certificates in the following store”. Browse “Trusted Root Certification Authorities”

- Verify that certificates are present using `certmgr.msc`.
  Example:
  - For rootCA certificate:



  - For GWS certificate:


certmgr - [Certificates - Current User\Personal\Certificates]
File Action View Help
Certificates - Current User
Personal
Certificates
Trusted Root Certification Au
Certificates
Enterprise Trust
Intermediate Certification Au
Active Directory User Object
Trusted Publishers
Untrusted Certificates
Third-Party Root Certificatior
Trusted People
Client Authentication Issuers
Other People
MSIEHistoryJournal
Certificate Enrollment Reque
Smart Card Trusted Roots
Issued To
Issued By
gws-centos7.genesys.com
Lab
<name.surname>@genesys.com
Communications Server
Certificate
General
Details
Certification Path
Certificate Information
This certificate is intended for the following purpose(s):
• All application policies
Issued to: gws-centos7.genesys.com
Issued by: Lab
Valid from 14-Feb-18 to 16-Feb-19
You have a private key that corresponds to this certificate.
Issuer Statement
OK
Personal store contains 2 certificates.