Genesys Mobile Services Deployment Guide

Genesys Mobile Engagement 8.5.0

1/15/2022

Table of Contents

Genesys Mobile Services Deployment Guide

Welcome to the Genesys Mobile Services Deployment Guide! This deployment guide can be used to install Genesys Mobile Services on your system, configure basic settings, as well as configure more advanced settings. It includes chapters with the following information:

- New in This Document — Provides a document change history.
- Planning information — Details related to planning and preparation for your Genesys Mobile Services installation, including prerequisites and multi-site deployment. Genesys recommends reading the prerequisites page before you begin to ensure that your system meets the minimum requirements for Genesys Mobile Services.
- Installation procedures — Step-by-step guide to installing Genesys Mobile Services and performing required configurations.
- Configuration — Includes additional configuration topics that you may wish to consider for your GMS installation.
- Configuration Options Reference — Provides a reference of the configuration options available for Genesys Mobile Services.
- Security — Provides security configurations that can be used with GMS.
- Starting and Stopping — Describes how to start and stop GMS using the Solution Control Interface or Genesys Administrator, and provides information about possible alarms.
- Troubleshooting — Frequently asked questions.
- Migrating GMS — Migration instructions.
- Testing your deployment — Describes how to test your installation by configuring and using samples.

New in This Document

- The Admin UI page has been moved to the Service Management Help.

The following topics have been added or changed in the GMS 8.5.006.09 release:

- Step 2 in the procedure Deploying DFM Files was updated.
- The Reporting section in the Service Management User Interface page was updated.
- A new alarm was added to the GMS Alarms page.
- The Configuration Options page was updated in the following sections:
  - GMS Section
  - Service Section
- The Basic Authentication page was updated with an important note about usernames/passwords.
- A new section, Testing Your Deployment has been added.
- The Execute CQL Commands section was updated under Authentication on External Cassandra.
- The Genesys Environment Prerequisites were updated.

The following topics have been added or changed in the GMS 8.5.005.08 release:

- The Genesys Environment Prerequisites were updated.
- The procedures in the Creating an Application Object page were updated for the role-based access feature. For Cluster deployments, see *Creating and Configuring the GMS Application Object (Cluster)*, Step 17, or for Single-node deployments, see *Creating and Configuring the GMS Application Object (Single Node)*, Step 12.
- Setting the ORS Option procedure was updated.
- Step 1, in the procedure Web API Server Configuration was updated.
- The Service Management User Interface page had the following changes to reflect the redesign of the UI:
  - Login URL's were updated.
  - The name of the *Home* tab changed to Monitor.
  - The Callback section was updated.
- A Limitations section was added to the *Configuring and Starting a Cluster* page.
- Configuration Options were updated for mobile push notifications to support non CometD-based (polling) chat sessions.
- The Troubleshooting page contains new entries.

Planning Your Deployment

For the 8.x release, Genesys Mobile Services allows you to develop mobile applications that take advantage of Genesys capabilities. Every Genesys product also includes a Release Note that provides any late-breaking product information that could not be included in the manual. This product information can often be important. To view it, open the `read_me.html` file in the application home directory, or follow the link under the Release Notes section of the product page to download the latest Release Note for this product.

What You Should Know

This guide is written for software developers and application architects who intend to create mobile applications that interact with Genesys environments. Before working with Genesys Mobile Services, you should have an understanding of:

- computer-telephony integration (CTI) concepts, processes, terminology, and applications
- mobile concepts and programming
- network design and operation
- your own network configurations
- Genesys Framework architecture

In addition to being familiar with your existing Genesys environment, it is a good idea to be aware of some of the security issues that are involved with deploying a Genesys Mobile Services-based solution. That information and an overview of the deployment topology are discussed under Security and Access Control.

Prerequisites

To work with Genesys Mobile Services (GMS), you must ensure that your system meets the software requirements established in the Genesys Supported Operating Environment Reference Manual, as well as meeting the following minimum requirements:

Hardware Requirements

The following are minimum requirements:

- CPU: Quad core
- Memory: 4GB
- Disk: 160GB
- At least 2-3 nodes recommended for redundancy and availability

OS Requirements

- Genesys Supported Operating Environment Reference Guide

> **Important**
> For Linux installations, the Linux compatibility packages must be installed prior to installing the Genesys IPs.

Browser Support

- Genesys Supported Operating Environment Reference Guide

Java Requirements

- GMS requires a JDK.
- GMS is compatible with the latest version of JDK 7.

Genesys Environment

In addition to having a Genesys Management Framework 8.1 environment installed and running, the following table lists the Genesys components that are used with a GMS installation.

Genesys Component	Minimum Version Required	Comments
Orchestration Server (ORS)	8.1.300.48	Mandatory, installed and running: • An HTTP port must be enabled in the related Application object. • The ORS server must use the `Orchestration Server` type in Configuration Manager.
Universal Routing Server (URS)	8.1.300.35	Mandatory, required for the GMS services.
Interaction Routing Designer (IRD)	8.1.400.12	Mandatory, required for strategies running on URS.
SIP Server	8.1.100.67	Mandatory, required for route point calls and incoming calls.
Chat Server	8.1.000.26	Used for Chat support.
Web API Server	8.1.200.05	Used for Chat support.
Interaction Server	8.0.200.11	Used for Chat support.
Stat Server	8.x	Used to obtain statistics.
Media Server	8.1.410.33	Used for Callback services, in order to play treatments and use Call Progress Detection (CPD) for outbound calls.
Resource Manager	8.1.410.13	Used for Callback services, in order to play treatments and use Call Progress Detection (CPD) for outbound calls.
Interaction Workspace	8.1.401.20	Used for Callback services.

Multi-site Deployment

For a multi-site deployment, you must consider the following questions:

- How are API requests from mobile devices going to be routed?
- How is the mobile/web client going to retry the request in case of network failure?
- How will the client find the addresses for the global load balancer or site load balancer?
- How is the telephony network going to route the call to the number the client is receiving through the data API call?
- Is the call origination direction user-originated or user-terminated?

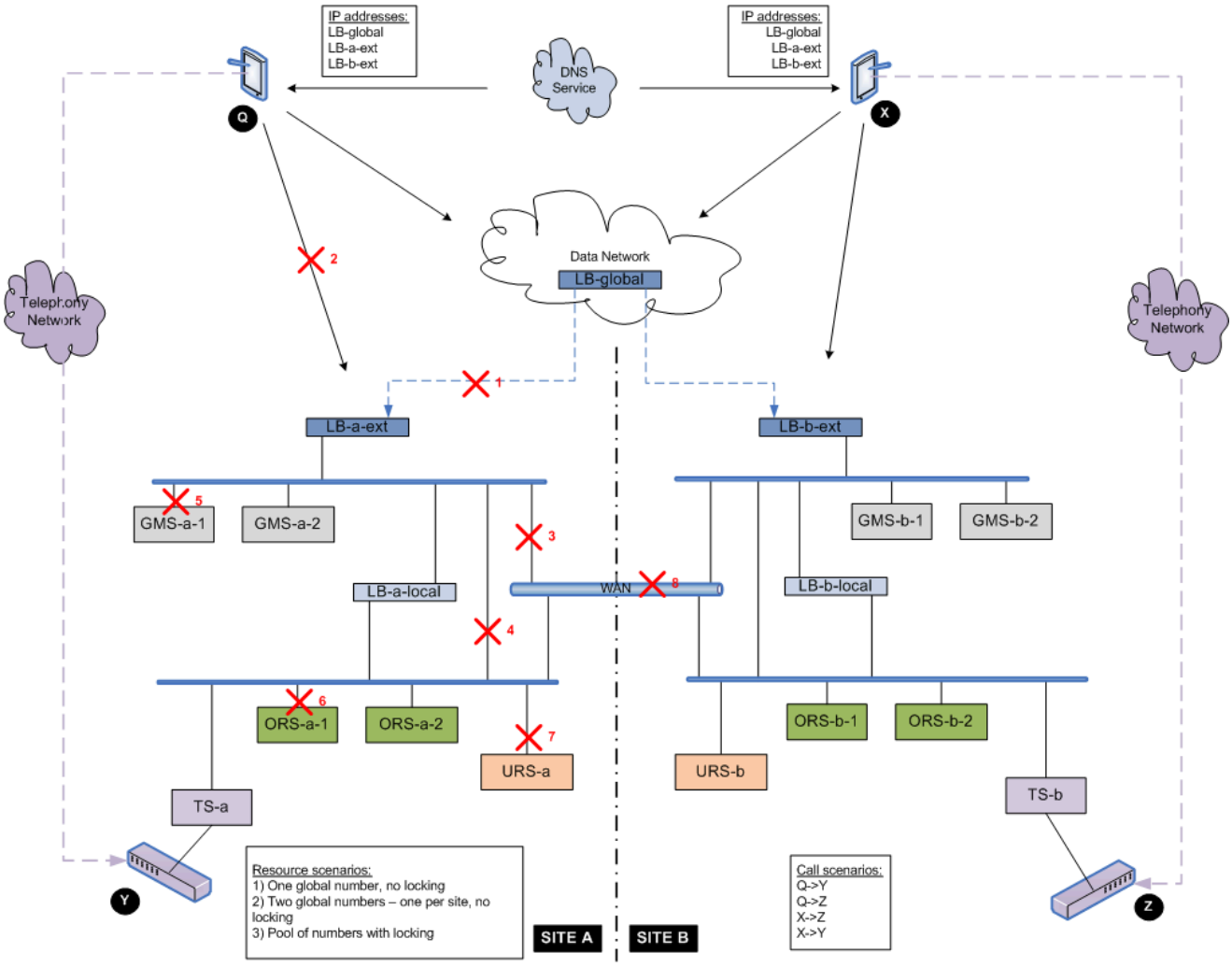
The general recommendation for user originated scenarios is to deploy an independent GMS cluster per site, and control call routing by ensuring that no intersecting pools of incoming phone numbers are configured. This way, if the client is requesting an access number through the data API call, it will be routed to Site A by the data network (internet), and then the call initiated by this client will land on the telephony switch/gateway located on (or associated with) Site A.

The matching API call will first be performed against the GMS cluster located on Site A. If Site A fails, then Site B is tried. This provides a good level of site isolation and simplifies maintenance, as well as day-to-day operations. Contact centers will be able to distribute telephony traffic between sites by controlling data API traffic (splitting between sites or directing to one site only) using standard IP load balancers, DNS records, and so on.

Using an approach of separate GMS clusters helps to avoid complex problems of "split brain" scenarios, where typically, some type of manual intervention is required either after the split, or later on when connection between sites is restored and clusters must be reconciled back together.

For user terminated scenarios, where call matching is not required, deploying a single GMS cluster across sites may provide some benefits. Still, it is not clear whether those benefits will outweigh the risks and be more beneficial as compared to the simplicity of a dedicated site cluster approach.

The following diagram shows major components of the solution and connectivity between them through local and wide area network segments. Possible network and/or component failures are numbered and described in more detail in the table below.



Failure Condition (see diagram)	Resource Locking Scenario	Two Separate GMS Clusters: One on Site A an	
Create Service/ Get Access Number	Match ixn Scenario		
	Same Site	Cross Site	
1 or 2	Global number, no locking.	Client has to retry the other site.	Not affected.
	Two global numbers – one per site, no locking.	Client has to retry the other site.	Not affected.
	Pool of numbers with locking.	Client has to retry the other site.	Not affected.

Failure Condition (see diagram)	Resource Locking Scenario	Two Separate GMS Clusters: One on Site A an	
3	Global number, no locking.	Not affected.	Not affected.
	Two global numbers – one per site, no locking.	Not affected.	Not affected.
	Pool of numbers with locking.	Not affected.	Not affected.
4	Global number, no locking.	Not affected.	Request fails. Retrying the other site is not p
	Two global numbers – one per site, no locking.	Not affected.	
	Pool of numbers with locking.	Not affected.	
8	Global number, no locking.	Not affected.	Not affected.
	Two global numbers – one per site, no locking.	Not affected.	Not affected.
	Pool of numbers with locking.	Not affected.	Not affected.

Installation

Overview

> **Important**
> Before you begin with the installation process, make sure that your environment meets the minimum requirements specified in the Prerequisites section.

Installing Genesys Mobile Service is a process that consists of the following tasks:

1. Creating a Genesys Mobile Services configuration object
2. Installing Genesys Mobile Services
3. Configuring ORS and Deploying DFM Files

Once the installation is complete, additional configuration is required before your Genesys Mobile Services deployment is ready to use:

- Basic Configuration
- Configuring Chat Support

Creating an Application Object

Before installing Genesys Mobile Services, use the templates included with your installation package to create an Application object in Configuration Manager and provide some basic configuration details. The following templates are provided:

- **ApplicationCluster_850.apd** - This template is for deploying all GMS's into the same cluster. The Cluster Application will contain shared configuration for GMS nodes.
- **GMS_850.apd** - This template is used to deploy GMS with default options. For a single node deployment, the GMS ApplicationCluster_850.apd template is not required. (See the Single Node Deployment section.)

Note: Configuration objects can also be created and configured in Genesys Administrator. Refer to the *Genesys Administrator Help* for information.

Starting Configuration Manager

Purpose: To start the Configuration Manager tool, which allows you to create the Genesys Mobile Services Application object and to configure Genesys Mobile Services options.

Start

1. Open Configuration Manager on your PC.
2. Enter the following information in the dialog box:
   - User name: Name of *Person* object defined in Configuration Manager.
   - User password: Password of *Person* object defined in Configuration Manager.
   - Application: Enter the name of the Configuration Manager Application object or default.
   - Host name: Name of the computer on which Configuration Server is running.
   - Port: Port number used by Configuration Server.
3. Click *OK* to start Configuration Manager.

End

Importing the Genesys Mobile Services Application Templates

Purpose: To import the GMS Application templates using Configuration Manager.

Start

1. In Configuration Manager, select *Environment* > *Application Templates*.
2. Right-click *Application Templates*.
3. From the shortcut menu that opens, select *Import Application Template*.
4. In the dialog box, navigate to the file for the Genesys Mobile Services Application Template. The following templates are included with your IP:
   - \Templates\ApplicationCluster_850.apd
   - \Templates\GMS_850.apd.
5. Select the ApplicationCluster_850.apd file and click *Open*.
6. In the Properties dialog box, click *OK*.
7. Repeat these steps to also import the GMS_850.apd template.

End

Next Steps:

- For a GMS cluster deployment, go to Cluster Deployment.
- For a single node deployment, go to Single Node Deployment.

Cluster Deployment

Creating and Configuring the GMS Application Cluster Object

Purpose: To create and configure a GMS Application Cluster object for Genesys Mobile Services.

Prerequisites: Import the Application Templates.

Start

1. In Configuration Manager, select *Environment* > *Applications*.
2. Right-click the *Applications* folder.
3. From the shortcut menu that opens, select *New* > *Folder*.
4. Select the *General* tab and, if desired, change the folder name (for example, ApplicationCluster_850). Do not use spaces in the folder name. Click *OK*.
5. Under Applications, right-click the folder that you just created.
6. From the shortcut menu that opens, select *New* > *Application*.
7. In the Open dialog box, locate the ApplicationCluster_850.apd template that you imported, and double-click it to open the Genesys Mobile Services Application object.
8. Select the *General* tab and, if desired, change the Application name (for example, ApplicationCluster_850). Do not use spaces in the Application name.
9. Make sure that the *State Enabled* check box is selected.

10. In a multi-tenant environment, select the *Tenants* tab and set up the list of tenants that use your Genesys Mobile Services application.
11. Click the *Server Info* tab and select the following:
    - *Host*—the name of the host on which Genesys Mobile Services resides.
    - *Port*—the port through which communication with Genesys Mobile Services can be established. After you select a Host, a default port is provided for your convenience. You can select the port and click *Edit Port* or you can configure a new port by clicking *Add Port*. Either action opens the New Port Info dialog box.
12. Select the *Start Info* tab and enter dummy values in *Working Directory* and *Command Line* fields. **Note:** The values entered at this time will be overwritten when Genesys Mobile Services is installed; however, having values in these fields is required to save the Application object.
13. Select the *Connections* tab and specify all of the servers to which Genesys Mobile Services must connect:
    - Orchestration Server (ORS) - optional
    - Statistics Server (Stat Server) - optional
    - Web API Server - optional
14. Click *Apply* to save your configured Application object. This creates the Application Cluster object. You must now add GMS nodes into your cluster by creating and configuring the GMS Application Object (Cluster).

End

Creating and Configuring the GMS Application Object (Cluster)

Purpose: To create and configure a GMS Application object for a GMS node that is part of a cluster.

Prerequisites:

- Import the Application Templates
- Create the GMS Application Cluster Object

Start

1. To create a GMS node in the cluster, right-click on your newly created Application Cluster object, and select *New > Application.*
2. In the Open dialog box, locate the GMS_850.apd template that you imported, and double-click it to open the Genesys Mobile Services Application object.
3. Select the *General* tab and change the Application name (for example, GMS_850_node1). Do not use spaces in the Application name.
4. Make sure that the *State Enabled* check box is selected.
5. In a multi-tenant environment, select the *Tenants* tab and set up the list of tenants that use your Genesys Mobile Services application.
6. Click the *Server Info* tab and select the following:
    - *Host*—the name of the host on which Genesys Mobile Services resides.

- *Port*—the port through which communication with Genesys Mobile Services can be established. After you select a Host, a default port is provided for your convenience. You can select the port and click *Edit Port* or you can configure a new port by clicking *Add Port*. Either action opens the New Port Info dialog box.

7. Select the *Start Info* tab and enter dummy values in *Working Directory* and *Command Line* fields. **Note:** The values entered at this time will be overwritten when Genesys Mobile Services is installed; however, having values in these fields is required to save the Application object.
8. Select the *Connections* tab. Click *Add*, and select the GMS Application Cluster application that you just created in the procedure Creating and Configuring the GMS Application Cluster Object.
9. Select the *Options* tab. The following default options will be listed.
   - server
   - resources
   - push
   - patterns
   - Log
   - chat
10. To share these options across the GMS cluster, select all of the options, right-click, and then select *Copy*. **Note:** Copying the *server* option into the GMS Application Cluster is recommended for the GMS cluster in order to use a load-balancer mechanism.
11. Click *Apply* to save your configured application object (GMS node). You must now copy the options into the GMS Application Cluster application, and configure the *server* option.
12. Open the GMS Application Cluster application.
13. Paste the copied options into the *Options* tab, and then click *Apply* to save.
14. Click the *server* option, and select *external_url_base*. Change the value of this option to a load balancer (frontend of the cluster nodes). Note that this step is optional if the GMS cluster uses a load-balancer, otherwise each GMS application has its own *server* option. Click *Apply*.
15. Open the GMS node application, and remove (delete) the options that you copied into the GMS Application Cluster application. When GMS is installed and running, it will check the *Connections* for an application cluster and will read the options from the cluster.
16. Select the *Security* tab. You must add a user that is allowed to read/write data into GMS related configuration objects (for example, GMS Application, Transaction Lists for Resources/Patterns, and so on). **Important:** All applications in the cluster must follow these steps:
    - Create a specific user such as *GMS Admin* (not an agent) and add this user as a *Member of Administrator Group*. Alternatively, you can simply use the *Default* user, which is already part of the Administrator Group.
    - Click *Permissions* and then select *Environment\Administrators.*
    - In the *Log On As* group, select *This Account* and set *GMS Admin* for the account.
17. Select the *Annex* tab of the person who will be logging into the GMS Service Management UI. You must set a role-based security option to grant users access to the GMS Service Management UI.
    - Create a new `gms` section. This new `gms` section must go into the user that is allowed to log into the Service Management UI. **Note**: If an Administrator changes a user's role during a Service Management UI session, the user will have to disconnect/reconnect for the new role to go into effect.

- Add Option Name: `roles`
- Add Option Value:
  - `Supervisor` - role used to monitor/configure the Callback Management feature only.
  - `Administrator` - role used to administer GMS. This role provides access to all panels and includes the Supervisor role.

18. Click *Apply* to save your configured Application object. This GMS node is now connected with the GMS Application Cluster. Repeat this procedure for each GMS node that will be in the GMS Cluster (typically for a production environment, at least three nodes should be part of the GMS Cluster).

End

Next Steps: Install Genesys Mobile Services.

Single Node Deployment

Creating and Configuring the GMS Application Object (Single Node)

Purpose: To create and configure a GMS Application object for a single GMS node (no cluster).

Prerequisites:

- Import the Application Templates

Start

1. In Configuration Manager, select *Environment* > *Applications*.
2. Right-click either the *Applications* folder or the subfolder in which you want to create your Application object.
3. From the shortcut menu that opens, select *New* > *Application*.
4. In the Open dialog box, locate the GMS_850.apd template that you imported, and double-click it to open the Genesys Mobile Services Application object.
5. Select the *General* tab and change the Application name (for example, GMS_850). Do not uses spaces in the Application name.
6. Make sure that the *State Enabled* check box is selected.
7. In a multi-tenant environment, select the *Tenants* tab and set up the list of tenants that use your Genesys Mobile Services application.
8. Click the *Server Info* tab and select the following:
   - *Host*—the name of the host on which Genesys Mobile Services resides.
   - *Port*—the port through which communication with Genesys Mobile Services can be established. After you select a Host, a default port is provided for your convenience. You can select the port and click *Edit Port* or you can configure a new port by clicking *Add Port*. Either action opens the New Port Info dialog box.
9. Select the *Start Info* tab and enter dummy values in *Working Directory* and *Command Line* fields.

Note: The values entered at this time will be overwritten when Genesys Mobile Services is installed; however, having values in these fields is required to save the Application object.

10. Select the *Connections* tab and specify all of the servers to which Genesys Mobile Services must connect:
    - Orchestration Server (ORS) - optional
    - Statistics Server (Stat Server) - optional
    - Web API Server - optional
11. Select the *Security* tab. You must add a user that is allowed to read/write data into GMS related configuration objects (for example, GMS Application, Transaction Lists for Resources/Patterns, and so on). To do this:
    - Create a specific user such as *GMS Admin* (not an agent) and add this user as a *Member of Administrator Group*. Alternatively, you can simply use the *Default* user, which is already part of the Administrator Group.
12. Select the *Annex* tab of the person who will be logging into the GMS Service Management UI. You must set a role-based security option to grant users access to the GMS Service Management UI.
    - Create a new `gms` section. This new `gms` section must go into the user that is allowed to log into the Service Management UI. **Note**: If an Administrator changes a user's role during a Service Management UI session, the user will have to disconnect/reconnect for the new role to go into effect.
    - Add Option Name: `roles`
    - Add Option Value:
      - `Supervisor` - role used to monitor/configure the Callback Management feature only.
      - `Administrator` - role used to administer GMS. This role provides access to all panels and includes the Supervisor role.
    - Click *Permissions* and then select *Environment\Administrators.*
    - In the *Log On As* group, select *This Account* and set *GMS Admin* for the account.
13. Click *Apply* to save your configured Application object.

End

Next Steps: Install Genesys Mobile Services.

Installing Genesys Mobile Services

With basic Configuration Server details in place, you are ready to complete the installation process.

Note: Genesys does not recommend installation of its components via a Microsoft Remote Desktop connection. The installation should be performed locally.

Install Genesys Mobile Services

Purpose: To install Genesys Mobile Services in your environment.

Prerequisites: Create and configure a Genesys Mobile Services Application Object.

Start

1. In your installation package, locate and double-click the setup application for your platform as specified below. The Install Shield opens the welcome screen.
   - Linux: *install.sh*
   - Windows: *setup.exe*
2. Click *Next*. The *Connection Parameters to the Configuration Server* screen appears.
3. Under *Host*, specify the host name and port number where Configuration Server is running. (This is the main "listening" port entered in the *Server Info* tab for Configuration Server, which is also used for authentication in the Configuration Manager login dialog box.)
4. Select *Use Client Side port*, if needed. Click *Next*.
5. Select the Genesys Mobile Services Application that you are installing. The Application Properties area shows the Type, Host, Working Directory, Command Line executable, and Command Line Arguments information previously entered in the *Server Info* and *Start Info* tabs of the selected Application object.
6. Click *Next*. The *Choose Destination Location* screen appears.
7. Under *Destination Folder*, keep the default destination or browse for the desired installation location.
8. Click *Next*. The *Server Configuration Parameters* screen appears.
9. Specify whether this instance of Genesys Mobile Services is going to be a seed node server (a primary server within a GMS cluster). In either case, you also need to specify the amount of RAM dedicated to maintaining the Apache Cassandra database that Genesys Mobile Services uses for its operations (Genesys recommends allocating 2Gb of RAM for jvm). Also, if you make this instance a backup server, you must specify the IP Address of the primary Genesys Mobile Services server before continuing.
10. Click *Next*. The *External Cassandra Instance Configuration* screen appears.
    - If you are using an External Cassandra instance, select the check box. Enter the IP Address(es) for the Cassandra nodes, and enter the Port Number for the Cassandra server.
    - If you are using the embedded Cassandra instance that is packaged with GMS, do not select the check box.
11. Click *Next*. The *Ready to Install* screen appears.

12. Click *Install*. The Genesys Installation Wizard indicates it is performing the requested operation. When through, the *Installation Complete* screen appears.
13. Click *Finish* to complete your installation of Genesys Mobile Services.
14. Repeat this procedure to install GMS on other hosts.

End

Next Steps: Configure ORS and Deploy DFM files.

ORS Configuration

This page details the ORS configuration steps required before you can use your Genesys Mobile Services installation.

Setting ORS Option

Start

1. Start Configuration Manager.
2. In Configuration Manager, select *Environment* > *Applications*.
3. Locate and open the Application object for your Orchestration Server. Note: This should be the same Application object you created a connection to when configuring your Genesys Mobile Services Application object.
4. Select the *Options* tab.
5. Open the *orchestration* section.
6. Set the value of the option *parse-start-params* to *false.*
7. Set the value of the option *mcr-pull-by-this-node* to *true.*
8. Click *OK* to save your changes.

End

Deploying DFM Files

Included with your installation are special configuration files, called DFM, which are required for Orchestration Server-based services. These files define Genesys Mobile Services-specific SCXML constructs that are required for the execution of SCXML applications used within Orchestration Server-based Services. For the Orchestration Server-based Services to function correctly, the following DFM files need to be configured in your Orchestration Server Application object:

- Storage
- Notification
- Callback
- Genesys Mobile-Based Services

The latest DFM definition files are included with the GMS installation and are available for download through the Service Management User Interface. Details about deploying these DFM in your

environment are provided below.

After deploying these DFM, you can use either an actual device with the demo application or an HTTP client (such as RestClient) to send API requests to Orchestration Server-based services. Refer to the Genesys Mobile Services API Reference for syntax of the requests.

> **Important**
>
> You must restart Orchestration Server after deploying DFM files for the changes to take effect.

Start

1. Download the DFM files from the Service Management User Interface.
2. Copy the DFM files onto your local file system where Orchestration Server (ORS) is running.
3. Start Configuration Manager.
4. In Configuration Manager, select *Environment* > *Applications*.
5. Locate and open the Application object for your ORS. Note: This should be the same Application object you created a connection to when configuring your Genesys Mobile Services Application object.
6. Select the *Options* tab.
7. Click *Add* to create the *dfm* section.
8. In the *dfm* section, create and configure one option for each DFM, using the option value to specify the file path. Details are provided in the table below.
9. Click *OK* to save your changes.
10. Restart Orchestration Server (ORS). ORS reads the DFM configuration on startup.

End

List of DFM Options for Orchestration Server

Service	Option Name	Option Value
Storage	gsgStorage	c:\dfms\storage.txt
Notification	gsgNotification	c:\dfms\notification.txt
Genesys Mobile-Based Services	gsgBasedServices	c:\dfms\basedservices.txt
Callback	gsgCallback	c:\dfsm\callback.txt

Next Steps

Although the GMS installation is now complete, there are additional steps required before using Genesys Mobile Services.

- Basic Configuration
- Configuring Chat Support

Basic Configuration

This page details the basic configuration steps required before you can use your Genesys Mobile Services installation. For a more general look at the configuration options available, refer to the Configuration Options Reference.

Basic Configuration Overview

Genesys Mobile Services provides a set of services or APIs that require configuration before the product can be used. The configuration of services is stored into Configuration Server.

Working in Configuration Manager

Genesys Mobile Services is represented by an Application object in the Configuration Server database. This Application object is based on the "Genesys Generic Server" template and contains typical settings for a Genesys application including Server Info, Start Info, and Connections to other servers. It also includes Options that correspond to configuration details for sub-services of Genesys Mobile Services.

Note: By design, settings in Configuration Manager for any configured service override the matching request parameters. This means that if *_provide_code* is set to *true* then this service will always respond with an access code - even if the *_provide_code* parameter received with the request is set to *false*.

Configuration settings are grouped for different service types, and stored in Option sections described below:

- log Section—Standard log file options for this Application object. For more information about these options, refer to your Genesys Framework documentation.
- gms Section—Configuration settings used across different services.
- push Section—Notification service parameters. Not monitored at run-time, so the Genesys Mobile Server instance must be restarted for changes to take effect.
- resource Section—Details about how resource groups are handled. Run-time configuration changes are supported, so changes take effect immediately.
- server Section—Cluster sub-service configuration details. Includes URL representation of this node of the cluster, consisting host, port and application name formatted in the following way: http://*web_host*:*web_port*/*app_name*. (Example: http://yourHostName:8080/gms). Run-time configuration changes are supported, but due to tight logical connection to the web-container configuration, a restart is needed in most cases.
- service.servicename Section—Additional configuration options for customized services.

Some services also rely on configuration details from a Transaction object in Configuration Manager that must be created and configured in your Genesys environment. **Note:** If setting up multiple Genesys Mobile Services nodes, the configuration options specified in the Application object must be the same for each instance. If you are familiar with working in a Genesys environment, this type of

configuration should be second nature. If you require additional information about how to work with Configuration Manager to edit these configuration options, refer to the Help file included with that product.

Creating and Configuring a Resource List

Some services included with Genesys Mobile Services require a list of resources, such as a list of access numbers that can be managed. Such lists are held in a Transactions object, which is then referenced by Options set in the Genesys Mobile Services Application object.

The steps required to create and populate a resource list are provided below. You can also configure these services through the GMS Service Management User Interface.

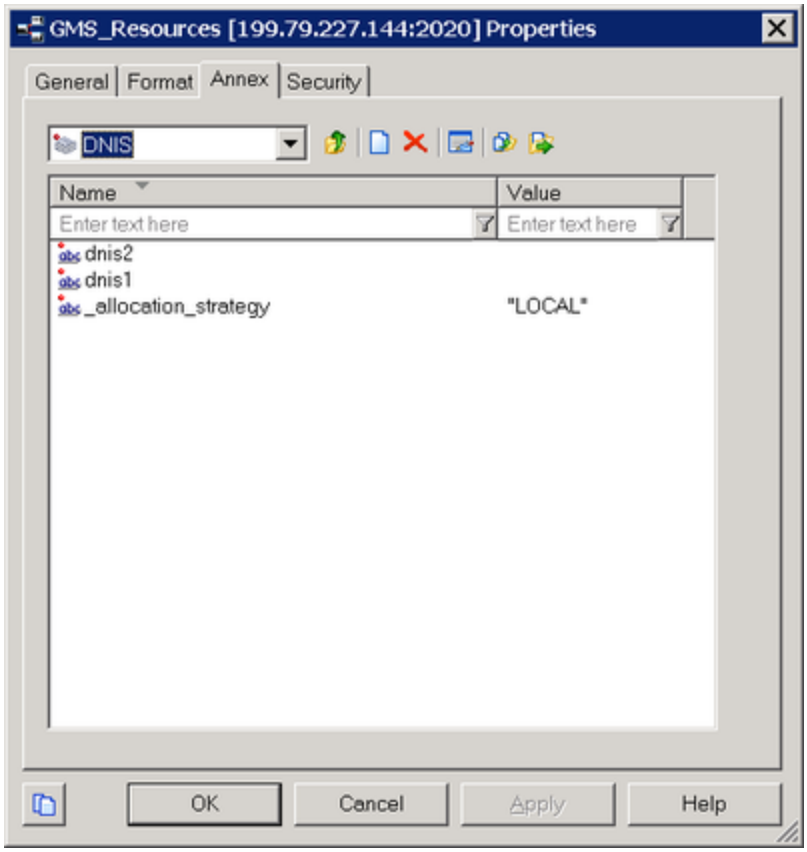
Note: Be sure that you have the *Show Annex tab in object properties* option selected from the *View > Options* menu before starting this procedure.

Start

1. Start Configuration Manager.
2. Under the tenant you are working with, open the Transactions folder.
3. Right-click and select *New > Transaction.*
4. On the *General* tab, configure the following fields:
   - Name—This name must match the *resources > resource_list_name* option value from your Genesys Mobile Services Application object. The default value is *GMS_Resources*.
   - Type—Select *List* from the drop down box.
   - Alias—Enter an alias of your choice.
5. On the *Annex* tab, create a new section. The section name used here must match the value of the *resource_group* option, located in the *service.servicename* section of your Genesys Mobile Services Application object.
6. Add options to the newly created section to create your resource list. The Resource name should not start with an underscore (_).
7. Add and set an *_allocation_strategy* option for this group. The valid values are *RANDOM, LOCAL,* or *CLUSTER.*

End

A sample resource list configuration is shown below.



Creating and Configuring a Pattern List

Some services included with Genesys Mobile Services require a list of patterns (for exceptions, and so on) to compare parameter values to a list of defined patterns. These lists are held in a Transactions object, which is then referenced by Options set in the Genesys Mobile Services Application object.

The steps required to create and populate a pattern list are provided below. You can also configure these services through the GMS Service Management User Interface.

Start

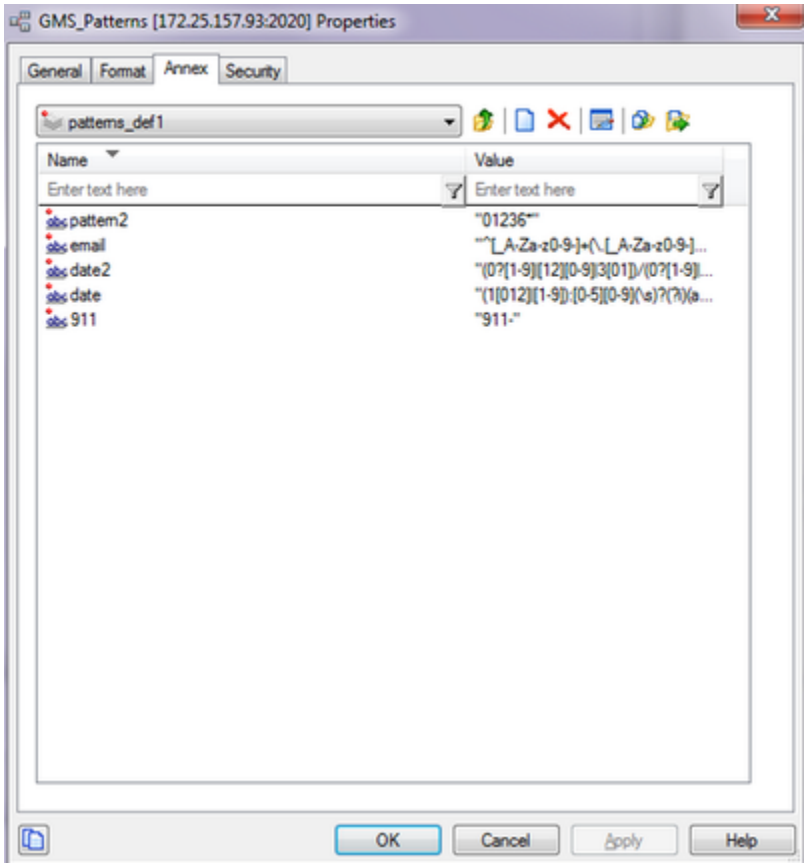
1. Start Configuration Manager.
2. Under the tenant you are working with, open the Transactions folder.
3. Right-click and select *New* > *Transaction.*
4. On the *General* tab, configure the following fields:
   - Name—This name must match the *patterns* > *pattern_list_name* option value from your Genesys Mobile Services Application object. The default value is *GMS_Patterns.*
   - Type—Select *List* from the drop down box.

- Alias—Enter an alias of your choice.

5. On the *Annex* tab, create a new section. The section name used here must match the value of the *patterns_group* option, located in the *service.servicename* section of your Genesys Mobile Services Application object.
6. Add options to the newly created section to create your pattern list.

End

A sample pattern list configuration is shown below.



Configuring Services for Genesys Mobile Services

To complete your deployment, the following GMS-based services need to be configured:

1. request-interaction
2. match-interaction
3. request-access

Required options are outlined below, with some sample values to help you get started. You can

configure these services through the GMS Service Management User Interface.

request-interaction

Required request-interaction Options

Option Name	Option Value
_type	builtin
_ttl	30
_service	request-interaction
_resource_group	(Use the section name created earlier under your GMS_Resources Transactions object.)
_provide_code	true

match-interaction

Required match-interaction Options

Option Value	
_type	builtin
_service	match-interaction
_delete_match	true
_http_status_on_error	404 (or a valid http status code)

request-access

Required request-access Options

Option Name	Option Value
_type	builtin
_ttl	30
_service	request-access
_resource_group	(Use the section name created earlier under your GMS_Resources Transactions object.)
_access_code_length	4

Additional Configuration

> **Important**

All nodes in your deployment (GMS, ORS, and so on) must be set with the same time.

Single Tenant Support

Note: Introduced in GMS 8.5.003.

For chat service, the default chat endpoint value in the chat section > `default_chat_endpoint` option can be used for all services. Or by service, you can customize the `_chat_endpoint` option. These options, `default_chat_endpoint` and `_chat_endpoint`, are composed of `<tenant DBID>:<chat endpoint name>`. The configuration for the tenant DBID is:

- Single-tenant Configuration Server, using the tenant name Resources, with DBID=101
- Multi-tenant Configuration Server, using the default tenant, tenant name is Environment, with DBID=1
- Multi-tenant Configuration Server, using a tenant other than Environment, with DBID distinct from 1

Next Steps

- Configuring Chat Support

Configuring Chat Support

This page details the specific configuration steps required to use the Chat API included with Genesys Mobile Services. For more details about this API, refer to Chat API.

Configuration Overview

Prerequisite: Before beginning the steps described here, you should have completed the basic configuration process. To use the Chat API with your Genesys Mobile Services deployment, you must specify configuration details in the Application objects for the following objects:

- Genesys Mobile Services
- Web API Server
- Chat Server

Note: For Genesys Mobile Services configuration, it is assumed that you already have Web API Server and Chat Server installed and configured. Refer to documentation for those products if you require additional details. The following sections provide details about configuration changes required to use chat with your Genesys Mobile Services deployment. Procedures and illustrations on this page use Genesys Administrator, although the configuration can also take place using Configuration Manager.

Genesys Mobile Services Configuration

The following configuration options must be specified in your Genesys Mobile Services Application object:

Start

1. Open Genesys Administrator in a web browser.
2. Locate and view the Genesys Mobile Services Application object you previously created and configured.
3. Under the *General* section of the *Configuration* tab, add a connection to the Web Server API Application object that will be used with your Genesys Mobile Services deployment.
4. Under the *Options* tab, in the *Chat* section, include the mandatory configuration options described in the table below.

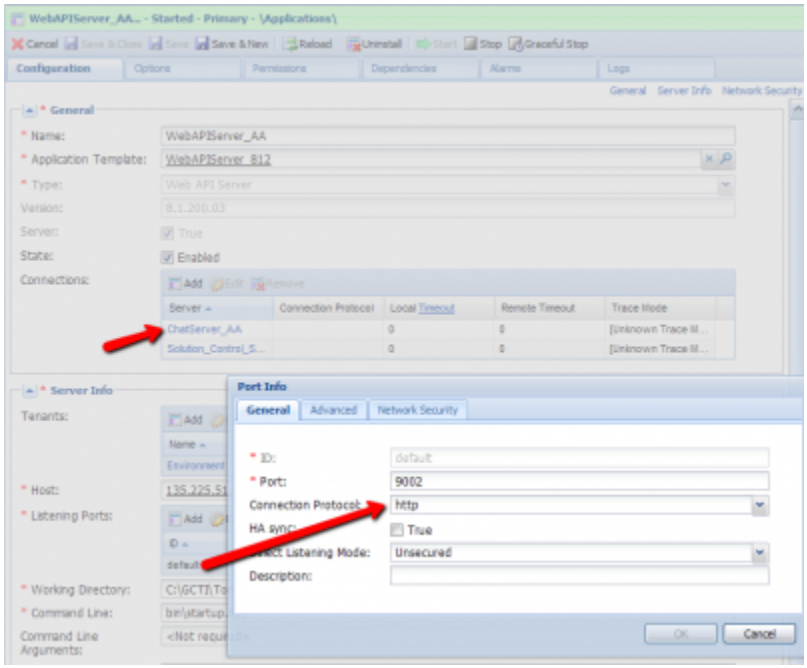
End

Genesys Mobile Services Options

Section: chat			
Option Name	**Required**	**Option Value**	**Description**
chat_load_balancer_url_path	true	WebAPI812/ SimpleSamples812/ ChatHA/ ChatLBServerInfo.jsp	URL to the load balancer (WebAPI) for Chat servers
chat_session_request_timeout	true	30000	Duration after which the chat interaction gets deleted
default_chat_endpoint	false	<tenant_name:chat_endpoint>	This option is used for all chat services in order to define the queue (URS) where the chat session initiated by GMS will enter. The value of this option is the tenant name on which the service(s) will proceed, and the chat endpoint as defined in the `ChatServer` option. For example, the section endpoints for the tenant `Environment` in the chat options is written as `endpoints:1`. This section contains the endpoint options (for example, default=queue). The chat endpoint value to use this `default` endpoint in the `Environment` tenant is `Environment:default`. Note: You can supersede this option for each chat service using the `_chat_endpoint` option with the same <tenant_name:chat_endpoint> value. The default value for this option is `Environment:default`.
_client_timeout	false		If the client does not interact with the Chat service (refresh, send message, send event), GMS stops to poll the Chat server, and the Chat session is closed. This option applies only to chat sessions

Section: chat			
			implemented using Cometd connections. For non-Cometd implementation, Chat server timeout parameters apply. The default value for this option is 15 minutes.

Web API Server Configuration



To configure the Web API Server, at least one Chat Server must be added and configured as an active connection. There can be multiple "primary" chat servers added as connections, in which case the Web API Server will balance between them. However, each chat server should have a warm standby backup server configured for reliability.

Use the following procedure to download a required chat-related file, and to update the Web API Server Application object that is being used by your GMS deployment:

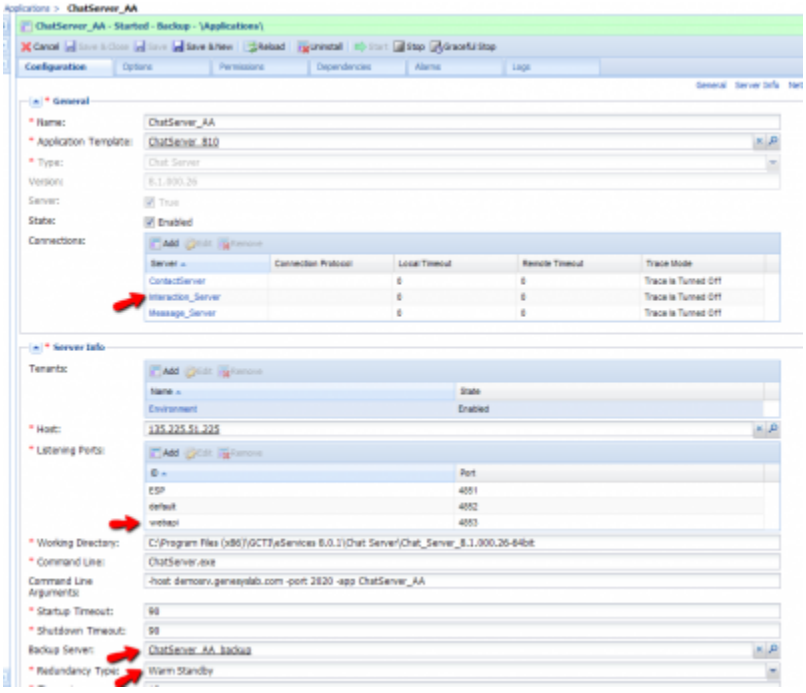
Start

1. Download the `ChatLBServerInfo.jsp` file, and then add the file into the Web API Server directory, in the `webapps` folder.
   - Download ChatLBServerInfo.jsp for Single Tenant
   - Download ChatLBServerInfo.jsp for Multi-Tenant

2. Open Genesys Administrator in a web browser.
3. Locate and view the Web Server API Application object associated with your Genesys Mobile Services deployment.
4. View the *Configuration* tab.
5. In the *General* section, find the *Connections* table and click *Add*.
6. Locate and select the Chat Server Application object that you want to use.
7. Click on the Chat Server connection you plan to use to edit Port Info.
8. Ensure the *Connection Protocol* associated with the Chat Server is *http*.
9. Repeat this procedure to add additional Chat Sever instances, as necessary.

End

Chat Server Configuration



The Chat Server Application object being used by your Genesys Mobile Services deployment should have the following configuration updates:

- Add a connection to Interaction Server.
- Listen for Web API Server traffic on the appropriate port.
- Set a backup server and specify the redundancy type.

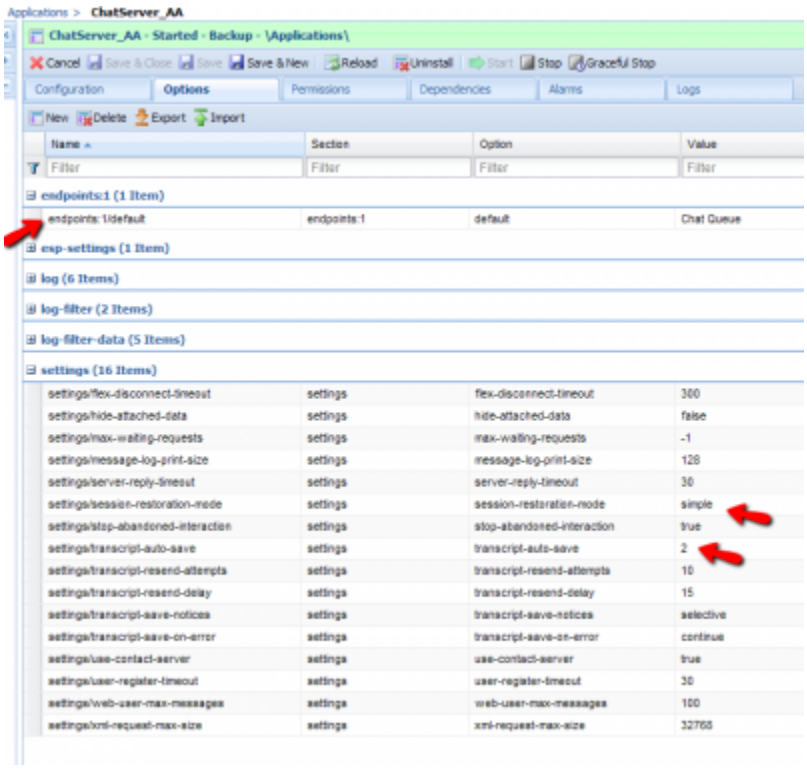
The detailed steps are provided below:

Start

1. Open Genesys Administrator in a web browser.
2. Locate and view the Chat Server Application object associated with your Genesys Mobile Services deployment.
3. View the *Configuration* tab.
4. In the *General* section, find the *Connections* table and click *Add*.
5. Locate and select the Interaction Server Application object that you want to use.
6. In the *Server Info* section, find the *Listening Ports* table and click *Add*.
7. Add the port being used by the Web API Server that you configured previously to work with this Chat Server Application object.
8. Repeat this procedure for each Chat Server associated with your Genesys Mobile Services deployment.

End

Setting Chat Server HA-Specific Options



Sample Chat Server Configuration

The following procedure should be followed to enable high availability (Requires Chat Server **8.1.000.20 or higher**):

Start

1. Open Genesys Administrator in a web browser.
2. Locate and view the Chat Server Application object associated with your Genesys Mobile Services deployment.
3. View the *Server Info* section on the *Configuration* tab.
4. Specify a *Backup Server* value.
5. Set the *Redundancy Type* to *Warm Standby*.
6. Under the *Options* tab, include the mandatory configuration options described in the table below.
7. Repeat this procedure for each (primary) Chat Server associated with your Genesys Mobile Services deployment.

End

Required Chat Server Options (HA)

Section: endpoints:1	
Option Name	**Option Value**
default	Chat In
Section: settings	
Option Name	**Option Value**
session-restoration-mode	simple
transcript-auto-save	2

Next Steps

With basic configurations now complete, you can start loading and managing your services, using the GMS Service Management User Interface.

- Service Management User Interface

You can also configure additional, advanced settings that are outlined in the following section:

- Configuration

Configuration

The following table lists additional configurations that can be used with your GMS installation.

Page	Summary
Configuring an External Cassandra	Provides configuration steps in order to use external Cassandra instances in your GMS deployment.
Load Balancer Configuration Examples	Recommendations for load balancing.
Configuring a GMS Service for ORS Load Balancing	Describes how to set up a GMS Service for ORS Load Balancing.
Configuring and Starting a GMS Cluster	Describes the process for initializing a GMS cluster.
ORS Cookie Support	Provides information about the cookie-based management that GMS supports.
Mobile Push Notifications	Details configuration for push notification service.
Custom Reporting	Basic configuration for Real-Time and Historical Reporting based on T-Server's UserEvent mechanism.
Impementing ADDP	Discusses the Advanced Disconnect Detection Protocol (ADDP) protocol.
Impementing IPv6	Provides information about Internet Protocol version 6 (IPv6).
Starting and Stopping GMS	Instructions to start and stop GMS, and configure a delay for a graceful shutdown.
GMS Alarms	Describes alarms that can be raised by GMS based on system status/configuration.

Configuring an External Cassandra

Genesys Mobile Services (GMS) is packaged with an embedded Cassandra; however, GMS also supports deployments with an external Cassandra(s). An external Cassandra might be used in the following scenarios:

- You already have a Cassandra/Datastax deployment.
- You are securing your data in segregated networks, cages, and racks.
- You want multiple redundancy features, such as distinct data centers, rack, and chassis awareness.

Configuring GMS for an external Cassandra is a multi-step process to enable connection, authentication, and authorization. The steps include setting configuration options in Configuration Manager (or Genesys Administrator), changing configuration settings in the `cassandra.yaml` file, and executing Cassandra Query Language (CQL) commands.

> **Important**
> All external Cassandra nodes must be of the same version.

Configuration Options

The following tables list the configuration options applicable to an external Cassandra deployment. Changes take effect after restart.

cassandra Section

Option	Default Value	Description
nodes		List of Cassandra hosts or IP addresses, comma separated. For example: `host1,192.168.1.2`
port	0	The listening port of the Cassandra server (that is, the port on which Thrift listens for clients).
create-embedded-server	true	Set this option to **`false`** to connect to an external Cassandra instance. If set to `true`, creates and connects to the Cassandra server embedded with GMS and ignores all other options in the cassandra section.

Option	Default Value	Description
create-schema	true	If set to `true`, creates (if needed) keyspaces and column families for GMS. If set to `false`, does not create keyspaces.
keyspace-prefix		Specifies the prefix for GMS keyspace naming. The default value is empty for backward compatibility. Note: If this value is left empty, the `gsg` and `gsg_dd` keyspaces will be created in Cassandra.
strategy-class	SimpleStrategy	Specifies the strategy class that Cassandra uses for the cluster. Valid values are: `SimpleStrategy`, which defines a single cluster without multiple Data Centers. `NetworkTopologyStrategy`, which is a network strategy in conjunction with the `cassandra-topology` properties file (located in the install configuration directory for each Cassandra instance), defines the Data Centers for the Cassandra cluster. Multiple Data Centers are typically geographically dispersed.
strategy-option	replication_factor:4	Specifies the replication factor value according to the strategy-class: If the strategy class is `SimpleStrategy`, set this value to `replication_factor:2`, where 2 is the number of Cassandra nodes. If the strategy class is `NetworkTopologyStrategy`, set this value to `DC1:2;DC2:3`, where `DC` is the Data Center topology.

cassandra-authentication-security Section

Note: The user name and password is replicated to all nodes.

Option	Description
username	the Cassandra user name
password	the Cassandra user password

Connection to an External Cassandra

The following steps are required to enable GMS to connect to an external Cassandra.

1. In Configuration Manager, locate and open your GMS Application object.
2. On the Options tab, `cassandra` section, set the following options:
   - nodes = <your Cassandra hosts or IP addresses>
   - port = <your Cassandra port>
   - create_embedded_server = `false`
   - strategy-class = `SimpleStrategy` or `NetworkTopologyStrategy`
   - strategy-option = `replication_factor:2` or `DC1:2;DC2:3`
3. Restart GMS.

Authentication on External Cassandra

Note: Supports Cassandra version 2.0.x and higher.

The following steps are prerequisites prior to enabling authentication.

Configure cassandra.yaml File

1. On the external Cassandra, open the `cassandra.yaml` configuration file.
2. Make sure that `cluster_name` is the same for all nodes.
3. Locate the seed nodes. This is the field for all Cassandra nodes; change it accordingly (for the seed node, this will be it's own port, for the non-seed nodes, this will be the IP address of the seed node).
4. Make sure that `listen_address` is changed from `127.0.0.1` to the current IP address.
5. Make sure that `rpc_address` is changed from `127.0.0.1` to the current IP address.

The following steps are required to enable authentication.

Configure cassandra.yaml File

1. On the external Cassandra, open the `cassandra.yaml` configuration file.
2. Locate the `authenticator` field.
3. Change the value from `AllowAllAuthenticator` to `PasswordAuthenticator`. Note: The full classname is `org.apache.cassandra.auth.PasswordAuthenticator`.
4. Save the file.
5. Repeat these steps on each external Cassandra instance.

Execute CQL Commands

1. On the external Cassandra, using the cqlsh utility (included with Cassandra), create your username and password. **Note:** The default superuser is `cassandra` with pasword `cassandra`. The following example shows a `genesys` user with `genesys` password.

Note: This step is required to be completed on only one external Cassandra instance. It will then be replicated to the other nodes.

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
> CREATE USER genesys WITH PASSWORD 'genesys';
> LIST USERS;
name | super
-----------+-------
 genesys | False
 cassandra | True
```

On Windows OS / Cassandra versions 2.1 or higher, replace:

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
```

with:

```
{path_to_cassandra}\bin>{path_to_python}\python.exe cqlsh cassandra_host -u cassandra
-p cassandra
```

Set the options of `cqlsh` before parameters or set your python 2.7 path in PATH environment variable like this:

```
PATH={path_to_python};%PATH%
```

Therefore, you can launch the `cqlsh` script using the `cqlsh.bat` command:

```
cqlsh.bat -u cassandra –p cassandra cassandra_host
```

Using the default cassandra port of `native_transport_port` (default is `9042`). Otherwise you will need to add the port parameter to the cqlsh script.

Set Configuration Options

1. In Configuration Manager, locate and open your GMS Application object.
2. On the Options tab, `cassandra-authentication-security` section, set the following options with the same username and password that you just created on the external Cassandra.
   - username, for example, `genesys`
   - password, for example, `genesys`
3. Restart GMS. The Pelops and Hector clients connect to the external Cassandra using the login and password.

Authorization on External Cassandra

Note: Supports Cassandra version 2.0.x and higher.

After creating the authentication, you must enable authorization and create keyspaces.

Configure cassandra.yaml File

1. On the external Cassandra, open the `cassandra.yaml` configuration file.
2. Locate the `authorizer` field.
3. Change the value from `AllowAllAuthorizer` to `CassandraAuthorizer`. Note: The full classname is `org.apache.cassandra.auth.CassandraAuthorizer`.
4. Save the file.
5. Repeat these steps on each external Cassandra instance.

Execute CQL Commands

To authorize actions on the keyspace, you must first create the keyspace(s), and then grant permissions on them.

1. On the external Cassandra, using the cqlsh utility (included with Cassandra), create your keyspaces. The following example shows the `gsg` and `gsg_dd` keyspace.

Note: This step is required to be completed on only one external Cassandra instance. It will then be replicated to the other nodes.

```
$ cqlsh -u cassandra -p cassandra cassandra_host cassandra_port
> LIST USERS;
name | super
-----------+-------
 genesys | False
 cassandra | True
> LIST ALL PERMISSIONS OF genesys;
 (0 rows)

> CREATE KEYSPACE gsg WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' :
3 };
> CREATE KEYSPACE gsg_dd WITH REPLICATION = { 'class' : 'SimpleStrategy',
'replication_factor' : 3 };
> GRANT ALTER ON KEYSPACE gsg TO genesys;
> GRANT CREATE ON KEYSPACE gsg TO genesys;
> GRANT DROP ON KEYSPACE gsg TO genesys;
> GRANT MODIFY ON KEYSPACE gsg TO genesys;
> GRANT SELECT ON KEYSPACE gsg TO genesys;
> LIST ALL PERMISSIONS OF genesys;
username | resource | permission
----------+----------------+------------
 genesys | <keyspace gsg> | CREATE
 genesys | <keyspace gsg> | ALTER
 genesys | <keyspace gsg> | DROP
 genesys | <keyspace gsg> | SELECT
 genesys | <keyspace gsg> | MODIFY
(5 rows)
```

```
> GRANT ALTER ON KEYSPACE gsg_dd TO genesys;
> GRANT CREATE ON KEYSPACE gsg_dd TO genesys;
> GRANT DROP ON KEYSPACE gsg_dd TO genesys;
> GRANT MODIFY ON KEYSPACE gsg_dd TO genesys;
> GRANT SELECT ON KEYSPACE gsg_dd TO genesys;
> LIST ALL PERMISSIONS OF genesys;
username | resource | permission
----------+------------------+------------
 genesys | <keyspace gsg> | CREATE
 genesys | <keyspace gsg> | ALTER
 genesys | <keyspace gsg> | DROP
 genesys | <keyspace gsg> | SELECT
 genesys | <keyspace gsg> | MODIFY
 genesys | <keyspace gsg_dd> | CREATE
 genesys | <keyspace gsg_dd> | ALTER
 genesys | <keyspace gsg_dd> | DROP
 genesys | <keyspace gsg_dd> | SELECT
 genesys | <keyspace gsg_dd> | MODIFY
(10 rows)
```

Note: You do not need to set any additional option in the GMS application.

2. Restart GMS. The Pelops and Hector clients connect to the external Cassandra and are authorized to manage the GMS keyspaces (`gsg` and `gsg_dd`).

Final Steps

- In the GMS Application > Security section > Log On As SYSTEM Account.
- The time zone for all nodes must be the same. Make sure that you synchronize the time before testing.

Configuring Apache Load Balancer

The following is an example of how to configure Apache load balancer that can be positioned in front of GMS nodes for API requests distribution. For configuration of other load balancing solutions, please refer to their documentation.

Configuration of mod_proxy_balancer

Install Apache 2.2 or higher, starting from this version mod_proxy is able to use the extension mod_proxy_balancer. Make sure the following modules are present in your Apache "modules\" folder, upload them in case they are absent:

- mod_proxy.so
- mod_proxy_balancer.so

Add the following strings to your Apache configuration "httpd.conf" file in order to load the modules:

- LoadModule proxy_module modules/mod_proxy.so
- LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
- LoadModule proxy_http_module modules/mod_proxy_http.so

Basically you need to add node based configuration, for this add the following to the httpd.conf file:

```
ProxyPass / balancer://my_cluster/ stickysession=JSESSIONID nofailover=On

<Proxy balancer://my_cluster>
   BalancerMember http://yourjetty1:8080 route=jetty1
   BalancerMember http://yourjetty2:8080 route=jetty2
</Proxy>
```

Proxy balancer:// - defines the nodes (workers) in the cluster. Each member may be a http:// or ajp:// URL or another balancer:// URL for cascaded load balancing configuration. If the worker name is not set for the Jetty servers, then session affinity (sticky sessions) will not work. The JSESSIONID cookie must have the format <sessionID>.<worker name>, in which worker name has the same value as the route specified in the BalancerMember above (in this case "jetty1" and "jetty2"). As an example: The following can be added to the jetty-web.xml file to set the worker name in case you need session affinity (sticky sessions):

```
<Configure class="org.mortbay.jetty.webapp.WebAppContext">
  <Get name="sessionHandler">
    <Get name="sessionManager">
      <Call name="setIdManager">
        <Arg>
          <New class="org.mortbay.jetty.servlet.HashSessionIdManager">
            <Set name="WorkerName">jetty1</Set>
          </New>
        </Arg>
      </Call>
```

```
    </Get>
  </Get>
</Configure>
```

Load Balancer Management Page

Apache provides balancer manager support so that the status of balancer can be viewed on a web page. The following configuration enables this UI at /balancer URL:

```
<Location /balancer>
SetHandler balancer-manager

 Order Deny,Allow
 Deny from all
 Allow from all
 </Location>
```

Configuring a GMS Service for ORS Load Balancing

Supported Load Balancing Features

Genesys Mobile Services (GMS) supports the following load balancing features:

- Ability to configure a list of service URLs to access a given list of nodes.
- Linear hunt strategy where requests are always delivered to the first available node in the list.
- Circular hunt strategy where requests are delivered in a round-robin fashion to the list of nodes/URLs.
- Ability to configure a linear hunt strategy or a circular hunt strategy in the service configuration. The default hunt mode is circular.

Configuration Options

If you configured a service for a GMS application, you can activate load balancing by adding an `_ors` option, which is used to set up a list of Orchestration Servers associated with the service.

> **Important**
> Your service must be of type `ors`.

For each service, you can also specify a load balancing strategy in the `_ors_lb_strategy` option. This ensures the possibility to define as many load balancing strategies as services with a distinct list of Orchestration Servers.

If you define an application cluster for your GMS applications, and if you define your load balancing properties for your service in this cluster application, these service options apply to all of the GMS nodes. If you modify these service options in one of the GMS nodes, the new options apply to this given node and cluster options are superseded in this node. This lets you define common service options for your cluster, with the possibility to fine-tune the service options in one or more GMS nodes.

The following table shows an example for the **section service.my-office-hours** service.

Option Name	Option Value
_name	The name of the service; for example, `my-office-hour`

Option Name	Option Value
_service	The service name; for example, `office-hours`.
_type	ors
_ors	`http://hostname1:port,http://hostname2:port,http://hostn`
_ors_lb_strategy	circular (default) or linear

In a circular ORS load balancing strategy, the `max_ors_request_attempts` option provides you with the ability to select the next ORS Server in the list of ORS Servers defined for the service when the request to the first ORS fails. For example, if you set the value of `max_ors_request_attempts` to 1, the first ORS Server in the list will be used only one time, and in case of ORS failure, the request will fail. If you set the value to 3, the first ORS Server in the the list will be used and if the request fails, the next ORS Server in the list will be used, and so on, until the third server. After the third server fails to respond, the request returns a failure.

A linear ORS load balancing strategy follows the same process with the `max_ors_request_attempts` option for retry on failure.

Note: The `max_ors_request_attempts` option is in the `ors` section; the default value is 3 (three attempts before failure); the value must be greater than `0`.

Setting up Load Balancing Configuration

Purpose
To set up ORS load balancing for a given GMS service.

Prerequisites

- You defined a service for your GMS application by using the Service Management User Interface, for example, a service of name `office-hours`.
- You installed and configured Orchestration Servers.

Start

- In the Service Management User Interface > Services > Configured Services > <your service name>:
  1. Click `Add Settings`.
  2. Enter `_ors` for the Name, and then enter the list of URLs, separated with commas, for the Value.
  3. (Optional) Click `Add Settings` again. Enter `_ors_lb_strategy` for the Name, and then enter `circular` or `linear` for the Value. If not specified, the default `_ors_lb_strategy` value is `circular`.
- In Configuration Manager:
  1. Open your application and select the Options tab. Edit your `service.<service_name>` section.
  2. Click `Create New Section/Option`.

3. Enter _ors for the Option Name, and then enter the list of URLs, separated with commas, for the Option Value. Click OK.
4. (Optional) Click `Create New Section/Option` again. Enter `_ors_lb_strategy` for the Name, and then enter `circular` or `linear` for the Value. If not specified, the default `_ors_lb_strategy` value is `circular`.

Stop

Configuring and Starting a GMS Cluster

Prerequisites

- GMS version 8.5.x
- Red Hat Linux version 5.0 (32 bit and 64 bit), 4Gb RAM or Higher (for the HA Proxy Load Balancer)
- JDK 1.6.30 or higher

Introduction

The process for initializing a GMS cluster (whether it is a single node, multiple nodes, or multiple data center cluster) is to first correctly configure the Node and Cluster Initialization Properties in each node's cassandra.yaml configuration file, and then start each node individually, starting with the seed node(s). Configuration file cassandra.yaml is automatically generated by GMS Installation Package, you don't need to update the file until you need specific settings. Installation Package proposes to choose between the type of node "seed node/Not a seed node". The following section explains how the GMS cluster is setup.

Initializing a Single-Node Cluster

GMS is intended to be run on multiple nodes, however you may want to start with a single node cluster for evaluation purposes. To start GMS on a single node:

1. Set the following required properties in the cassandra.yaml file:

```
cluster_name: GMS Cluster
initial_token:
```

(Optional) The following properties are already correctly configured for a single node instance of Cassandra. However, if you plan on expanding to more nodes after your single-node evaluation, setting these correctly the first time you start the node is recommended.

```
seeds: <IP of GMS node>
listen_address: <IP of GMS node>
rpc_address: <IP of GMS node>

Start GSG on the node.
```

Initializing a Multi-Node or Multi-Data Center Cluster

To correctly configure a multi-node or multi-datacenter cluster you must determine the following information:

- A name for your cluster
- How many total nodes your cluster will have, and how many nodes per data center (or replication group)
- The IP addresses of each node
- The token for each node (see Calculating Tokens). If you are deploying a multi-datacenter cluster, make sure to assign tokens so that data is evenly distributed within each data center or replication grouping (see Calculating Tokens for Multiple Data Centers).
- Which nodes will serve as the seed nodes. If you are deploying a multi-datacenter cluster, the seed list should include a node from *each* data center or replication group.

This information will be used to configure the Node and Cluster Initialization Properties in the cassandra.yaml configuration file on each node in the cluster. Each node should be correctly configured before starting up the cluster, one node at a time (starting with the seed nodes). For example, suppose you are configuring a 4 nodes cluster spanning 1 rack in a single data center. The nodes have the following IPs, and one node in the rack will serve as a seed:

- GMS node 172.25.157.171 (seed)
- GMS node1 172.25.157.177
- GMS node2 172.25.157.179
- GMS node3 172.25.157.185

The cassandra.yaml files for each node would then have the following modified property settings.
node0

```
cluster_name: 'GMS Cluster'
initial_token:
seed_provider:
       - seeds: '172.25.157.171'
listen_address: 172.25.157.171
rpc_address: 172.25.157.171
```

node1

```
cluster_name: 'GMS Cluster'
initial_token:
seed_provider:
       - seeds: '172.25.157.171'
listen_address: 172.25.157.177
rpc_address: 172.25.157.177
```

node2

```
cluster_name: 'GMS Cluster'
initial_token:
seed_provider:
       - seeds: '172.25.157.171'
```

```
listen_address: 172.25.157.179
rpc_address: 172.25.157.179
```

node3

```
cluster_name: 'GMS Cluster'
initial_token:
seed_provider:
       - seeds: '172.25.157.171'
listen_address: 172.25.157.185
rpc_address: 172.25.157.185
```

When the installation and configuration are done for all GMS's, you can start each instance.

Load Balancing Between GMS Instances

Load balancing is a computer networking methodology to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives. In a GMS Cluster, Load Balancing is used to distribute the workload across multiple GMS instances. The installation of HAProxy is described here. See also How to setup HAProxy as Load Balancer for Nginx on CentOS 7. Once installed, you have to create a configuration file for HAProxy "haproxy-gms.cfg" and copy the following in the file:

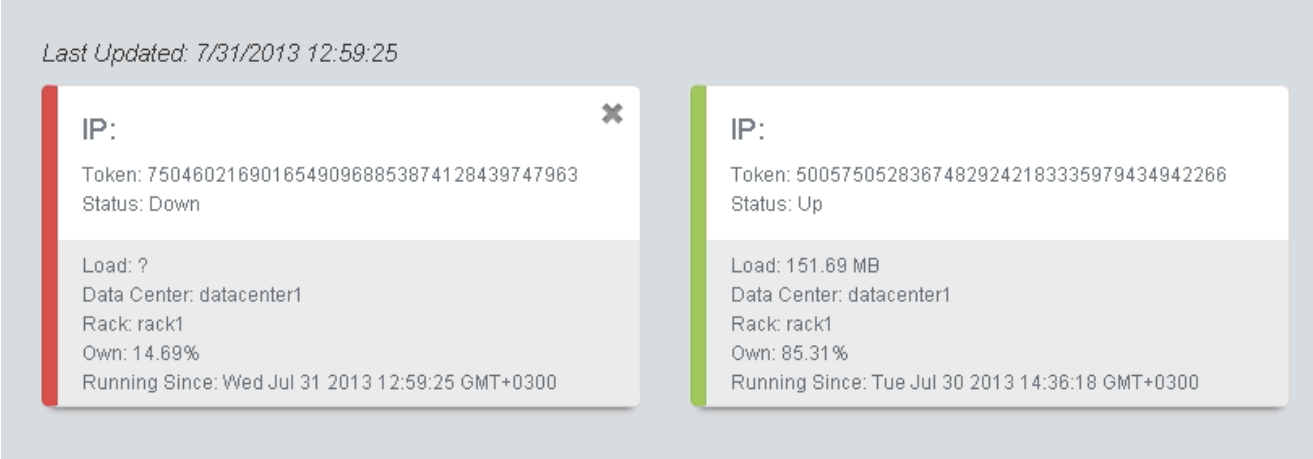
```
global
    daemon
    maxconn 256
defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms
frontend http-in
    bind *:8080
    default_backend cluster_gms
listen admin
    bind *:9090
    stats enable
backend cluster_gms
        balance roundrobin # Load Balancing algorithm
        #following http check, is used to know the status of a GMS (using NodeService from
GMS)
        option httpchk GET /genesys/1/node
        option forwardfor # This sets X-Forwarded-For
        ## Define your servers to balance
        server server1 172.25.157.171:8080 weight 1 maxconn 512 check
        server server2 172.25.157.177:8080 weight 1 maxconn 512 check
        server server3 172.25.157.179:8080 weight 1 maxconn 512 check
        server server4 172.25.157.185:8080 weight 1 maxconn 512 check
```

Once done, you can start HAProxy using the following command:

```
[root@bsgenhaproxy haproxy]# ./haproxy -f haproxy-gms.cfg
```

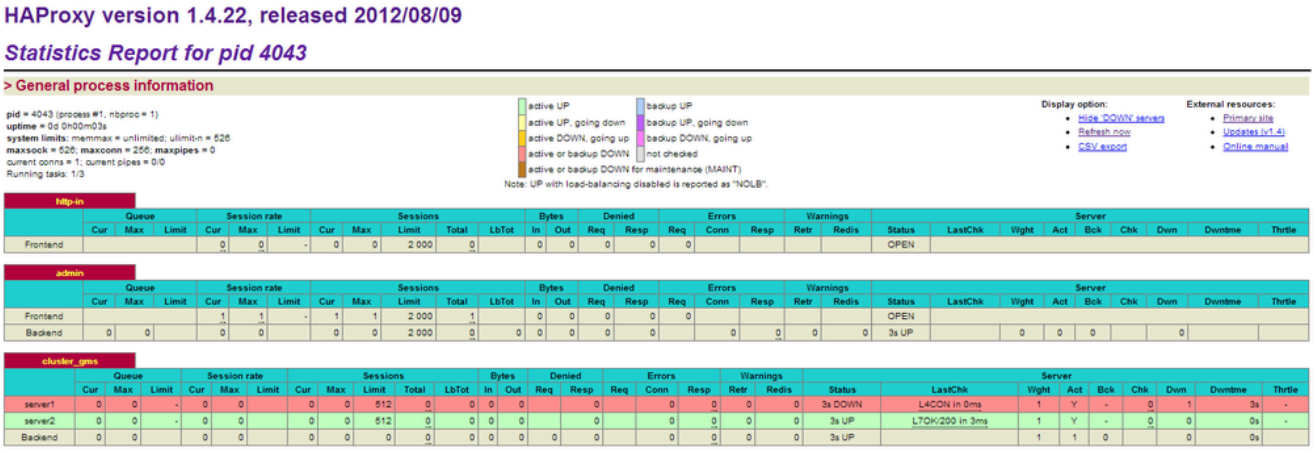
GMS Service Management UI

Cluster view in the GMS Service Management User Interface, Home page:



HAProxy Statistics Report page

The following page is available at: http://<haproxy_host>:9090/haproxy?stats

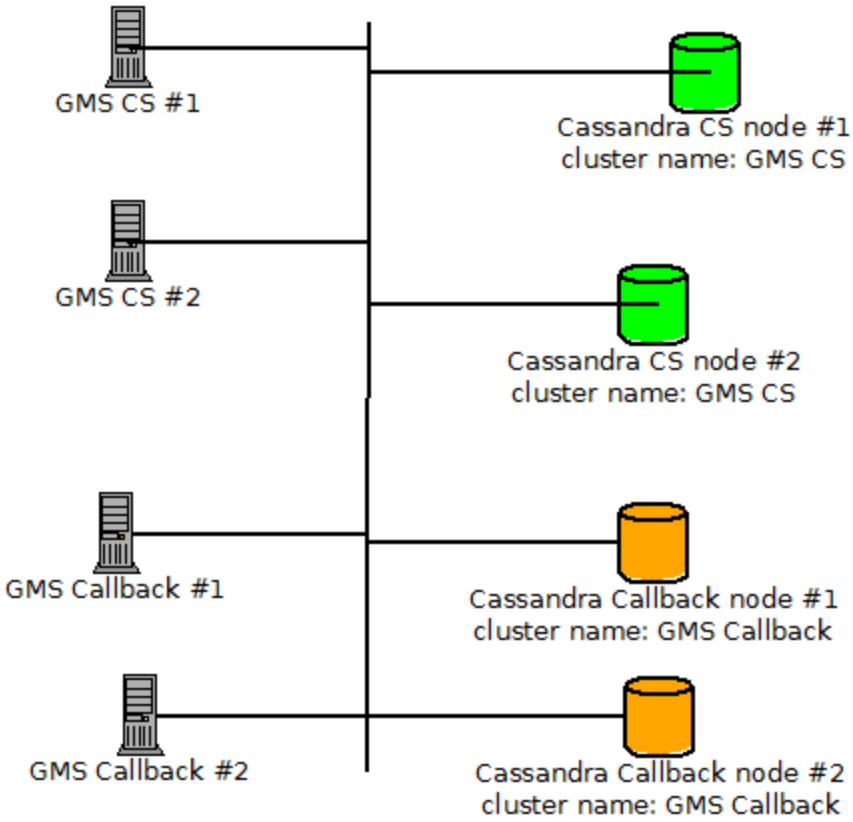


You can now use the HAProxy endpoint http://<haproxy_host>:<haproxy_port> as the main entry point for the Cluster.

Limitations

If you are setting up different clusters of GMS for different purposes (Callback, and so on), you must change the cluster name in order to be able to start GMS (this applies to embedded, and external Cassandra clusters if they are set up for different purposes).

Example of GMS architecture with Cassandra clusters for different purposes:


GMS CS #1
Cassandra CS node #1
cluster name: GMS CS
GMS CS #2
Cassandra CS node #2
cluster name: GMS CS
GMS Callback #1
Cassandra Callback node #1
cluster name: GMS Callback
GMS Callback #2
Cassandra Callback node #2
cluster name: GMS Callback

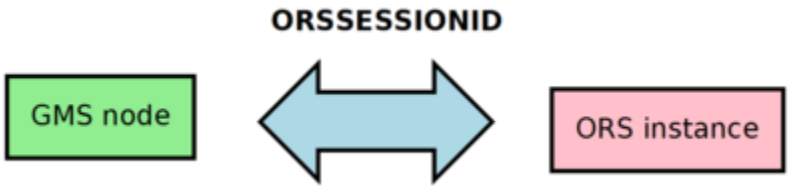
ORS Cookie Support

GMS now supports Cookie-based management, with the ability to store ORS cookies for a given service ID. GMS will use ORS-generated cookies in subsequent requests to the same service.

The ORS cookie prefix is `ORSSESSIONID`.

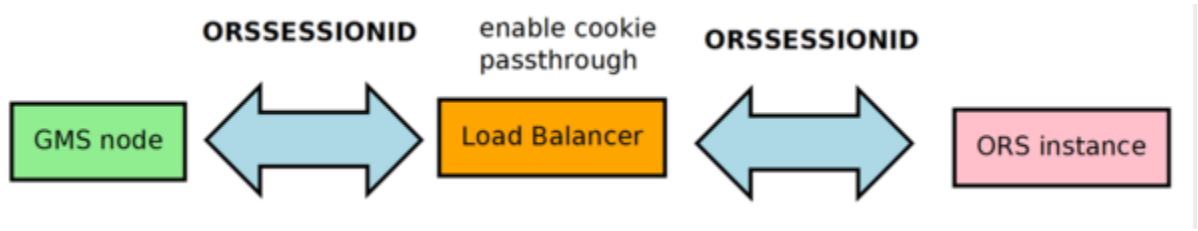
Basic Cookie Usage

When GMS nodes are directly connected to ORS instances, that is, without a load balancer between GMS and ORS, the ORS cookies are managed and stored in GMS for subsequent requests. When another request for the same service session ID is processed and also requires ORS processing, the GMS request to ORS will contain the previously stored cookie.



Load Balancer Cookie Usage

When the environment contains a load balancer between GMS nodes and ORS instances, the load balancer should take into account that ORS cookies must be transferred from and to the GMS nodes. In the load balancer configuration, make sure that cookie management is enabled in order for the ORS cookies to pass back and forth.



Mobile Push Notifications

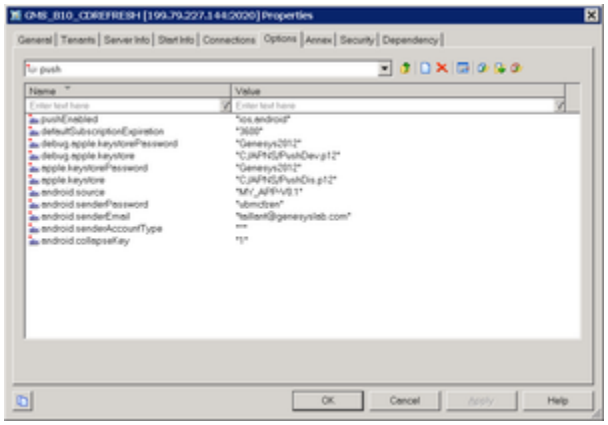
Native Push (Notification Service)

If you are using push notification service, the configurations detailed here should be implemented; however these steps are not mandatory to complete your GMS installation. See also Push Notification Service for more information.

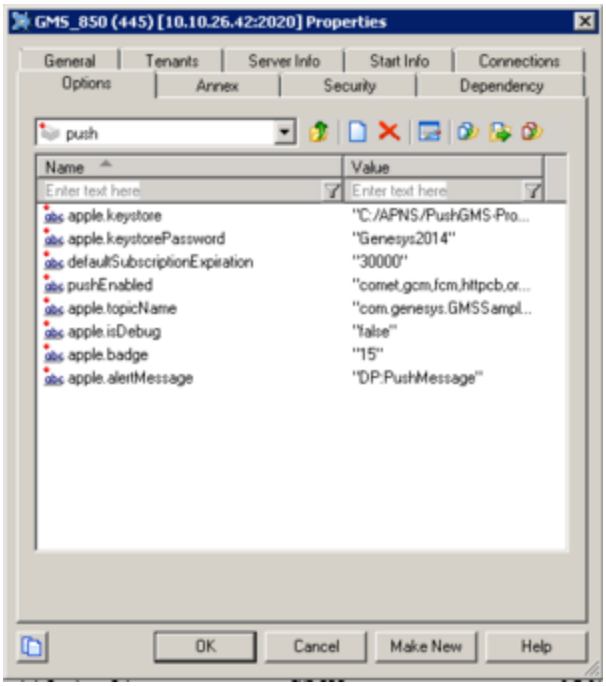
Some services send native push messages to the mobile device. For this to work, both general and device-specific settings need to be configured correctly in the *push* section of your Genesys Mobile Services Application object.

Options set in the *push* section determine how all push notifications are handled by Genesys Mobile Services, regardless of which service is sending the notification.

Note that it is possible to configure this native push notification service for more than one type of device by using a comma-delimited string in the *pushEnabled* option. In this case, be sure to configure the mandatory options for all available device types.



GMS multi-device notification.png



GMS iOS notification.png

Common Device Settings:

- pushEnabled – Device operating system.
- defaultSubscriptionExpiration
- customhttp.url

Mandatory iOS Device Settings:

- debug.apple.keystore - Location of the debug keystore holding the certificates for push notification.
- debug.apple.keystorePassword - Password for the debug keystore.
- apple.keystore - Location of the production keystore holding the certificates for push notification.
- apple.keystorePassword - Password for the production keystore.

Note: The specified location of the Apple iOS push keystore is environment specific, and must be configured based on your environment for iOS push notification to work.

Mandatory Android GCM Device Settings:

- android.gcm.apiKey - A valid Google API key value (Notifications are sent on behalf of this API key, see http://developer.android.com/guide/google/gcm/gs.html).
- android.gcm.retryNumber- Number of retries in case of service unavailability errors.

For additional detail about these options and the allowed values, see the push Section documentation.

Mandatory Android C2DM Device Settings:

Note: Google has deprecated the C2DM Service, and no new users are being accepted. Please use the GCM Service, described above.

- android.senderEmail - Name of a valid mail account. (Notifications are sent on behalf of this account.)
- android.senderPassword - Password of mail account specified in android.senderEmail.
- android.senderAccountType - Specified when initializing C2DM push service.
- android.source - Specified when sending push notifications.
- android.collapseKey - An arbitrary string used to collapse a group of like messages when the device is offline, so that only the last message gets sent to the client.

Mandatory customhttp Settings:

- customhttp.url - http://xxxx/xx
- pushEnabled - comet,gcm,customhttp,orscb,ios

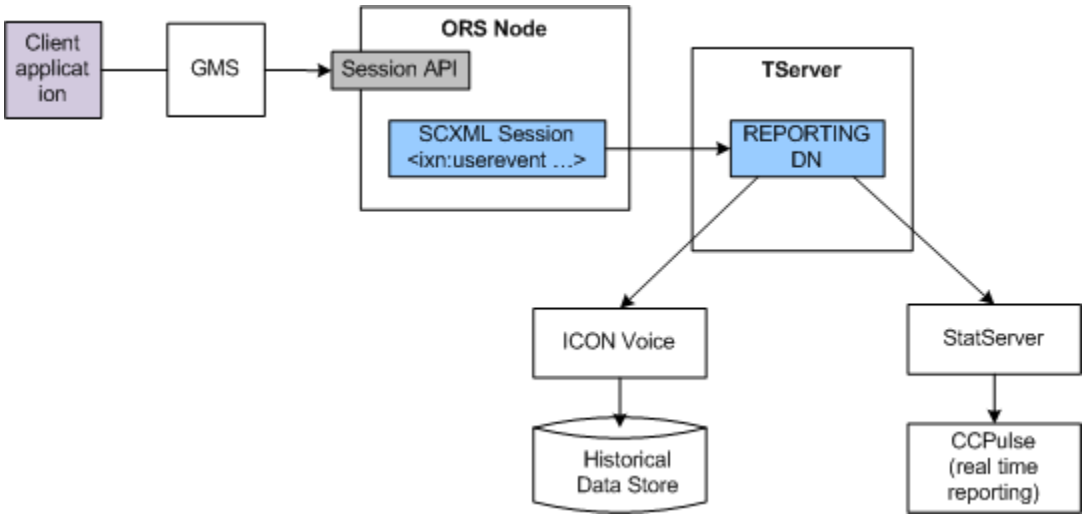
Custom Reporting

Basic Configuration for Real-Time and Historical Reporting Based on T-Server's UserEvent Mechanism

Prerequisites

1. ORS is connected to T-Server
2. StatServer is connected to the same T-Server (if you need real-time reporting)
3. Icon is connected to T-Server and configured to store user events to G_CUSTOM_DATA_P table (if you need historical reporting)

Architecture



Configuration Instructions

1. Create a new DN of type **Extension**. The name of the DN is not important, but it is used inside SCXML scripts so it should be meaningful and recognizable.
Example: Sip_Switch -> DN -> REPORTING

2. Make sure Icon and StatServer are connected to the T-Server that is servicing the switch specified in step 1.
Example: Sip_Server.

3. In Icon configuration in Configuration Manager, add the 'custom-states' section under Options and

create the **GlobalData** option there. List attached data fields you want to capture preceded with data type.
Example:

```
 custom-states/GlobalData=char,gms_SessionId, char,gms_SessionEventSeq, char,gms_ServiceName,
char,gms_UserId, char,gms_externalId,
                         char,gms_ServiceStartAt, char,gms_WaitingForAgent,
char,gms_AgentAvailable, char,gms_UserConnected,
                         char,gms_AgentConnected, char,gms_IxnCompleted,
char,gms_ServiceStoppedAt
```

4. Start Icon and StatServer (if not already started) and use logs to verify they registered on REPORTING DN.

5. Add the following block of code to the beginning of your SCXML flow. This code will set up the `_data.userevent_udata_to_send` variable to store all significant state changes you want to capture from the point of view of reporting. Names should match Icon's `GlobalData` configuration option and StatServer/CCPulse reporting and statistics templates.
Example:

```
 <datamodel>


 </datamodel>
 <script>
        _data.userevent_udata_to_send = {
                'gms_SessionId':_sessionid,
                'gms_SessionEventSeq':0,
                'gms_ServiceName':'your service name here',
                'gms_UserId':,
                'gms_externalId':,
                // service state change timestamps
                'gms_ServiceStartAt': ,
                'gms_WaitingForAgent':,
                'gms_AgentAvailable':,
                'gms_UserConnected':,
                'gms_AgentConnected':,
                'gms_IxnCompleted':,
                'gms_ServiceStoppedAt':
        };
 </script>
```

6. Add the following block of code into your SCXML flow where significant state change is happening:

```
 <script>
        _data.userevent_udata_to_send.gms_WaitingForAgent = new Date().getTime().toString();
        _data.userevent_udata_to_send.gms_SessionEventSeq =
_data.userevent_udata_to_send.gms_SessionEventSeq + 1;
 </script>
 <ixn:userevent requestid="_data.userevent_reqid" resource="({'switch':'SIP_Switch',
'dn':'REPORTING'})"
                    udata="_data.userevent_udata_to_send"/>
```

Example:

```
 <state id="waitForAgent">
        <onentry>
                <script>
                        _data.userevent_udata_to_send.gms_WaitingForAgent = new
Date().getTime().toString();
```

```
                        _data.userevent_udata_to_send.gms_SessionEventSeq =
_data.userevent_udata_to_send.gms_SessionEventSeq + 1;
                </script>
                <ixn:userevent requestid="_data.userevent_reqid"
resource="({'switch':'SIP_Switch', 'dn':'REPORTING'})"
                                udata="_data.userevent_udata_to_send"/>

                <queue:submit route="false"  timeout="_data.queueSubmitTimeout">
                        <queue:targets type="agentgroup">
                                <queue:target name="_data.defaultAgentGroup"/>
                        </queue:targets>
                </queue:submit>
        </onentry>

        <transition event="queue.submit.done" target="agentAvailable"/>
        <transition event="error.queue.submit" target="error">
        </transition>
        <transition event="service.ttl.expired" target="error">
        </transition>
 </state>
```

Verifying Reporting Data

1. Run your scenario by triggering Genesys Mobile Services and Orchestration Server (ORS) APIs directly.
2. Make sure user events are being delivered to StatServer and Icon applications by checking T-Server logs. You should see something like this:

```
 00:34:20.757 Int 04543 Interaction message "RequestDistributeUserEvent" received from 516
("OrchestrationServer")
 -- Absent ThisDN, REPORTING was used
 @00:34:20.7570 [0] 8.1.000.62 send_to_client: message EventACK
        AttributeEventSequenceNumber        0000000000000ef8
        AttributeCustomerID        'Environment'
        AttributeTimeinuSecs        757000
        AttributeTimeinSecs        1348817660 (00:34:20)
        AttributeReferenceID        431
        AttributeThisDN        'REPORTING'
        AttributeUserEvent        RequestDistributeUserEvent
 00:34:20.757 Trc 04542 EventACK sent to [516] (00000003 OrchestrationServer
192.168.27.50:40727)
 @00:34:20.7570 [0] 8.1.000.62 distribute_user_event: message EventUserEvent
        AttributeEventSequenceNumber        0000000000000ef9
        AttributeCustomerID        'Environment'
        AttributeTimeinuSecs        757000
        AttributeTimeinSecs        1348817660 (00:34:20)
        AttributeUserEvent        EventUserEvent
        AttributeUserData        [347] 00 0c 00 00..
                'gms_AgentAvailable'        '1348817660755'
                'gms_AgentConnected'
                'gms_IxnCompleted'
                'gms_ServiceName'        'inbound-delay'
                'gms_ServiceStartAt'        '1348817660553'
                'gms_ServiceStoppedAt'
                'gms_SessionEventSeq'        3
                'gms_SessionId'        '65UA6ISSJH76R340BNDQ2DG0DG000036'
                'gms_UserConnected'
                'gms_UserId'
                'gms_WaitingForAgent'        '1348817660744'
                'gms_externalId'
        AttributeANI        '777'
```

```
        AttributeDNIS        '333'
        AttributeReferenceID        431
        AttributeThisDN        'REPORTING'
 00:34:20.758 Trc 04542 EventUserEvent sent to [508] (0000000c Icon_Voice 192.168.27.50:42678)
 00:34:20.758 Trc 04542 EventUserEvent sent to [588] (00000004 Stat_Server
192.168.27.50:40728)
 00:34:20.758 Trc 04542 EventUserEvent sent to [592] (00000005 Universal_Routing_Server
192.168.27.50:40744)
```

3. Check your Icon log and G_CUSTOM_DATA_P table and make sure data is recorded properly. Examples: **Icon log:**

```
 00:39:19.569 Int 04543 Interaction message "EventUserEvent" received from 65200
("SIP_Server@REPORTING")
 00:39:19.751 Int 04543 Interaction message "EventUserEvent" received from 65200
("SIP_Server@REPORTING")
 00:39:19.766 Int 04543 Interaction message "EventUserEvent" received from 65200
("SIP_Server@REPORTING")
 00:39:19.987 Trc 25016 Persistent Queue GUD: transaction 10929 is committed. 5 records
written into the queue
 00:39:19.987 Trc 25003 Database queue [GUD]: persistent queue transaction 10929 is being
processed.
 00:39:20.001 Trc 25004 Database queue [GUD]: persistent queue transaction 10929 is
processed, committed and removed. 5 records are written.
```

Icon's G_CUSTOM_DATA_P table:

```
 select * from dbo.G_CUSTOM_DATA_P

 8        0        830        REPORTING        0        101        1        2012-09-28
07:43:09.443        1348818189        4496060        65UA6ISSJH76R340BNDQ2DG0DG000038
1        inbound-delay                        1348818189441
 9        0        830        REPORTING        0        101        1        2012-09-28
07:43:09.590        1348818189        4496060        65UA6ISSJH76R340BNDQ2DG0DG000038
2        inbound-delay                        1348818189441        1348818189590
 10        0        830        REPORTING        0        101        1        2012-09-28
07:43:09.600        1348818189        4496060        65UA6ISSJH76R340BNDQ2DG0DG000038
3        inbound-delay                        1348818189441        1348818189590
1348818189596
```

4. Start CCPulse and create a reporting template for monitoring REPORTING DN. If you request the following statistic with time-profile below, it will start accumulating the value of the given key in user-data of your user-events.

```
[TotalCustomValue_UserEvent]
Category=TotalCustomValue
Objects=RegDN,Agent,Place,GroupAgents,GroupPlaces
MainMask=UserEvent
Subject=DNAction
Formula=GetGlobalNumber( "<your key>" )

[TimeProfiles]
Sel2,Selection=2
```

Congratulations: you are done!

Implementing ADDP

The Genesys Advanced Disconnect Detection Protocol (ADDP) protocol is a Keep-Alive protocol between servers. ADDP is embedded into the underlying connection to a server (a protocol in another protocol). The Keep-Alive protocol is shared: client-to-server, and server-to-client. ADDP traces can be activated at the client-side only, server-side only, both sides, or no traces. The protocol is responsible for sending and receiving Keep-Alive packets, setting and canceling timers waiting for the next packet, and sending an appropriate event to the application if the connection is lost.

Configuring ADDP

To enable ADDP between two applications, specify addp as the Connection Protocol when configuring the connection between applications; also, set values for the Local Timeout, Remote Timeout, and Trace Mode properties (off, client side, server side, both). For more information, refer to the *Framework Configuration Manager Help*. For complete instructions on configuring ADDP between two applications using either Genesys Administrator or Configuration Manager, refer to *Appendix A* in the *Framework Deployment Guide*.

Implementing IPv6

Overview

Internet Protocol version 6, commonly known as IPv6, is a network layer protocol for packet-switched inter-networks. It is designated as the successor of IPv4, the current version of the Internet Protocol, for general use on the Internet. Genesys Mobile Services (GMS) supports IPv6.

Note: Refer to the *Framework 8.1 Deployment Guide* for more information about IPv6. In particular, see the IPv6 Appendix.

Genesys Environment Variable

When to use an environment variable:

- If an IPv6 connection is to be established before an application is able to, or must, read information from Configuration Server.
- If you want all Genesys applications on the same host to support IPv6. You only have to configure the host once, rather than configure each application on that host individually. In addition, this host-level setting will override any application-level setting.

Set the environment variable GCTI_CONN_IPV6_ON to true, represented by any non-zero value, to enable IPv6. The default value of this environment variable is false (0), indicating that IPv6 support is disabled. This default value ensures backward compatibility.

Genesys Configuration Option

Do **not** use this procedure if:

- An IPv6 connection is to be established before an application is able to, or must, read information from Configuration Server.
- You want all Genesys applications on a specific host to support IPv6, and you want to set it only once.

Using either Genesys Administrator or Configuration Manager, set the following configuration option in the common section of the options of the component's Application object:

enable-ipv6

Valid Values:

0 - Off (default), IPv6 support is disabled.

1 - On, IPv6 support is enabled.

Mixed IPv4 and IPv6 Environments

You can configure IPv6 and IPv4 in the same environment. In this mixed environment, you can configure connections with servers that support IPv4, IPv6, and both.

To configure this choice, use the Transport parameter, ip-version on the Advanced tab of the Connection Info dialog box for the connection:

ip-version

Default Value: 4,6

Valid Values: 4,6 and 6,4

Specifies the order in which IPv4 (4) and IPv6 (6) are used for the connection with a server that has a mixed IPv4/IPv6 configuration. This parameter has no effect if the environment variable GCTI_CONN_IPV6_ON or the option enable-ipv6 is set to 0. Management Framework components do not support this option. This option also has no affect on connections to Configuration Server that are established before the option value can be read from the Configuration Database.

Java Properties

You can use the following system properties:

- java.net.preferIPv4Stack and
- java.net.preferIPv6Addresses

GMS Alarms

The following alarms can be raised by Genesys Mobile Services based on system status/ configuration:

- Resources Configuration Alarm (EventId 2000): This alarm is raised by GMS when the server detects a problem on resources configuration (Duplicated DN on same/different Groups)
- No more resources Alarm (EventId 2001): This alarm is raised by GMS when no more resources are available in GMS (LOCAL or CLUSTER strategy).
- web_port option Alarm (EventId 2002): This alarm is raised by GMS when at startup, a GMS is not available on `web_port`.
- NO_JDK Alarm (EventId 2003): This alarm is raised when GMS is not started using a JDK (used to compile DFM files). (Introduced in 8.5.006.09)

To create Alarms, refer to Creating the SCS Alarm Conditions.

Configuration Options Reference

This page provides descriptions and explanations of Genesys Mobile Services-specific options.

Overview

By default, the Options tab for your Genesys Mobile Services Application object contains several sections with configuration values.

- Log — Standard log file options for this Application object. For more information about these options, refer to your Genesys Framework documentation.
- gms — Configuration settings used across different services.
- push — Configuration settings for the Notification sub-service.
- resources — Configuration details for handling of resource groups.
- server — This section describes configuration options specific to each Genesys Mobile Services Application instance.
- service.servicename — Every service you want to provide using this instance of Genesys Mobile Services can have a custom entry created using this format. The default installation provides two examples:
  - service.request-interaction
  - service.query
- port_restrictions — Configuration settings for port restrictions.

gms Section

Changes take effect: Immediately.

Option name	Option type	Default value	Restriction on value	Description
http.use_lax_redirect_strategy	Boolean	true	Optional Valid boolean	Instructs GMS to use the Lax Redirect Strategy implementation that automatically redirects all HEAD, GET and POST requests. This strategy relaxes restrictions on automatic redirection of POST

Option name	Option type	Default value	Restriction on value	Description
				methods imposed by the HTTP specification. (Introduced in GMS 8.5.002.02)
http.connection_timeout	Integer	10	Optional Valid integer (seconds)	Connection timeout (in seconds) for http connections to be established from gms to other servers (ORS, httpcb and cluster resource service). Default is set pretty low, so should be on the fast network.
http.socket_timeout	Integer	10	Optional Valid integer (seconds)	Socket timeout (in seconds) for reading data over established http connection from gms to other servers(ORS, httpcb and cluster resource service). Default is set pretty low, so should be on the fast network.
http.max_connections_per_route	Integer	20	Optional Positive integer	GMS will use these number of concurrent connections to connect to each http server. All subsequent concurrent requests will be queued.
http.max_connections_total	Integer	100	Optional Positive integer	GMS will use these number of concurrent connections to connect to any of the http servers.
http.client_port_range	Integer Range (eg., 52000-53000)	System assigned	Optional Max Range (0-65535)	All http client requests from GMS to other servers will use a client socket port from

Option name	Option type	Default value	Restriction on value	Description
				the specified range. If the selected port is already in use, then the request is tried using the next port in a serial fashion. If this option is not specified then the OS will assign a random available port for the request.
http.proxy-auto-config-file	String		Optional Valid URL	Specifies the proxy auto-config (PAC) file location. For example: • file:///C:/GMS/proxy.pac : for a local file • http://127.0.0.1:8082/deploy/proxy.pac (Introduced in 8.5.006.09)
http.proxy-cache-size	Integer	32	Optional Valid integer	Size of the cache that stores URLs that were already processed. If the requested URL is in the cache, GMS will not process the PAC file. (Introduced in 8.5.006.09)
http.proxy-ttl	Integer	5	Optional Valid integer (minutes)	Specifies the interval to refresh PAC content. (Introduced in 8.5.006.09)

push Section

Changes take effect: After restart. The push configuration includes three logical groups of options:

general configuration, push provider configuration, and OS-specific message formatting. For more information about providers and OS-specific message formatting refer to Genesys Mobile Services Push Notification Service.

Note: It is possible for some mandatory options to be absent in this section. In this case, the corresponding push type will be disabled (even if enabled using the *push.pushEnabled* option) and a log entry will be created.

In the following table, values for the **affinity** column can be:

- *general* - The option applies to general behavior.
- *provider* - The option describes the provider configuration used for accessing the target (APPLE APNS service, GOOGLE C2DM service, http address).
- *OS-formatting* - The option affects the resulting OS-specific message output.

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
Common Notification Options					
customhttp.url	provider	string	Mandatory	Valid URL	This is the URL where the notifications will be pushed. The subscriber must provide a URL that will be invoked. GMS posts the payload to this URL (using HTTP POST). The Payload is a JSON object that contains two properties: the `deviceId`, which is the custom id provided at subscription time by subscriber, and the `message`, which is the notification message.
defaultSubscriptionExpiration	general	Integer	Optional	Any Integer>=30	Default subscription expiration (in seconds). If not set or assigned an incorrect value, the default value

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					(30) will be used.
pushEnabled	provider	Collection<String>	Mandatory		A comma-delimited list of strings that describe the enabled push types. Currently, the following push types are supported: **android** and/or **gcm** and/or **ios** and/or **httpcb** and/or **orscb** and/or **customhttp**. Any other push type will be ignored. If an option value is not set then it will be handled as empty string option value (that is, push will be disabled for all supported types and the push service will not work at all).
Apple Notification Options **Note:** Please see the relevant documentation at developer.apple.com for information about OS-Specific message formatting options. Note that if no alert-related options are specified, the *alert* dictionary entry will not be included in the JSON sent to the Apple device.					
apple.keystore	provider	String	Mandatory	valid path	The keystore location (path to the file) for iOS push notifications.
apple.keystorePassword	provider	String	Mandatory	not null (but may be empty string)	The password used to access the keystore. If the password is incorrect then attempts to push messages will fail with corresponding

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					log entries.
apple.content-available	OS-formatting	Integer	Optional	Any String	Provide this key with a value of 1 to indicate that new content is available. This is used to support Newsstand apps and background content downloads. Newsstand apps are guaranteed to be able to receive at least one push with this key per 24-hour window.
apple.alert	OS-formatting	String	Optional	Any String	If specified (not null), it becomes the message text of an alert with two buttons: Close and View. If the user taps View, the application is launched.
apple.alertMessage.body	OS-formatting	String	Optional	Any String	If specified (not null), used as *body* entry in *alert* dictionary (iOS-specific).
apple.alertMessage.action-loc-key	OS-formatting	String	Optional	Any String	If specified (not null), used as *action-loc-key* entry in *alert* dictionary (iOS-specific).
apple.alertMessage.loc-key	OS-formatting	String	Optional	Any String	If specified (not null), used as *loc-key* entry in *alert* dictionary (iOS-specific).
apple.alertMessage.loc-argnames	OS-formatting	String	Optional	Any String	If specified (not null), used as *loc-args* entry in *alert*

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					dictionary (iOS-specific).
apple.alertMessage.launch-image	OS-formatting	String	Optional	Any String	If specified (not null), used as *launch-image* entry in *alert* dictionary (iOS-specific).
apple.badge	OS-formatting	Integer	Optional	Any Integer	If specified (not null), used as *badge* entry in *aps* dictionary (iOS-specific).
apple.sound	OS-formatting	String	Optional	Any String	If specified (not null), used as *sound* entry in *aps* dictionary (iOS-specific).
Android Notification Options					
android.senderEmail	provider	String	Mandatory	Valid mail (sender account registered in Google service)	The valid name of a mail account. Notifications will be sent on behalf of this account. After signing up for C2DM, the sender account will be assigned the default quota, which currently corresponds to approximately 200,000 messages per day. If the default quota is not sufficient for your purposes, please see http://code.google.com/android/c2dm/quotas.html.
android.senderPassword	provider	String	Mandatory	Valid password of registered account	The password for the specified mail account.
android.senderAccountType	provider	String	Mandatory	Not null, may	Specified when

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
				be empty	initializing a C2DM push service.
android.source	provider	String	Mandatory	Not empty	Specified when initializing a C2DM push service.
android.ssl_trust_all	provider	Boolean	Optional		If included and true, indicates that any SSL certificate provided during an establishing HTTPS connection to https://www.google.com/accounts/ClientLogin and https://android.apis.google.com/c2dm/send addresses are considered valid, regardless of their presence in keystore/truststore used by environment. Default value: *false*. Please note that setting this option to *true* is **not recommended**. It is preferred behavior to configure the security system so that only received certificates are permitted.
android.delayWhileIdle	GCM formatting	Boolean	Optional		If included and true, indicates that the message should not be sent immediately if the device is idle. The server will wait for the device to

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					become active (only the last message will be delivered to device when it becomes active). Default, or unspecified, value: *false*.
android.collapseKey	OS-formatting	String	Mandatory	Not empty	An arbitrary string that is used to collapse a group of like messages when the device is offline, so that only the last message gets sent to the client. This is intended to avoid sending too many messages to the phone when it comes back online. Note that since there is no guarantee regarding the order in which messages are sent, the "last" message in this case may not actually be the last message sent by the application server.
android.unavailability_retry_timeout	provider	Integer	Optional	Any Positive Integer	This parameter specifies the default timeout (in seconds) to wait before Google C2DM service can be accessed again if the request

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					returned the 503 code (Service unavailable). Note that this value is ignored if the 503 response from Google contains valid **Retry-After** header. The default value, used if a value is not specified or is incorrect, is 120.
android.gcm.apiKey	provider	String	Mandatory	Not empty	Valid Google API Key. See Google CDM description. Please see http://developer.android.com/guide/google/gcm/gs.html
android.gcm.retryLimit	provider	Integer	Optional		Retry attempts (in case the GCM servers are unavailable).
localizationFileLocation	provider	String	Optional		Location of the file containing the list of localized messages. Please see Localization File.

Note: Please note that the number of C2DM messages being sent is limited. For details, refer to http://code.google.com/android/c2dm/quotas.html.

Each provider can contain 2 *channels* for message sending - **production** and **debug** for each target type. The provider-affiliated options enlisted above describe the production channel. For each provider-related option **<option-name>** the sibling option can be provided with name **debug.<option-name>**. Such options will describe the provider-specific configuration of debug channel for corresponding target type. The debug channel will be enabled for enabled target type only if all mandatory options will be specified for debug channel. The OS-message formatting options do not have production-debug differentiation.

push.provider.*providername* Section

It is possible to create providers by adding **push.provider.*providername*** sections which contain the appropriate credential configuration options that are associated with a given provider. This allows you to control and isolate notifications and events between a given provider and the associated services/applications that are using it. This type of provider name section can only contain provider-related options (as listed in **push** section). All providers are isolated - if the option is not specified in provider's section, then it is not specified. If a mandatory option is missing then the corresponding target type will not be enabled, even if that type is present in the **pushEnabled** option. Please note that we have the following restriction on **providername**: it may only contain alphanumeric characters, the underscore (_), and the minus sign (**-**).

push.provider.event Section

You can define the event definitions associated across providers by adding your **push.provider.event** section, and then setting the appropriate OS-specific attribute options within. This will allow you to add OS-specific attributes to a published event message that is going to any provider's push notification system. This section can contain OS formatting-related options. All other options will be ignored. For more information about providers and OS-specific message formatting refer to Genesys Mobile Services Push Notification Service.

push.provider.event.*eventname* Section

You can define the event definitions associated across providers by adding a custom push.provider.event.**eventname** section, and then setting the appropriate OS-specific attribute options within. This will allow you add OS-specific attributes to a published event message that is going to a specific channel for given group of events tags. This section can contain OS formatting-related options. All other options will be ignored. For more information about providers and OS-specific message formatting refer to Genesys Mobile Services Push Notification Service.

push.provider.*providername*.event.*eventname* Section

You can define the event definitions associated with given provider by adding your push.provider.*providername*.event.**eventname** section, and then setting the appropriate OS-specific attribute options within. This will allow you add OS-specific attributes to a published event message that is going to a specific provider and channel for given group of events tags. This section can contain OS formatting-related options. All other options will be ignored. For more information about providers and OS-specific message formatting refer to Genesys Mobile Services Push Notification Service.

resources Section

Changes take effect: Immediately.

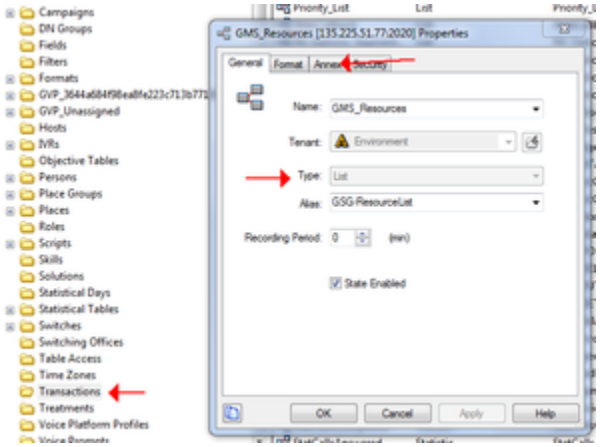
Option name	Option type	Default value	Restriction on value	Description
resources_list_name	String	GMS_Resources	Mandatory	Name of the Strategy

Option name	Option type	Default value	Restriction on value	Description
				configuration object (of type List) which holds configuration details of resources and resource groups.
user_control	String	false	If not present, the default value is used.	This option enables GMS to control resource access based on gms_user header passed in the GMS request. Option is dynamic.
List Object Options: Each section in the Annex is a group that should have distinct list options specified.				
_allocation_strategy	String	RANDOM	Should correspond to one of the supported allocation strategies. Otherwise the default strategy will be used.	Supported strategies: • Random - Allocate a randomly selected resource from the group. No reservations or locks are made, so the same resource can be selected by different users at the same time. • Local - A resource is allocated from the group and reserved/ locked, so that only one user can hold it at the time. For the resource to return to the group it should be released either by the corresponding API call or by a timeout.

Option name	Option type	Default value	Restriction on value	Description
				• Cluster - A resource is allocated from the group and reserved/ locked through the GSG cluster, so that only one user can hold it at the time. For the resource to return to the group it should be released either by the corresponding API call or by a timeout.
_booking_expiration_time	Integer	30	Valid integer (seconds)	Determines the maximum amount of time, in seconds, that a resource may be allocated. If the resource is not released before this time limit elapses, it is automatically returned to the pool of available resources. This option is used with the LOCAL and CLUSTER allocation strategies.
_backup_resource	String		Existing resource	The resource returned if there are no regular resources available. This option is used with the LOCAL and CLUSTER allocation strategies.
List Entries				
All keys not starting with # or _	String			The value is put into the pool of

Option name	Option type	Default value	Restriction on value	Description
				resources. The option name may be anything (since that value is not currently used).

The following screenshot shows an example of an application object configured in Configuration Manager.



Example

```
[Dnis_Pool]
_allocation_strategy = LOCAL
_booking_expiration_timeout = 20
dnis1 = 1-888-call-me1
dnis2 = 1-888-call-me2
dnis3 = 1-888-call-me3
```

Note: For testing purposes, Genesys recommends that you include at least three numbers in the pool. If only a single number is defined in the pool, when the API call is made, that number is allocated for 30 seconds (default). If another API call is made before the number is returned to the pool, an error will occur. Alternatively, if using a single number, use `_allocation_strategy = RANDOM`.

server Section

Changes take effect: Immediately.

Option name	Option type	Default value	Restriction on value	Description
access_code_prefix	Integer		Optional	This value is a range of access_code; the

Option name	Option type	Default value	Restriction on value	Description
				value must be unique for each GMS node accross the cluster. GMS will randomly choose within this range the `access_code_prefix` that it will associate as the prefix for `access_code`. If the option is not present, GMS will use the `nodeId` value instead. An example range is `455,456-458` where the prefix can be `455`, `456`, `457`, or `458`.
external_url_base	String		Optional	Specifies the external url used by the Storage Service to allow the retrieval of a binary attachment. This is useful in the case of a Load Balancer deployment. The valid value is http://<web_host>:<web_port>/, where `web_host` is used by the cluster service to identify a node; and where `web_port` is used by the cluster service to identify a node. The `web_port` value must be the same as the GMS port described in the jetty configuration file, otherwise an alarm will be displayed in Solution Control Interface (SCI) and GMS will stop.
node_id	Integer		Mandatory, two-digit number	Specifies a two-digit number that

Option name	Option type	Default value	Restriction on value	Description
				should be unique in the Genesys Mobile Services deployment. It is used in the generation of DTMF access tokens.
dateFormat	String		Optional	The string used to format dates. Syntax of the string should meet the expectations of java class java.text.SimpleDateFormat. See Simple Date Format for details.
Cluster Service options				
web_host	String	result of InetAddress.getLocalHost().getHostName()	Optional, valid host name	InetAddress.getLocalHost() is not only default value, it is the value that will be used in the most cases. This configuration value is used in cases when there are problems obtaining local name.
web_port	Integer	80	Optional, valid TCP port	Use this option to set a different port than the port that GMS uses (Note: GMS uses port 8080, which can be changed in the `jetty.xml` file). This option can be used in the case of proxy (role of the customer to forward requests). At startup, GMS checks that a GMS is available on the port specified in `web_port`. If a GMS is not available, the web_port option alarm (EventId 2002) is thrown.

Option name	Option type	Default value	Restriction on value	Description
app_name	String	gsg_web	Optional, valid http path	Web application "context" path.

service.*servicename* Section

You can create customized services by adding your service.*servicename* section, and then setting the appropriate options within.

Option name	Option type	Default value	Restriction on value	Description
_type	String		Mandatory	• For Genesys Mobile Services-based services: builtin • For Orchestration Server-based services: ors
_service	String		Mandatory	• For Genesys Mobile Services-based services: The name of the matching service. • For Orchestration Server-based services: The URL of the service's SCXML application.
_ors	String		Optional	**Note:** Only used for Orchestration Server-based services. The URI of the ORS instance or load balancer for this service, allowing different ORS clusters to be used by a single

Option name	Option type	Default value	Restriction on value	Description
				Genesys Mobile Services deployment. Overrides any ORS connections, if they are present. Should be used to direct Genesys Mobile Services to a load balancer that is located in front of multiple ORS instances. This option is specific to each service. If not present, Genesys Mobile Services will use the ORS instance defined in the Application object during installation. Only the base URL needs to be specified. Genesys Mobile Services will use the standard URL path to start ORS session and pass events. Example: `http://<host>:<port>`
_booking_expiration_timeout	Integer		Optional Valid values: Lower limit is 5 seconds and upper limit is 1800 seconds (30 minutes).	This option is specific to the **service.request-interaction** and **service.request-access** services, and applies only to LOCAL and CLUSTER allocation strategies. This option allows you to set a different value per service for the booking expiration timeout. This value can also be passed through the request-access URI parameter. Note that the value passed through the request-access URI parameter will override the value in the service section.
_return_pool_health	Boolean		Optional Valid values: true/false	This option is specific to the **service.request-**

Option name	Option type	Default value	Restriction on value	Description
				interaction and **service.request-access** services, and applies only to LOCAL and CLUSTER allocation strategies. This option allows you to return metrics about pool health used to allocate the resource.
_agent_timeout_notification_message	String		Optional Customer is not online	This option is specific to Chat services using an auto-push notification solution. This option specifies the message that will be sent to the agent in a chat session when the customer is not online but the session is still alive. CometD channel is not working and a new message arrives from Agent.
_client_timeout_notification_message	String		Optional New message from Agent	This option is specific to Chat services using an auto-push notification solution. This option specifies the message that will be sent to the customer as a notification on the device specified at subscription time. This message will be sent when the customer chat session, which is still alive, is not running as an active application (CometD is not working) and the

Option name	Option type	Default value	Restriction on value	Description
				agent is sending a message. The subscription Id that is retrieved from the GMS subscription request must be set as a parameter of the invoked chat service (parameter key: 'subscriptionID') to be able to receive auto push chat notification.
_mandatory_customer_lookup_keys	String	_customer_number	Mandatory	This option is specific to the Callback services. The value is a list of mandatory customer lookup keys that must be in the Callback schedule request. If a lookup key is missing the schedule request, it is rejected with an invalid option message and an HTTP Bad Request error code. By default, the value is `_customer_number`. Example 1: Options: • `_customer_lookup_keys` is empty. • `_mandatory_customer_lookup_keys` is empty. Result: The default value for both options is `_customer_number`. Example 2: Options: • `_customer_lookup_keys = "_customer_number, _email_address"` • `_mandatory_customer_lookup_keys =`

Option name	Option type	Default value	Restriction on value	Description
				"_customer_number" Result: The mandatory value in the request is _customer_number, however, you can also search by the _email_address value. Note: _customer_lookup_keys (for lookup) is a superset of (⊇) _mandatory_customer_lookup_keys (Introduced in 8.5.006.09)

Additional options vary depending on the type of service being created. For more information, refer to documentation for the corresponding service in the Genesys Mobile Services API Reference.

port_restrictions Section

See Restricting Ports for information about these configuration options. Changes in this section require an update to the jetty.xml file on all GMS nodes, and then restarting GMS.

Security

The following table lists additional security configurations that can be used with your GMS installation.

Page	Summary
Security and Access Control	An outline of key security concerns and information about providing access control.
Cassandra Security	Describes security configurations for Cassandra.
Restricting Ports	Provides instructions to configure and enable port control.
Basic Authentication	Describes configuration for basic authentication.
Transport Layer Security	Describes Transport Layer Security (TLS), which enables cryptographic and trusted communications between Genesys clients and servers.
Single Sign-On	Provides the settings needed to configure GMS to use your existing SSO infrastructure.

Security and Access Control

This page discusses deployment topology and advanced configuration that will allow you to secure your Genesys Mobile Services solution.

Note: Although some load balancing considerations are discussed on this page with respect to solution architecture, configuration recommendations are provided on the Configuring Apache Load Balancer page.

Overview of Security, Access Control, and Load Balancing

Genesys Mobile Services makes contact center functionality accessible through a set of REST- and CometD-based APIs. Since these APIs can be used both by clients residing inside and outside the enterprise network, it is important to understand how to protect data that is travelling between solution components. The Genesys Mobile Services solution is designed to work with your existing security infrastructure, relying on third-party components (security proxies) to provide encryption and authorization capabilities.

Deployment requirements

Genesys Mobile Services should always be deployed behind an HTTP security gateway (proxy) performing:

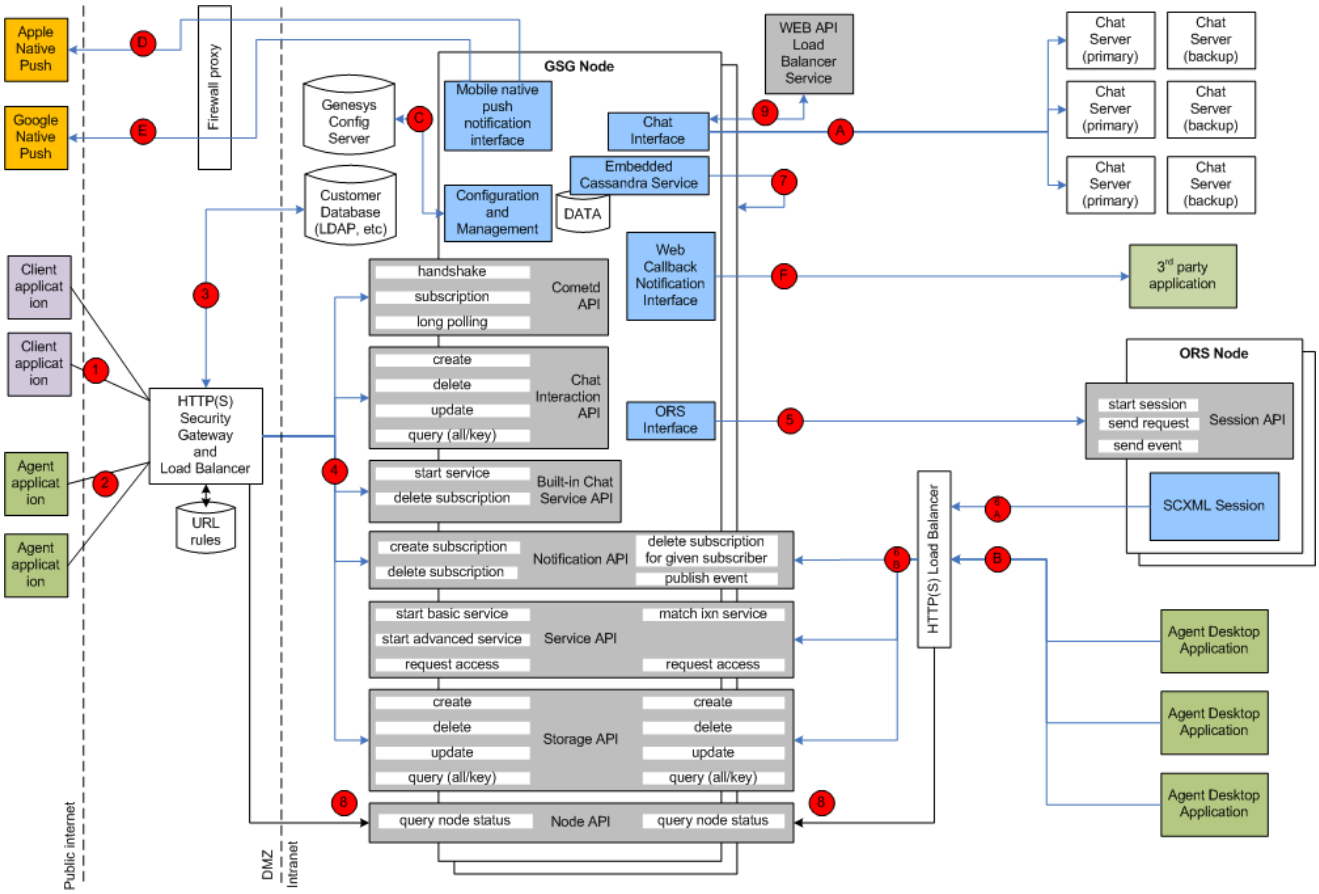
- client authentication (optional)
- TCP port and URL access control
- load balancing functionality, distributing the load between multiple Genesys Mobile Services nodes responsible for processing API requests
- HTTP connection encryption (SSL)

In addition, the HTTP security gateway (proxy) could perform following functions:

- protect against denial of service (DoS) attacks
- authenticate requests using HTTP Basic/Digest, oAuth or other authentication/authorization protocol
- manage client access using IP-based access control
- rate limit API traffic using quotas
- inspect packets for threats and sensitive data

Genesys Mobile Services Deployment Topology

The following image shows architecture and communication links between different components in a typical Genesys Mobile Services solution. A table with recommendations on how to secure these connections follows immediately after.



List of Connections Utilized by a Typical Genesys Mobile Services Deployment

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
Security Gateway and Load Balancer in Front of Genesys Mobile Services Nodes (Customer Client Side)	(1)	User/ customer facing application inside or outside of the corporate firewall: mobile, web or ... based application	Cometd based notification, Chat, Service and Storage APIs exposed through Genesys Mobile Services component	HTTP/ HTTPS	Any deployment-specific port convenient for client applications. For example: 80. See third party gateway/ proxy documentation if you need to	Client is typically a hard-coded server port as part of the API access URL	Client applications should access Genesys Mobile Services APIs through SSL-protected HTTP connections with optional basic/

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
					change/ configure this port. See also GMS->Options->/server/ external_url_base configuration option in Configuration Manager		digest/ oAuth/... client authentication. If a client application has no access to user credentials, then anonymous access is supported but will result in a lower level of security. Clients should be blocked from accessing certain URLs of the API according to the access rules below.
Security Gateway and Load Balancer in Front of Genesys Mobile Services Nodes (Agent Client Side)	(2)	Agent-facing application inside or outside of the corporate firewall: mobile, web or ... based application	CometD-based notification, Chat, Service and Storage APIs exposed through Genesys Mobile Services components	HTTP/ HTTPS	Any deployment-specific port convenient for client applications. For example: 81 See also *GMS/server/ external_url_base* configuration option in Configuration Manager. **Note:** Use a different port from connection (1).	Client is typically a hard-coded server port as part of the API access URL.	Agent applications residing outside of the enterprise intranet can also access Genesys Mobile Services APIs through an SSL-protected HTTP connection. Agent authentication can be performed

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							against Configuration Server, or other service such as LDAP, by the security gateway. Deployments with lower security requirements can allow API access without agent authentication, relying only on uniqueness of the service ID supplied by the application. In this case, it is assumed that only trusted applications would gain access to service ID. Agent authentication is encouraged especially if used outside of the corporate network. Clients should be blocked from accessing certain URLs of the API

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							according to the access rules below.
Enterprise Authentication Server (LDAP, etc)	(3)	Security gateway and load balancer in front of Genesys Mobile Services nodes (client side)	API user client authentication	HTTP/ HTTPS or deployment specific	Deployment-specific port. See third party documentation.	Port is hard coded as part of the security gateway configuration.	Use an appropriate level of security, as required by the Enterprise authentication server.
Two or More Genesys Mobile Services Nodes	(4)	Security gateway and load balancer in front of Genesys Mobile Services nodes (client side)	CometD-based notification, Chat, Service, Storage APIs	HTTP/ HTTPS	Default: 8080. Configured inside *<Genesys Mobile Services deployment directory>/launcher.xml:* *<parameter name="http_port" displayName="Jetty port" mandatory="false">* See also the *GMS->Options->/server/external_url_base* configuration option in Configuration Manager.	Port is hard coded as part of the security gateway configuration.	Jetty container hosting Genesys Mobile Services can be configured to accept SSL-protected connections from security and load balancing gateway. Mutual authentication can also be enabled if required. See below for more information about how to configure SSL connector in Jetty. HTTP basic authentication for security and load

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							balancing gateway can also be configured with the help of Genesys Professional Services. See Restricting Ports for more information about Port Number control.
Two or More ORS Nodes	(5)	Two or more Genesys Mobile Services nodes	ORS scxml session start, stop, send event, etc	HTTP	Default: 7210. Configured in Configuration Manager, inside *<Genesys Mobile Services deployment directory>/lau<parameter name="http_port" displayName="Listening Port" mandatory="false">*	Each Genesys Mobile Services node should have a connection configured in Configuration Manager to each ORS application used by the Genesys Mobile Services solution: *scxml: Applications->getInput Mobile Services->Connections->ORS(1..n)->Server Info->Host and Listening Ports->http.* See *Applications->getInput Mobile Services->Connections->ORS(1..n)->Server Info->Host and Listening Ports->http*.	
Security	(6A)	ORS node	Invoking	HTTP	Deployment	Hard	Security

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
and Load Balancing Gateway in Front of Genesys Mobile Services Nodes		running SCXML session	Genesys Mobile Services APIs: Service (match-interaction), Notification and Storage APIs.		specific. See 3rd party documentation	coded as part of the Genesys Mobile Services API URL inside SCXML session. Could be configured in Configuration Manager and read be SCXML session	proxy should be configured to block access to the API URLs according to the access rules below.
Two or More Genesys Mobile Services Nodes	(6B)	Security and load balancing gateway in front of Genesys Mobile Services nodes	Invoking Genesys Mobile Services APIs: Service (match-interaction), Notification and Storage APIs.	HTTP	Default: 8080. Configured in Configuration Manager, inside *<Genesys Mobile Services deployment directory>/launch.xml:* *<parameter name="http_port" displayName="jetty.port" mandatory="false">*	See 3rd party security and load balancing gateway configuration for URL and port mapping	Jetty container hosting Genesys Mobile Services could be configured to accept SSL protected connections from security and load balancing gateway. Mutual authentication could also be enabled if required. See below for more information about how to configure SSL connector in Jetty. HTTP basic authentication for security

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							and load balancing gateway can also be configured with the help of Genesys Professional Services. See Restricting Ports for more information about Port Number control.
Cassandra Instance Embedded into Genesys Mobile Services Node	(7)	Local and remote Genesys Mobile Services nodes accessing distributed Cassandra storage	Cassandra client API	TCP/IP	Defaults: • rpc_port: 9159 • storage_port: 6934 • ssl_storage_port: 6935 • JMX port: 9192 Configuration: see *<Genesys Mobile Services install dir>/etc/cassandra.yaml*. Check the following parameters: listening, encryption_options. Enable inter-node encryption to protect data travelling between Genesys Mobile Services nodes. see *<Genesys Mobile*	Uses local connection to embedded Cassandra instance.	Most of the transient service session related data is stored in Cassandra database that uses memory and file system of the Genesys Mobile Services node. See *<Genesys Mobile Services install directory>/data* folder. Files there should be protected from unauthorized access. Access to Cassandra ports used

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
					Services install dir>/launcher.xml for JMX configuration.		for synchronization between Genesys Mobile Services nodes and client access should only be allowed from Genesys Mobile Services nodes. Enable internode encryption in Cassandra configuration. More about Cassandra encryption below.
Genesys Mobile Services Server Node	(8)	Security and load balancing gateway in front of the two or more Genesys Mobile Services nodes	Genesys Mobile Services Node API - health check performed by load balancing gateway to improve switchover in case of a Genesys Mobile Services node failure	HTTP	Default port: 8080. Configured inside *<Genesys Mobile Services deployment directory>/launcher.xml: <parameter name="http_port" displayName="jetty.port" mandatory="false">* See also *GMS->Options->/server/external_url_base* configuration option in Configuration Manager.	Hard coded in security and load balancing gateway configuration. See third-party documentation for information how to change it.	No special security is required for this connection. Read-only operation. See Restricting Ports for more information about Port Number control.
Genesys WEB API Server	(9)	Genesys Mobile Services	Chat server load	HTTP	Default port: 9002. Configured	Read dynamically from	No special security is required

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
Node		node	balancing and high availability API		in Configuration Manager. See GMS->Connections->WEB API Server->Server Info->Listening Ports->default port	Configuration Manager configuration: GMS->Connections->WEB API Server. See also GMS->Options->chat/ chat_load_balancer_url_path in Configuration Manager.	for this connection. Read only operation.
Genesys Chat Server - N+1 Primary/ Backup Pairs	(A)	Genesys Mobile Services node	FlexChat protocol	TCP/IP, TLS encrypted if required	Default port: 4856. Configured in Configuration Manager. See *GMS->Connections->WEB API Server->Connections->Chat Server (primary)->Server Info->Listening Ports->webapi port*.	Read dynamically before each chat API call (refresh/ connect/ etc). See: *GMS->Options->chat/ chat_load_balancer_url_path and GMS->Connections->Web API Server* in Configuration Manager. You must confirm TLS is supported and configured for this connection.	Enable TLS encryption if a higher level of protection is required. This is a Platform SDK-based connection supporting standard Genesys security.
Security and Load Balancing Gateway in Front of Genesys Mobile Services Nodes	(B)	Agent desktop application	Invoking Genesys Mobile Services APIs: Service, Notification and Storage APIs.	HTTP/ HTTPS	Deployment specific. See third party documentation.	Hard coded as part of the Genesys Mobile Services API URL inside agent desktop client code.	Security proxy should be configured to block access to the API URLs according to the access rules

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
						Could be configured in Configuration Manager and read dynamically by desktop application.	below.
Genesys Configuration Server	(C)	Genesys Mobile Services node	Reading Genesys Mobile Services node configuration and reacting on changes	TCP/IP protected by TLS is required	Default: 2020. Configured in Configuration Manager configuration file.	Configuration is in *<Genesys Mobile Services installation directory>/startscripts.bat*. The parameters **-app', '-host' and '-port**. See *launcher.xml* in the same directory for description of the parameters.	Enable TLS encryption if a higher level of protection is required. This is a Platform SDK-based connection supporting standard Genesys security. Specify a secure Configuration Server port for the connection.
Apple Mobile Native Message Push Service	(D)	Genesys Mobile Services node	Sending messages to mobile clients using Apple's devices	TCP/IP, binary	host: *gateway.push.apple.com*, port 2195; Controlled by Apple	See Apple Notification for more details on how to configure Genesys Mobile Services to access Apple Push Notification Service.	Apple requires an SSL/ TLS-enabled connection with client side certificates. Make sure your firewall allows access to the *https://gateway.push.appl* URL from all Genesys Mobile Services nodes if

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							you are planning to use this interface.
Google Mobile Native Message Push Service (also called Google Cloud Messaging)	(E)	Genesys Mobile Services node	Sending messages to mobile clients using Android devices	HTTPS	Default: 80. Configured by Google.	See GCM Service for more details on how to configure Genesys Mobile Services to access Google Cloud Messaging	Google require SSL protected connection with client side certificates. Make sure your firewall allows access to the *https://android.googleapis gcm/send* URL from all Genesys Mobile Services nodes if you are planning to use this interface.
Third Party Application Inside Enterprise Network	(F)	Genesys Mobile Services node	Sending Web Callback Notification messages to a third party application subscribed for notifications	HTTP	Third party application specific. Provided by the application as a web callback URL parameter when the subscription is created.	Client configuration: Genesys Mobile Services node will use the web callback URL provided by the third party application when a subscription is created.	Currently HTTPS is not supported for this connection. Used only for notifications within a corporate network. Configure SSL proxy if stronger protection is needed.

Customer Facing API Access Rules for Security Gateway in Front

of Genesys Mobile Services

The following access rules should be used as a model for your environment, allowing a list of services provided by your Genesys Mobile Services deployment to be accessed while restricting the use of internal-only services. **Note:** Only allow access to the limited set of services listed by name.

```
Allow access from the end user application or proxy - mobile, web, etc:
-- async notifications over HTTP API:
If client is using cometd transport (typically for chat):
{base url:port}/genesys/cometd
-- service API:
{base url:port}/genesys/{version}/service/{service name}
{base url:port}/genesys/{version}/service/{id}/storage
{base url:port}/genesys/{version}/service/{id}
{base url:port}/genesys/{version}/service/{id}/{request name}
-- chat media API:
{base url:port}/genesys/{version}/service/{id}/ixn/chat
{base url:port}/genesys/{version}/service/{id}/ixn/chat/refresh
{base url:port}/genesys/{version}/service/{id}/ixn/chat/disconnect
{base url:port}/genesys/{version}/service/{id}/ixn/chat/startTyping
{base url:port}/genesys/{version}/service/{id}/ixn/chat/stopTyping
{base url:port}/genesys/{version}/service/{id}/ixn/chat/*

Only when client need direct access allow (in most cases only ORS/scxml need it):
-- storage API
{base url:port}/genesys/{version}/storage/{ttl}
{base url:port}/genesys/{version}/storage/{data id}/{ttl}
{base url:port}/genesys/{version}/storage/{data id}
{base url:port}/genesys/{version}/storage/{data id}/{key}
-- callback (management) API
{base url:port}/genesys/{version}/service/callback/{callback-execution-name}/{service_id}
{base url:port}/genesys/{version}/service/callback/{callback-execution-name}/{service_id}
{base url:port}/genesys/{version}/service/callback/{callback-execution-
name}/availability?{timestamp=value}
{base url:port}/genesys/{version}/admin/callback/
queues?target={target_name}&end_time={iso_end_time}

Allow access from load balancer for node health check:
{base url:port}/genesys/{version}/node

Allow access from the intranet:
-- admin UI
{base url:port}/genesys/admin/*
{base url:port}/genesys/{version}/admin/*
{base url:port}/genesys/{version}/reports/*
{base url:port}/genesys/{version}/statistic/*
-- datadepot (typically accessed from ORS/scxml):
{base url:port}/genesys/{version}/datadepot/{tenantId}/{activityName}/dates
{base url:port}/genesys/{version}/datadepot/{tenantId}/activities
{base url:port}/genesys/{version}/datadepot/{tenantId}/{activityName}/aggregates
{base url:port}/genesys/{version}/datadepot/{tenantId}/{activityName}/aggregates/unique
-- all built-in services (except chat)
{base url:port}/genesys/{version}/service/request-access
{base url:port}/genesys/{version}/service/match-interaction
-- notification API (typically accessed from ORS/scxml):
{base url:port}/genesys/{version}/notification/subscription
{base url:port}/genesys/{version}/notification/subscription/{id}
{base url:port}/genesys/{version}/notification/publish
```

Mobile Native Push Notification Configuration Details

Two major mobile platforms are currently supported: Android and Apple. Details about Genesys Mobile Services configuration could be found on the Push Notification Service page of the API Reference.

Client Side Port Definitions for Genesys Framework Connections

The following connections support client side port definition functionality:

- Connection to Configuration Server - This port definition is defined as part of the installation, and as an environment variable.
- HTTP Connections between Genesys Mobile Services nodes – This port definition is defined as part of the configuration data associated with the Genesys Mobile Services Application object, and is used to create the connections between Genesys Mobile Services nodes.
- HTTP Connection from Genesys Mobile Services to ORS – This port definition is defined as part of the configuration data associated with the Genesys Mobile Services Application object and is used to create the connection with ORS.
- Genesys Mobile Services to Chat Server Connections - this port definition is defined as part of the configuration data associated with the Genesys Mobile Services application object and is used to create the connection with Chat Server.

Note: Connections from client applications (CometD and REST API requests) do *not* have client side port definitions.

Hiding Sensitive Data in Logs

Genesys recommends using log data filtering to hide sensitive configuration data in log files. The given product will then use that filter to determine what content should be filtered or masked when creating log files. The only real interface to Genesys Mobile Services are APIs, and having to configure filter criteria every time you add a new parameter or API is cumbersome and complicated. Therefore, Genesys Mobile Services supports filtering and masking data in log files for any API parameter that matches the following naming convention:

- Filter: Any parameter name that is prefixed with XXX will be filtered out of the log files entirely. Example: *XXX_FilterParam*
- Mask: Any parameter name that is prefixed with MMM will be masked in the log files, showing in the log with the letters MMM. Example: *MMM_MaskParam*

Configuring SSL Connection Listener for a Jetty Container

Beginning with Jetty 7.3.1, the preferred way to configure SSL parameters for the connector is by configuring the *SslContextFactory* object and passing it to the connector's constructor. Jetty has two SSL connectors–the *SslSocketConnector* and the *SslSelectChannelConnector*. The *SslSocketConnector* is built on top of the Jetty *SocketConnector* which is Jetty's implementation of a blocking connector. It uses Java's *SslSocket* to add the security layer. The *SslSelectChannelConnector* is an extension of Jetty's *SelectChannelConnector* which uses non-blocking IO. For its security layer, it uses java nio *SslEngine*. You can configure these two connectors similarly; the difference is in the implementation. The following is an example of an *SslSelectChannelConnector* configuration. You can configure an *SslSocketConnector* the same way, just change the value of the class to *org.eclipse.jetty.server.ssl.SslSocketConnector*.

```
 <Call name="addConnector">
    <Arg>
      <New class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">
        <Arg>
          <New class="org.eclipse.jetty.http.ssl.SslContextFactory">
            <Set name="keyStore"><SystemProperty name="jetty.home" default="." />/etc/
keystore</Set>
            <Set name="keyStorePassword">OBF:1vny1zlo1x8e1vnw1vn61x8g1zlu1vn4</Set>
            <Set name="keyManagerPassword">OBF:1u2u1wml1z7s1z7a1wnl1u2g</Set>
            <Set name="trustStore"><SystemProperty name="jetty.home" default="." />/etc/
keystore</Set>
            <Set name="trustStorePassword">OBF:1vny1zlo1x8e1vnw1vn61x8g1zlu1vn4</Set>
          </New>
        </Arg>
        <Set name="port">8443</Set>
        <Set name="maxIdleTime">30000</Set>
      </New>
    </Arg>
 </Call>
```

Choosing a Directory for the Keystore

The keystore file in the example above is given relative to the Jetty home directory. For production, choose a private directory with restricted access to keep your keystore in. Even though it has a password on it, the password may be configured into the runtime environment and is vulnerable to theft. You can now start Jetty the normal way (make sure that *jcert.jar, jnet.jar* and *jsse.jar* are on your classpath) and SSL can be used with a URL like: https://localhost:8443/

Setting the Port for HTTPS

Remember that the default port for HTTPS is 443 not 80, so change 8443 to 443 if you want to be able to use URLs without explicit port numbers. For a production site it normally makes sense to have an *HttpListener* on port 80 and a *SunJsseListener* on port 443. Because these are privileged ports, you might want to use a redirection mechanism to map port 80 to 8080 and 443 to 8443, for example. The most common mistake at this point is to try to access port 8443 with HTTP rather than HTTPS.

Redirecting HTTP requests to HTTPS

To redirect HTTP to HTTPS, the webapp should indicate it needs CONFIDENTIAL or INTEGRAL connections from users. This is done in web.xml:

```
<web-app>
...
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Everything in the webapp</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
</web-app>
```

Then you need to tell the plain HTTP connector that if users try to access that webapp with plain HTTP, they should be redirected to the port of your SSL connector (the "confidential port"):

```
<Call name="addConnector">
 <Arg>
    <New class="org.eclipse.jetty.nio.SelectChannelConnector">
       ...
       <Set name="confidentialPort">443</Set>
    </New>
 </Arg>
</Call>
```

Using the Denial of Service Filter

The Denial of Service (DoS) filter limits exposure to request flooding, whether malicious, or as a result of a misconfigured client. The DoS filter keeps track of the number of requests from a connection per second. If the requests exceed the limit, Jetty rejects, delays, or throttles the request, and sends a warning message. The filter works on the assumption that the attacker might be written in simple blocking style, so by suspending requests you are hopefully consuming the attacker's resources. The DoS filter is related to the QoS filter, using Continuations to prioritize requests and avoid thread starvation. See the Jetty documentation for more information: http://wiki.eclipse.org/Jetty/Reference/DoSFilter

The DoS filter is configured in the GMS `web.xml` file, located in the `webapp/WEB-INF` directory. The default configuration is:

```
<filter>
<filter-name>DoSFilter</filter-name>
<filter-class>org.eclipse.jetty.servlets.DoSFilter</filter-class>
<init-param>
<param-name>maxRequestsPerSec</param-name>
<param-value>100</param-value>
</init-param>
<init-param>
<param-name>ipWhitelist</param-name>
<param-value>127.0.0.1</param-value>
</init-param>
</filter>
```

Cassandra Security

This page discusses security configurations for Cassandra.

Protecting Data Stored in the Cassandra Database

The *<Genesys Mobile Services installation directory>/etc/cassandra.yaml* file enables or disables encryption of Cassandra inter-node communication using TLS_RSA_WITH_AES_128_CBC_SHA as the cipher suite for authentication, key exchange, and encryption of the actual data transfers. To encrypt all inter-node communications, set to *all*. You must also generate keys, and provide the appropriate key and trust store locations and passwords. Details about Cassandra options are available from:

- internode_encryption
- authenticator

All transient service session-related data is stored in a Cassandra database that uses memory and the file system. See the *<Genesys Mobile Services installation directory>/data* folder. Files located here should be protected from unauthorized access.

Cassandra Authentication

The following steps show how to protect access to the Cassandra `gsg` keyspace. All of the files and options noted below are in the GMS installation package.

1. Before editing the following files, stop the GMS node.
   - To comment a line, add a # at the beginning of the line.
   - To uncomment a line, delete the # from the beginning of the line.
2. When you have finished editing and saving the files, restart the GMS node.

cassandra.yaml file

1. Open the *<Genesys Mobile Services installation directory>/etc/cassandra.yaml* file.
   - Comment the line: `authenticator: org.apache.cassandra.auth.AllowAllAuthenticator`.
   - Uncomment the line: `#authenticator: com.genesyslab.gsg.storage.auth.SimpleAuthenticator`.
   - Comment the line: `authority: org.apache.cassandra.auth.AllowAllAuthority`.
   - Uncomment the line: `#authority: com.genesyslab.gsg.storage.auth.SimpleAuthority`.
2. Save and close the file.

access.properties file

1. Open the *<Genesys Mobile Services installation directory>/etc/access.properties* file.
   - Uncomment the line: `#<modify-keyspaces>=cassandra`. Note that the value, `cassandra`, is the user login that will be used in the GMS server node (server-side and client-side). You must change this user login. However, for the examples shown here, `cassandra` will continue to be used as the user login. This line tells the Cassandra server that the `cassandra` user can modify keyspaces, which is needed in order to create the `gsg` keyspace.
   - Uncomment the line: `#gsg.<rw>=cassandra`. This grants read/write permissions to the `cassandra` user login to be able to read and write into the `gsg` keyspace.
2. Save and close the file.

passwd.properties file

1. Open the *<Genesys Mobile Services installation directory>etc/passwd.properties* file.
2. Uncomment one of the following lines:
   - `#cassandra=pelops` if you want to use a PLAIN text password in this file. Make sure that you also change the user login and password.
   - `#cassandra=b57694b2c9cfc6fbaf00a7033b2a7e4c` if you want to use an MD5 password in this file. Make sure that you also change the user login and password. You can use any MD5 tools to generate the MD5 result of your password, for example, using md5sum:

```
$ echo -n "pelops" | md5sum

b57694b2c9cfc6fbaf00a7033b2a7e4c
```

3. Save and close the file.

launcher.xml

1. Open the *launcher.xml* file.
2. Locate the parameter `cassandra.login`.
3. In the `<format type="string" default=""/>` line within this parameter, change the default by using your cassandra user login previously defined. For example, `<format type="string" default="cassandra"/>`.
4. Locate the parameter `cassandra.password` .
5. Change the following line in this parameter: `<format type="string" default=""/>` using the password defined for your cassandra user login. This password must be in PLAIN text (even if you used MD5 for hashing your password in the previous steps). For example, `<format type="string" default="pelops"/>`.
6. Locate the parameter `cassandrapasswordmode` and change the mode according to the way you recorded your password in the `passwd.properties` file:
   - `<format type="string" default="-Dpasswd.mode=MD5" />` for a MD5 password or
   - `<format type="string" default="-Dpasswd.mode=PLAIN" />` if your password was recorded in plain text.

7. Save and close the file.

The following shows an example of the parameters in the launcher.xml file:

```
<parameter name="cassandralogin" displayName="cassandra.login" mandatory="false">
<description><![CDATA[ Cassandra Server Login for Client]]></description>
<valid-description><![CDATA[]]></valid-description>
<effective-description/>
<format type="string" default="user"/>
<validation>
</validation>
</parameter>

<parameter name="cassandrapassword" displayName="cassandra.password" mandatory="false">
<description><![CDATA[ Cassandra Server Password for Client]]></description>
<valid-description><![CDATA[]]></valid-description>
<effective-description/>
<format type="string" default="password "/>
<validation>
</validation>
</parameter>



<parameter name="password" displayName="cassandrapassword" mandatory="true" hidden="true"
readOnly="true">

<description><![CDATA[Security: cassandra password file]]></description>
<valid-description><![CDATA[]]></valid-description>
<effective-description/>
<format type="string" default="-Dpasswd.properties=./etc/passwd.properties" />
<validation></validation>
</parameter>
<parameter name="access" displayName="cassandraaccess" mandatory="true" hidden="true"
readOnly="true">
<description><![CDATA[Security: cassandra access file]]></description>
<valid-description><![CDATA[]]></valid-description>
<effective-description/>
<format type="string" default="-Daccess.properties=./etc/access.properties" />
<validation></validation>
</parameter>
<parameter name="passwdmode" displayName="cassandrapasswordmode" mandatory="true"
hidden="true" readOnly="true">
<description><![CDATA[Security: cassandra password mode]]></description>
<valid-description><![CDATA[]]></valid-description>
<effective-description/>
<format type="string" default="-Dpasswd.mode=MD5" /> # or PLAIN
<validation></validation>
</parameter>
```

Limitation: If the password in GMS is no longer required, you must undo all of the changes in the files.

Cassandra Gossip TLS

In Cassandra version 1.1.x (the Cassandra version in GMS), internode (gossip) encryption is set up in the *cassandra.yaml* file. Locate the following lines in the *cassandra.yaml* file:

```
encryption_options:
 internode_encryption: none
 keystore: conf/.keystore
 keystore_password: cassandra
 truststore: conf/.truststore
 truststore_password: cassandra
```

Replace `internode_encryption: none` with `internode_encryption: all`, as shown in the following example:

```
encryption_options:
 internode_encryption: all
 keystore: conf/.keystore
 keystore_password: cassandra
 truststore: conf/.truststore
 truststore_password: cassandra
```

For managing keystore and truststore (and password), see the Oracle documentation keytool-Key and Certificate Management Tool or the Oracle security guide.

Cassandra JMX TLS

Cassandra monitoring and management can be done using a Java Management Extensions (JMX) tool. The JMX access must be protected in order to avoid any remote managing on the GMS embedded Cassandra. To protect JMX access, edit the `launcher.xml` file that contains the following lines (by default):

```
<parameter name="jmxport" displayName="jmxport" mandatory="true" hidden="true"
readOnly="true">
    <description><![CDATA[JMX related]]></description>
    <valid-description><![CDATA[]]></valid-description>
    <effective-description/>
    <format type="string" default="-Dcom.sun.management.jmxremote.port=9192" />
    <validation></validation>
</parameter>
<parameter name="jmxssl" displayName="jmxssl" mandatory="true" hidden="true" readOnly="true">
    <description><![CDATA[virtual machine related]]></description>
    <valid-description><![CDATA[]]></valid-description>
    <effective-description/>
    <format type="string" default="-Dcom.sun.management.jmxremote.ssl=false" />
    <validation></validation>
</parameter>
<parameter name="jmxauthenticate" displayName="jmxauthenticate" mandatory="true"
hidden="true" readOnly="true">
    <description><![CDATA[virtual machine related]]></description>
    <valid-description><![CDATA[]]></valid-description>
    <effective-description/>
    <format type="string" default="-Dcom.sun.management.jmxremote.authenticate=false" />
    <validation></validation>
</parameter>
```

By default, the TLS and authentication parameters are disabled:

```
   com.sun.management.jmxremote.ssl=false
   com.sun.management.jmxremote.authenticate=false
```

For information about enabling these parameters and managing JMX and TLS, see the *Monitoring and*

Management Using JMX Technology chapter in the Oracle Java SE Monitoring and Management Guide.

Restricting Ports

You can control access to GMS APIs by configuring your firewall to allow or block other hosts (such as public internet, intranet, specific IP addresses, and so on) from accessing TCP/IP ports on the host where GMS is running.

You can configure and enable port control through the following process:

1. Set configuration options.
2. Copy code snippet from Service Management UI.
3. Paste code snippet into the jetty.xml file.
4. Restart GMS.

Configuration

Configuration Options

You can control port access to GMS APIs by adding a `port_restrictions` section in the GMS configuration, at the node level or cluster level. This section is optional and not defined in the default template. The content of this section is a list of key/values. Where key is an URI pattern (`/genesys/1/storage/*, /genesys/1/service/*, /genesys/1/service/request-interaction`, and so on), and the value is a list of ports or a port range.

1. In Configuration Manager, select *Environment* > *Applications.*
2. Locate and open the Application object for GMS.
3. Select the *Options* tab.
4. Add the `port_restrictions` section, and then set the options and values with the URL and ports you wish to control.
5. Save your changes.

Example `port_restrictions` section:

Option Name	Option Value	Description
/genesys/1/storage*	80-90	Storage API will be accessible from port 80 to port 90.
/genesys/1/service/*	92-98,100	Services API will be accessible from port 92 to port 98, plus the port 100.

Notes:

- There are no default values or default option names. You can define various URL patterns; such as

`/genesys/1/resource*, /genesys/1/resource*, /genesys/1/service/*, /genesys/1/service/request-interaction`, and so on.

- If the request is sent on another port, an HTTP error `403 Forbidden` occurs.
- The Admin UI and APIs not listed in the `port_restrictions` section will be available on all ports listed in the `port_restrictions` section.

Service Management UI

1. In the Service Management User Interface, go to the *Lab* > *Config* tab. The xml snippet is displayed.
2. Select and copy the entire `Set connectors` code snippet.

Example code snippet:

Ports Restriction Configuration:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/jetty/configure.dtd">

<!-- ============================================================= -->
<!-- Configure the Jetty Server                                    -->
<!-- This file has been generated. Do not modify it                -->
<!-- ============================================================= -->


<Configure id="Server" class="org.eclipse.jetty.server.Server">
        ...

    <!-- ============================================================= -->
    <!-- Set connectors                                                -->
    <!-- ============================================================= -->

        <Call name="addConnector">
      <Arg>
          <New class="org.eclipse.jetty.server.nio.SelectChannelConnector">
            <Set name="host"><Property name="jetty.host" /></Set>
            <Set name="port"><Property name="jetty.port" default="8080"/></Set>
            <Set name="maxIdleTime">300000</Set>
            <Set name="Acceptors">2</Set>
            <Set name="statsOn">false</Set>
            <Set name="confidentialPort">8443</Set>
            <Set name="lowResourcesConnections">20000</Set>
            <Set name="lowResourcesMaxIdleTime">5000</Set>
          </New>
      </Arg>
```

jetty.xml File

1. Go to the `<GMS_HOME>/etc/jetty.xml` file, and add the code snippet in the `Set connectors` section of the file, after the GMS default HTTP connector (used to open default port 8080).

Example `Set connectors` section:

```
<!-- ============================================================= -->
   <!-- Set connectors                                                -->
```

```
<!-- ============================================================= -->

     <Call name="addConnector">
  <Arg>
      <New class="org.eclipse.jetty.server.nio.SelectChannelConnector">
        <Set name="host"><Property name="jetty.host" /></Set>
        <Set name="port"><Property name="jetty.port" default="8080"/></Set>
        <Set name="maxIdleTime">300000</Set>
        <Set name="Acceptors">2</Set>
        <Set name="statsOn">false</Set>
        <Set name="confidentialPort">8443</Set>
        <Set name="lowResourcesConnections">20000</Set>
        <Set name="lowResourcesMaxIdleTime">5000</Set>
      </New>
  </Arg>
</Call>


    <!-- Paste Port Definition Snippet here -->
```

2. Restart GMS.

Disabling Port Restrictions

1. In Configuration Manager, select *Environment* > *Applications.*
2. Locate and open the Application object for GMS.
3. Select the *Options* tab.
4. Select the `port_restrictions` section.
5. Right-click, and enter a hash tag (#) in front of `port_restrictions` so it appears like this: `#port_restrictions`. The port restrictions are now disabled, and the Service Management User Interface > *Lab* > *Config* tab will display: `port restrictions has not been enabled.`

Basic Authentication

HTTP Basic Authentication is a method for an HTTP client to provide a user name and a user password for each HTTP request where a resource needs access control.

GMS supports Basic Authentication on the following services:

- Storage service
- Notification service
- and all services that have a section (with the prefix **service.**) in the GMS application

Note: Best practice is to use Basic Authentication over HTTPS.

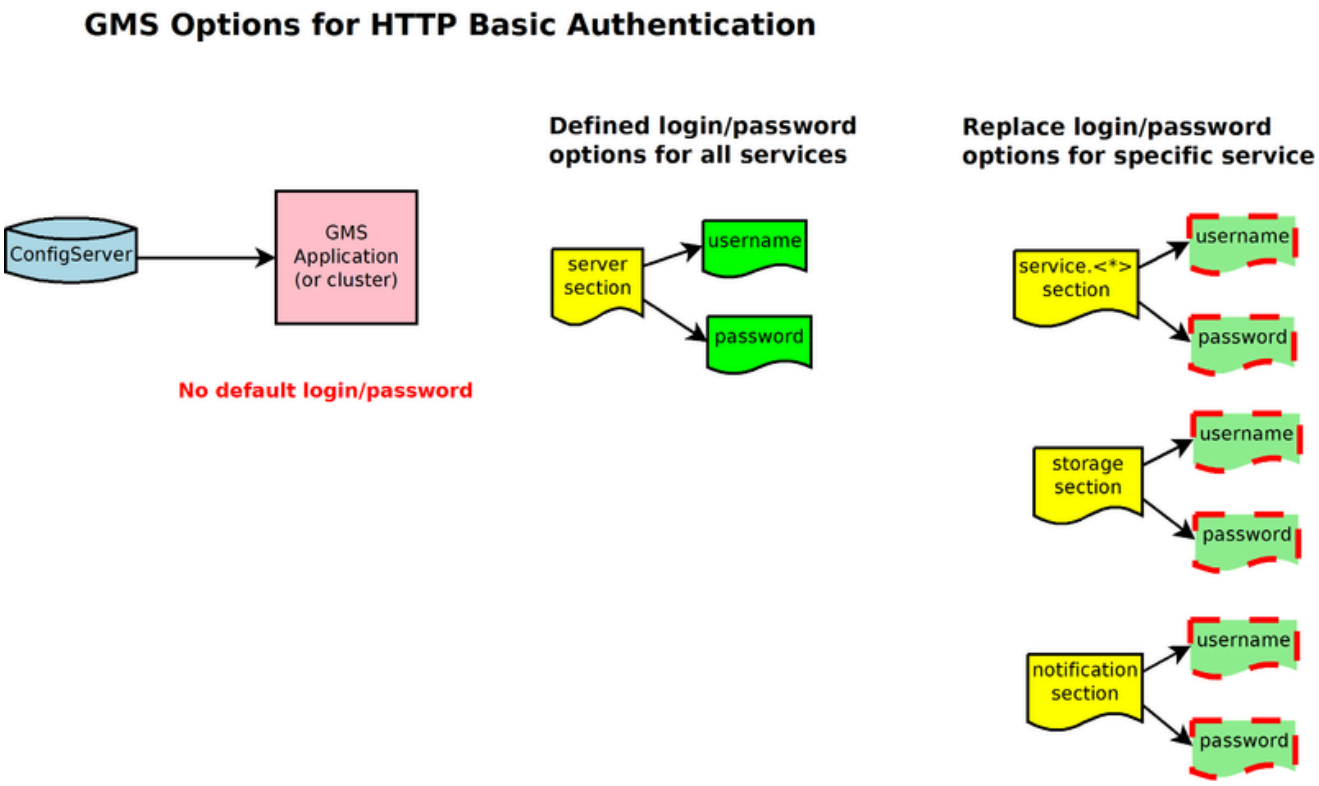
Configuration

By default, the Basic Authentication feature is turned off. The following sections and options must be set in the GMS application in order to turn on this feature.

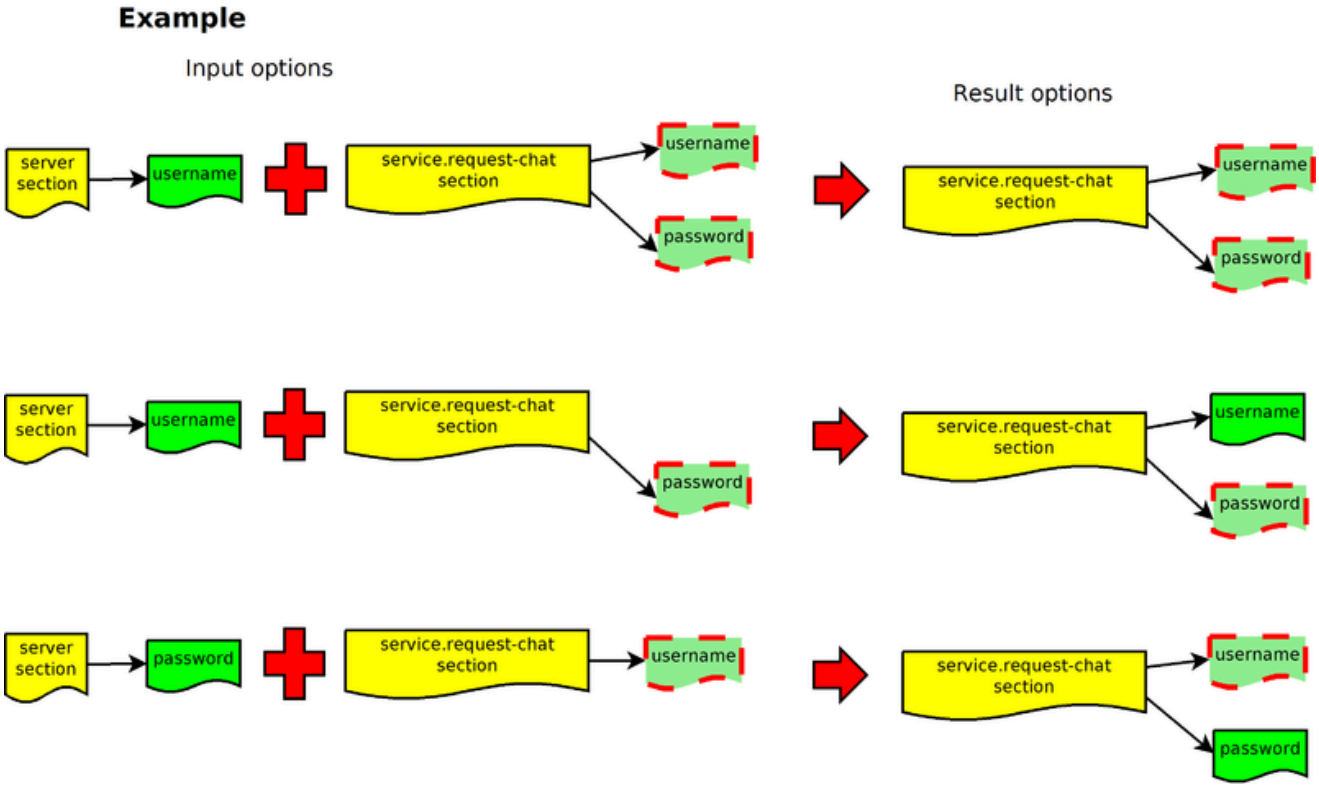
Note: Basic Authentication options are taken into account dynamically.

Section	Option	Supersedes	Description
server	realm		Defines the authentication scheme. The default value is `Genesys Application Configuration Needed.`
server	username		Defines a global username for all services. Note: Without the password option, no authentication is effective.
server	password		Defines a global password for all services. With this option, Basic Authentication is turned on for all services.
service.*	username	server/username	Defines a specific username for one service. Note: Without the password option in the same service section, the server section password is

Section	Option	Supersedes	Description
			used. If there is no server section password, no authentication is applied.
service.*	password	server/password	Defines a specific password for one service.
storage	username	server/username	Defines a specific username for the storage service. Note: Without the password option in the same service section, the server section password is used. If there is no server section password, no authentication is applied.
storage	password	server/password	Defines a specific password for the storage service.
notification	username	server/username	Defines a specific username for the notification service. Note: Without password option in the same service section, the server section password is used. If there is no server section password, no authentication is applied.
notification	password	server/password	Defines a specific password for notification service.

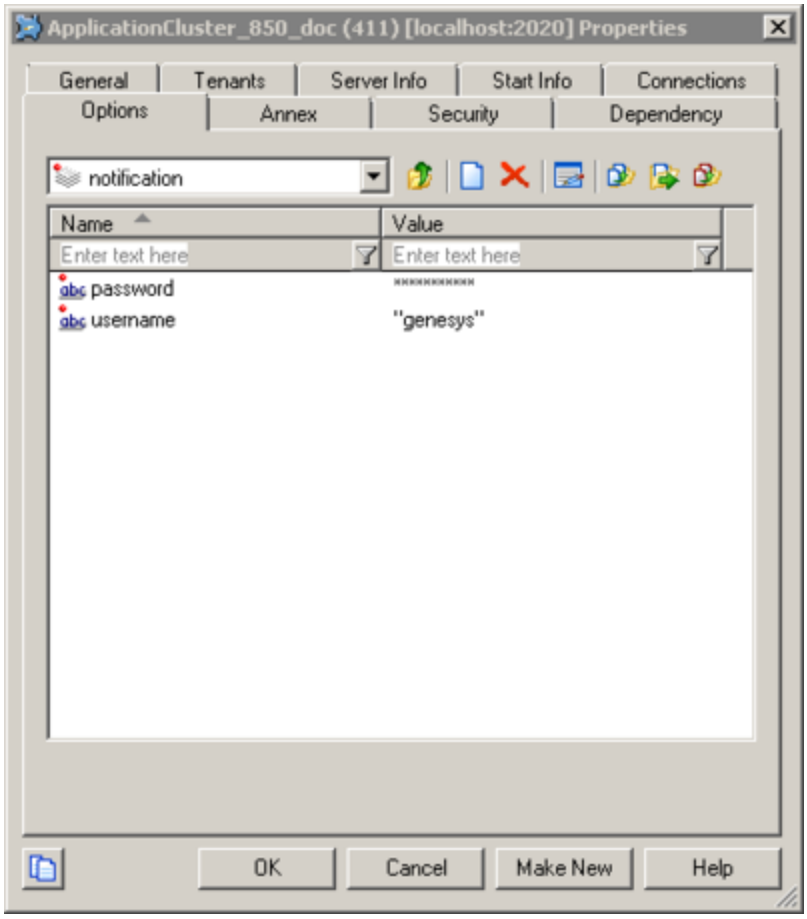

GMS Options for HTTP Basic Authentication
Defined login/password options for all services
Replace login/password options for specific service
ConfigServer
GMS Application (or cluster)
No default login/password
server section
username
password
service.<*> section
username
password
storage section
username
password
notification section
username
password

Precedence Example

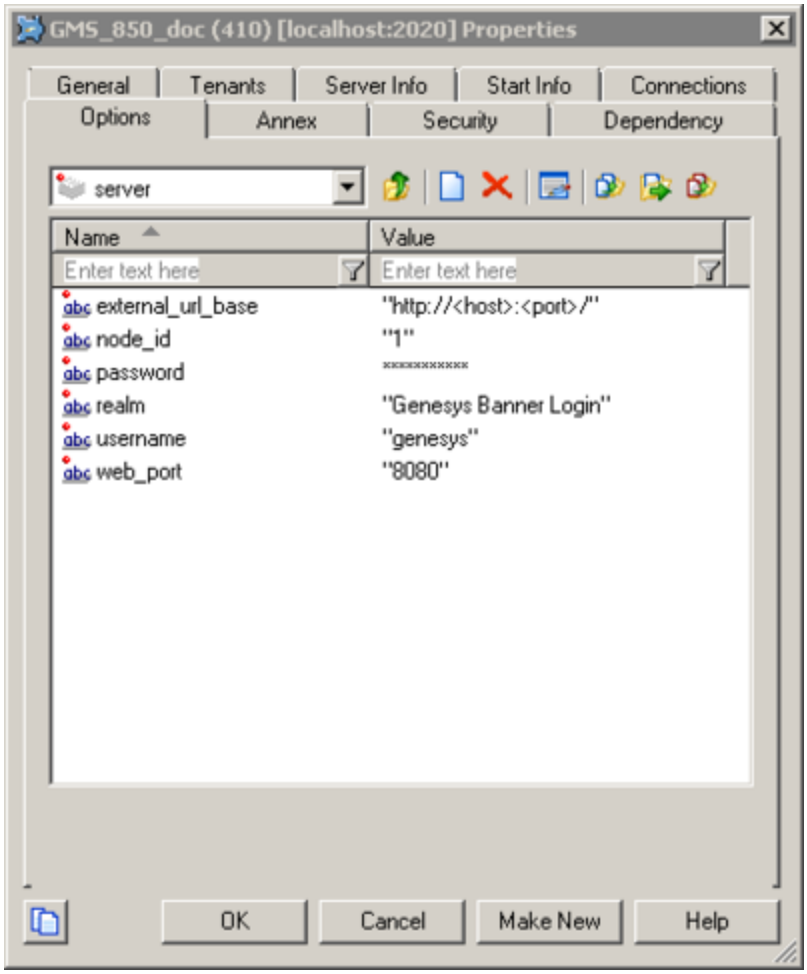


Configuration Option Examples

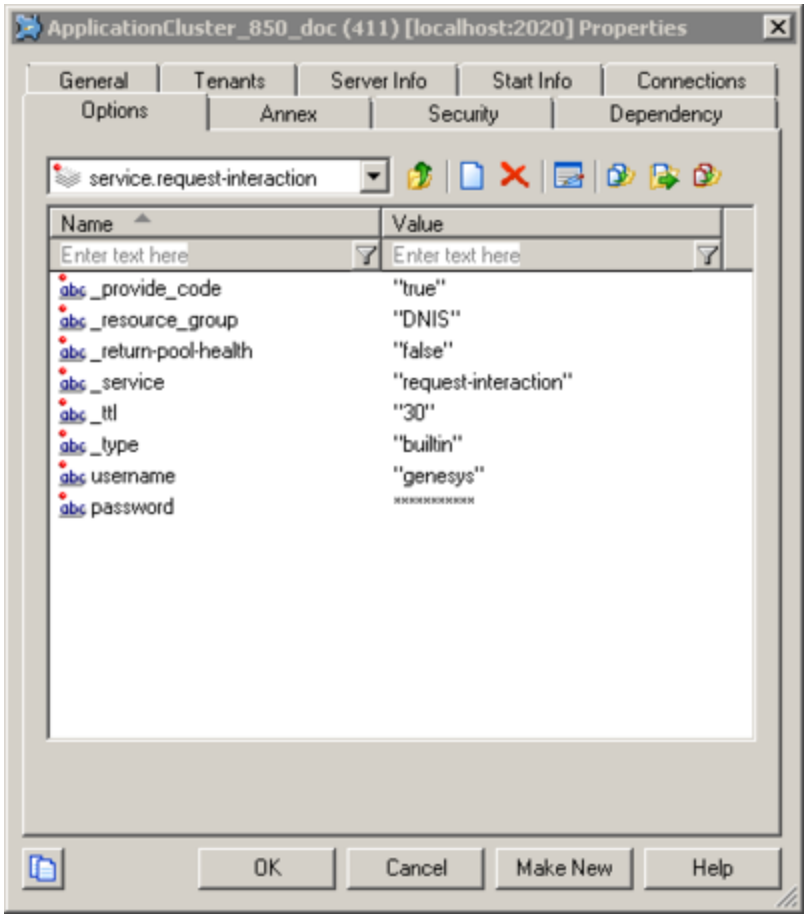
Basic Authentication Username and Password for Notification API:



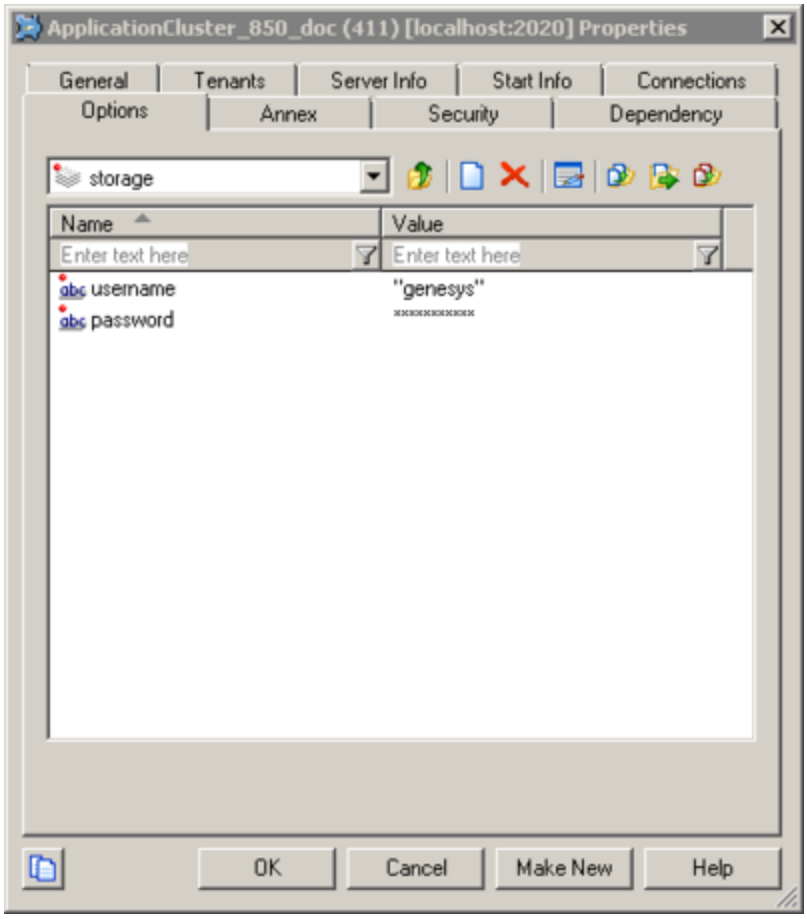
Basic Authentication Username and Password for GMS Node API:



Basic Authentication Username and Password for Service API:



Basic Authentication Username Password for Storage API:



Client Side

When Basic Authentication is turned on from the GMS server-side, the client must manage the HTTP Authentication header to set the username and password in the Authorization header. Only HTTP Basic Authentication is supported. The header request for authentication should look like the following (credential is a string with a specific format [username:password], and base64 encoded):

```
Request
> GET /genesys/1/storage/id HTTP/1.1
> Host: localhost:8080
> Accept: */*
> Authorization: Basic ZGVmYXVsdDpwYXNzd29yZA==

Response
< HTTP/1.1 200 OK
< Date: Thu, 13 Feb 2014 14:44:14 GMT
```

If the authentication fails, the response looks like this:

```
Request
> GET /genesys/1/storage/id HTTP/1.1
> Host: localhost:8080
```

```
> Accept: */*
> Authorization: Basic ZGVmYBVsdDpwYXNzd29yZA==

Response
< HTTP/1.1 401 Unauthorized
< Date: Thu, 13 Feb 2014 14:47:55 GMT
< WWW-Authenticate: Basic realm="Genesys Application Configuration Needed"
< Content-Length: 0
```

Example:

Configuration Manager is configured for the `service.request-interaction` section using the following basic authentication parameters:

- username = genesys
- password = genesys

Request without credential:

```
POST /genesys/1/service/request-interaction HTTP/1.1
Host: localhost:8080
Accept: */*
Content-Length: 40
Content-Type: application/x-www-form-urlencoded
```

Response with the authentication error:

```
HTTP/1.1 401 Unauthorized
Date: Thu, 13 Mar 2014 07:55:38 GMT
WWW-Authenticate: Basic realm="Genesys Application Configuration Needed"
```

The same request with credential:

```
POST /genesys/1/service/request-interaction HTTP/1.1
Authorization: Basic Z2VuZXN5czpnZW5lc3lz
Host: localhost:8080
Accept: */*
Content-Length: 40
Content-Type: application/x-www-form-urlencoded
```

Response of the request:

```
HTTP/1.1 200 OK
Date: Thu, 13 Mar 2014 07:55:55 GMT
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: application/json;charset=UTF-8
{"_access_code":"152606","_expiration_time":"29","_id":"413-ac85eb82-3e5e-414e-9ed2-08392320f234",
 "_access_number":"6504664630"}
```

DFM Configuration

When using Basic Authentication, you must also update DFM in order to protect the username/

password.

1. In Configuration Manager, locate and open the Application object for your Orchestration Server (ORS).
2. Select the *Options* tab.
3. Add a new section `dfm.<server_name>`.
4. Add the following options and values:
   - `username`, value = `genesys1`
   - `password`, value = `password1`
   - `maxage`, value = `60` (optional)
   - `maxstale`, value = `60` (optional)
5. Repeat Steps 3 and 4 for each server.
6. Save the ORS application object and restart ORS.

> **Important**
>
> If the username/password is changed in GMS sections (`server`, `storage`, `notification`, `service[s]`), the username/password must also be changed in ORS DFM accordingly. For example, if you set username/password in `server` section, you can use the same username/password for all GMS DFM in ORS (if you change the GMS username/password, you must also change it in ORS DFM for GMS). If you set a specific username/password for Service, Storage, or Notification API rather than using the main one (in `server` section), you must also change the settings in each GMS DFM in ORS in order to manage different username/passwords.

Transport Layer Security

Genesys Mobile Services (GMS) supports Transport Layer Security (TLS), which enables cryptographic and trusted communications between Genesys clients and servers.

TLS features to note:

- Upgrade mode for Configuration Server
- No mutual TLS mode where server and client exchange their certificate (only server certificate is checked)

See the *Genesys Security Deployment Guide* for additional information about TLS.

Chat Server Specifics

GMS has no direct connection to Chat Server.

To implement TLS to Chat Server: this is the connection from Web API Server to Chat Server that must be configured using the same TLS option as what is described from direct connection from GMS to Message Server or Stat Server.

In background, for each Chat polling (5s/chat session):

- GMS requests to load-balancer for Chat Server information.
- GMS gets ChatServer host:port, TLS information, and build connection.
- If connection is secured, GMS must be configured with certificate on host or application level (it is not possible on the connection level).

Summary

The following table summarizes the GMS TLS connection support for Genesys servers.

GMS connect to	TLS support	Comment
Configuration Server	Yes	Upgrade mode only.
Message Server	Yes	TLS server port must be enabled.
Statistics Server	See comments.	Not configured at startup, but should work.
Chat Server	Yes	Connection information returned by Web API Server Load-Balancer.

GMS connect to	TLS support	Comment
Orchestration Server	No	An HTTP connection. Not configured at startup (that is, not in the GMS Connection tab). Note: GMS uses HTTPClientFactory, and a TLS option can be set (section gms, option http.ssl_trust_all, value=false, true).
Web API Server	No	An HTTP connection. Not configured at startup (that is, not in the GMS Connection tab). Note: GMS uses HTTPClientFactory, and a TLS option can be set (section gms, option http.ssl_trust_all, value=false, true).

Single Sign-On

> **Important**
>
> This feature requires specific configuration/updates on Genesys Management Framework components (LDAP, IdP, Config Server, and so on).

GMS version 8.5.003.xx and higher enables you to use Single Sign-on (SSO) to access the GMS Service Management UI. This page describes the settings needed to configure GMS to use your existing SSO infrastructure.

Login

Initiates Security Assertion Markup Language (SAML) login procedure.

```
http://<gmshost>:<gmsportport>/genesys/admin/
```

All authenticated Genesys users defined in Configuration Manager can access the GMS Service Management UI. GMS requires a valid user defined in Configuration Manager in order to allow administration tasks. Genesys Config Server must be configured to use external authentication functionality; Config Server users must be defined to use external authentication, pointing to the authentication system (LDAP, and so on).

Logout

Close the browser or remove browser cookies.

Deployment

SSO deployment requires the following steps:

Start

1. Uncomment the SAML parameter in the launcher.xml file.
2. Create keystore.
3. Create server-settings.yaml file and configure the following settings:
    - adminUrl
    - caCertificate

- jksPassword
- encryptionKeyName
- signingKeyName
- identityProviderMetadata: idp-metadata.xml

4. Start GMS.
   - Generate GMS metadata.
   - Update IdP information with GMS metadata.

End

Launcher.xml

Uncomment the following parameter in launcher.xml:

```
<parameter name="saml-settings" displayName="saml-settings" mandatory="false">
  <description><![CDATA[GMS Server SAML init]]></description>
  <valid-description><![CDATA[]]></valid-description>
  <effective-description/>
        <format type="string" default="server-settings.yaml" />
  <validation></validation>
</parameter>
```

Generating Security Keys

To generate a keystore, you can use the keytool utility that is included with Java SDK. To generate a JKS keystore, use the following command:

```
keytool -genkey -keystore keystore.jks -alias <encryptionKeyName>  -keypass <signingKeyName>
-storepass <jksPassword> -dname <distinguished_name>
```

server-settings.yaml

Security Keys

In order to enable SAML, you must specify the following mandatory properties in a general section into server-settings.yaml:

- adminUrl (mandatory) - the URL will be used as unique entity ID in SP metadata.
- caCertificate (mandatory) - a path to a key storage in JKS format containing all necessary keys.
- jksPassword (mandatory) - a password for the key storage specified above.

Example:

```
adminUrl: http://<gmshost>:<gmsportport>/genesys/admin
caCertificate: c:\GMS\keystore.jks
jksPassword: password
```

SAML Settings Section

In order to enable SAML, you must specify the following mandatory properties in the samlSettings section into server-settings.yaml:

- encryptionKeyName
- signingKeyName
- identityProviderMetadata

Example:

```
adminUrl: http://<gmshost>:<gmsportport>/genesys/admin
caCertificate: c:/GMS//keystore.jks
jksPassword: password
samlSettings:
    encryptionKeyName: client
    signingKeyName: client
    identityProviderMetadata: idp-metadata.xml
```

Settings

Name	Mandatory?	Description
encryptionKeyName	Yes	SAML encryption key name. This key must be present in the JKS key storage specified above. This key is used to encrypt SAML message sent to IdP.
signingKeyName	Yes	SAML signing key name. This key must be present in the JKS key storage specified above.
responseSkewTime	No	Sets maximum difference between local time and time of the assertion creation, which still allows messages to be processed. Determines the maximum difference between clocks of the IdP and SP servers. Defaults to 60 seconds.

Note: You can use the same key for signing and encryption.

Identity Provider

Name	Mandatory?	Description
identityProviderMetadata	Yes	Identity Provider XML metadata file path or URL. If the IdP metadata file is exposed by the remote server over HTTP, it is possible to also specify the URL (default request timeout of 5 seconds will be applied). Check

Name	Mandatory?	Description
		the metadata URL of your IdP server.

Generating GMS Metadata

GMS metadata (SP metadata) are available at the following URL:

```
http://<gmshost>:<gmsportport>/genesys/saml/metadata
```

Use this file to update your IdP server.

Starting and Stopping GMS

Overview

You can start and stop Genesys Mobile Services in the following ways:

Objective	Related procedures and actions
Using Solution Control Interface (SCI)	Complete the following procedure: Starting and Stopping GMS Using Solution Control Interface
Using Genesys Administrator	Complete the following procedure: Starting and Stopping GMS Using Genesys Administrator

Starting and Stopping GMS Applications Using Solution Control Interface

Prerequisites

- Genesys Mobile Services is installed.

Start

1. From the Applications view in SCI, select Genesys Mobile Services Application object on the list pane.
2. Click the appropriate button (`Start`, `Stop`, or `Stop Gracefully`) on the toolbar, or select that command from either the Action menu or the shortcut menu. (Right-clicking your Application object displays the shortcut menu.)
3. Click Yes in the confirmation box that appears. Your application obeys the command that you selected.

End

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

Starting and Stopping GMS Applications Using Genesys Administrator

Prerequisites

- Genesys Mobile Services is installed.

Start

1. Log in to Genesys Administrator.
2. On the Provisioning tab, select Environment > Applications.
3. Select the GMS Application.
4. Right-click the application, and then select the appropriate command from the drop-down menu. These three choices apply:
   - `Start applications`
   - `Stop applications`
   - `Stop applications gracefully`

End

Configuring a Delay for Graceful Shutdown

When you select a graceful shutdown, the Solution Control Server (SCS) sends a suspend message to GMS, so that GMS sets itself to OFFLINE. Load Balancer will no longer send requests to this GMS. After a configurable delay, SCS sends a STOP signal, and the GMS will stop. To configure this delay:

Start

1. In Configuration Manager, select the GMS Application.
2. On the *Annex* tab, *sml* section, set the suspending-wait-timeout to the desired value:
   - Default Value: 10
   - Valid Values: 5-600

For additional information about this option, see suspending-wait-timeout in the Common Configuration Options Reference.

End

Troubleshooting

Error when starting cqlsh

If you start bin/cqlsh and receive the following error:

```
Traceback (most recent call last):
```

...

LookupError: unknown encoding:

Change the line in bin/cqlsh from:

```
self.output_codec = codecs.lookup(encoding)
```

to:

```
self.output_codec = codecs.lookup("UTF-8")
```

Endpoint Collision: External Cassandra (on start message) A node with address <IP address> already exists, cancelling join

See the Cassandra documentation for replacing a dead seed node.

Error in GMS logs: WARNING: Failed to check connection: java.net.SocketException: Too many open files

On RHEL Sysems, change the system limits from `1024` to `10240` in `/etc/security/limits.d/90-nproc.conf`, and then start a new shell for these changes to take effect:

```
soft nproc 10240
```

See the Cassandra documentation for recommended settings.

GMS does not install / start on Linux 64

When installing GMS IPs on a 64-bit Linux host, the compatibility packages must be installed on the Linux host. The compatibility packages are available with the OS distribution media.

When several GMS nodes are in a cluster, the following logs are on all GMSs:

```
New node has joined the ring: http://135.39.40.56:8080/genesys
```

Node has been removed from the ring: http://135.39.40.56:8080/genesys

New node has joined the ring: http://135.39.40.56:8080/genesys

Node has been removed from the ring: http://135.39.40.56:8080/genesys

New node has joined the ring: http://135.39.40.56:8080/genesys

Node has been removed from the ring: http://135.39.40.56:8080/genesys

New node has joined the ring: http://135.39.40.56:8080/genesys

Node has been removed from the ring: http://135.39.40.56:8080/genesys

Synchronize the clock of all GMS nodes. On Windows, use following command:

```
net time \\<ComputerName> /set /yes
```

Where `\\<ComputerName>` specifies the name of a server you want to check or with which you want to synchronize.

I cannot update GMS configuration through the GMS Service Management UI, and the following error is in the GMS logs:

```
Unsufficient permissions to perform this operation
com.genesyslab.platform.applicationblocks.com.ConfigServerException: Unsufficient
permissions to perform this operation (ErrorType=CFGNoPermission,
ObjectType=CFGApplication, ObjectProperty=CFGDBID)
```

Make sure that all GMS applications have the user assigned in the *Security* tab, in Configuration Manager.

Migrating GMS 8.1.x to 8.5

GMS Cluster Application Configuration

In order to deploy Genesys Mobile Services (GMS) 8.1.x, a specific application had to be created for each GMS, and then each application had to be configured with the same options.

The GMS 8.5 release introduces a new deployment model; a single Application Cluster object that can be configured for a cluster of GMS's. Each GMS requires its own GMS application in Configuration Manager, which includes a link to the Application Cluster Object in its Connection tab. This enables grouping the common options into a single place. See Creating an Application Object for more information about this deployment model.

GMS Cluster Migration (Cassandra)

For a cluster of three nodes:

1. Stop one of the GMS 8.1.x nodes. This can be either a seed-node, or non seed-node.
2. Create a new GMS application using the GMS 8.5 template (see the GMS 8.5 deployment instructions).
3. Install GMS 8.5.
4. Start GMS 8.5. GMS 8.5 synchronizes with the other GMS 8.1.1 nodes in the cluster.
5. Repeat these steps for all of the GMS 8.1.x nodes. When complete, the cluster is upgraded to GMS 8.5 with data from the GMS 8.1.x cluster.

Service Migration

Starting in GMS 8.1.2, default service options are prefixed with an underscore (_). The following shows an example migration:

8.1.1

[service.match-interaction]

Option	Value
delete_match	true
service	match-interaction
type	builtin

8.5

[service.match-interaction]

Option	Value
_delete_match	true
_service	match-interaction
_type	builtin

Parameters that begin with an underscore (_) are passed to ORS. Anything else is considered user data, and saved in storage. The stored data can be retrieved using the `_data_id` parameter passed in scxml.

Notification

Google has deprecated the C2DM Service, and no new users are being accepted; however, apps using C2DM will continue to work. For more information, see the Android Cloud to Device Messaging Framework documentation.

For customers that want to send notifications to Android devices, using the GCM notification provider is encouraged.

Note that you do not need to prefix options with an underscore.

Chat

Existing 8.1.x Chat service will continue to work with GMS 8.5. Chat service options must be migrated according to the preceding Service Migration section. Genesys recommends migrating to Callback Chat service as defined in the Callback User's Guide.

Port Configuration

Note: This section applies to GMS 8.5.001 and higher.

Because of the port restriction feature introduced in GMS 8.5.001, the GMS HTTP port is no longer configured in the GMS `<GMS_HOME>/launcher.xml` file. To configure HTTP port(s), you must add the HTTP connector into the `<GMS_HOME>/etc/jetty.xml` file:

```
<Call name="addConnector">
      <Arg>
          <New class="org.eclipse.jetty.server.nio.SelectChannelConnector">
            <Set name="host"><Property name="jetty.host" /></Set>
            <Set name="port"><Property name="jetty.port" default="8080"/></Set>
            <Set name="maxIdleTime">300000</Set>
            <Set name="Acceptors">2</Set>
            <Set name="statsOn">false</Set>
            <Set name="confidentialPort">8443</Set>
```

```
            <Set name="lowResourcesConnections">20000</Set>
            <Set name="lowResourcesMaxIdleTime">5000</Set>
          </New>
      </Arg>
</Call>
```

Testing Your Deployment

Testing your deployment is a process that consists of the following tasks:

1. Configuring Dependencies
2. Configuring the GMS Builtin Services
3. Testing the GMS Builtin Services
4. Configuring the ORS-based Services
5. Testing the ORS-based Services

Let's get started with the first task: Configuring Dependencies.

Configuring Dependencies

Before getting started with your GMS services, you must first ensure that external dependencies are configured properly. The following outline will guide you through each dependency.

Note: These procedures assume a multi-tenant configuration and `Tenant = Environment`.

Orchestration Server

1. The GMS DFM configuration files extend standard SCXML with custom tags specific to GMS. You must download the DFM files from the GMS Service Management UI, and then configure the ORS Application in Configuration Manager. To do this, see the procedures Download DFM Files and Deploy DFM Files for details. Make sure that you download the files as follows:
   - `Callback.jsp` to `C:\dfms\callback.txt`
   - `Notification.jsp` to `C:\dfms\notification.txt`
   - `Services.jsp` to `C:\dfms\services.txt`
   - `Storage.jsp` to `C:\dfms\storage.txt`
2. Enable auto conversion of parameter values to proper data types to the SCXML applications. To do this, set the value of the `parse-start-params` configuration option to `false`. See the procedure Setting ORS Option for details.
3. Enable ORS to pull interactions. To do this, set the value of the `mcr-pull-by-this-node` configuration option to `true`. See the procedure Setting ORS Option for details.
4. Set up two connections to Interaction Server to handle multimedia (chat) by creating a second connection to a dummy T-Server application with the same port. See the Orchestration Server Deployment Guide for details.
5. Set up Cassandra for ORS session recovery and multimedia capability. See the Orchestration Server Deployment Guide for details.
6. Set up the Cluster object in the transactions list object if ORS is deployed as a cluster. See the Orchestration Server Deployment Guide for details.
7. Set up the Application object in the transactions list object. See the Orchestration Server Deployment Guide for details.

Universal Routing Server

1. Enable the HTTP interface. This step is required because GMS requests URS to start strategies by HTTP, and GMS receives asynchronous Callbacks from URS by HTTP. To enable the HTTP interface:
   - Add an HTTP listening port with a port ID as `http` and the protocol as `http`. Make a note of this port number as you will need it later when configuring your GMS service.
   - Make sure that there is a `web` section defined in the URS Application Object configuration with its

own `http_port` and `enable_web_access` settings:

- `http_port` = `5580` (or some other port, used internally)
- `enable_web_access` = `true`

2. Deploy URS delay strategies. This step is required because when a service request is received by GMS, the request is sent to ORS for execution. ORS then sends a request to URS to create a virtual interaction and to place it in the specified virtual queue. When an agent is available, URS sends an asynchronous response containing the selected target information to GMS, via a URL specified at the time of creation of the virtual interaction. For samples, you will create a new virtual queue in which to place the interactions, however, for a real-world scenario, the virtual queue must be selected appropriately. To deploy URS delay strategies:
   - Go to Configuration Manager > Switches > SIP_Switch > DN > Virtual Queue. Create a virtual queue `GMS_VQ` with alias `GMS_VQ_SIP_Switch`.
   - Download the URS Strategies and import them into IRD. See the procedure URS Strategy to access the downloadable files and for more details.
3. Enable ORS to pull interactions. To do this:
   - Go to Configuration Manager > Universal Routing Server Application, and set the option `Strategy=ORS`.

SIP Server

1. Enable the answering machine connection, which is required for user-terminated scenarios with Call Progress Detection (CPD) capability. To do this:
   - Go to Configuration Manager > SIP Server Application, and set `TServer/am-detected = connect`.
2. Enable MSML, which is required so SIP Server can communicate with GVP as a Media Server to delegate outbound calls, play treatments, and CPD. To do this, go to Configuration Manager > SIP Server Application, and set the following:
   - `TServer/enable-msml-media-services = true`
   - `TServer/msml-support=true`
   - `TServer/refer-enable=true`

Web API Server

1. GMS uses Web API Server for load balancing Chat Servers. To set this up:
   - Download and install the load balancing servlet. See the procedure Web API Server Configuration to access the downloadable file and for more details.
   - Add an HTTP listening port with a port ID `http` and protocol as `http`. The port number must be the same as the default port number.
   - Add a connection to each of the Chat Servers.
   - Ensure that the load balancer is able to return the Chat Servers: `http://<WebAPIServerHost:WebAPIServerPort>/<gmsoption:chat/chat_load_balancer_url>`

Media Server (GVP)

Note: See the Genesys Voice Platform Deployment Guide for additional details.

1. GMS Callback uses Media Server via SIP Server:
   - To play treatments.
   - For CPD (Call Progress Detection).
   - To make outbound calls.
2. SIP Server talks to Media Server using MSML and requires the following configuration to enable:
   - Go to Configuration Manager > SIP_Switch > DN > VOIP Service > MSML_Service.
   - Make sure that the following options are configured for `MSML_Service` to enable outbound:
     - `make-call-rfc3725-flow=1`
     - `refer-enabled=false`
     - `ring-tone-on-make-call=false`
     - `userdata-map-filter=*`
   - Configure the Routing Point for outbound source DN. To do this, go to Configuration Manager > Switches > SIP_Switch > DN > Routing Point.
     - Create a Routing Point object with name 8999 and alias `8999_SIP_Switch`.

Next Steps

Configure the GMS Builtin Services

Configuring the GMS Builtin Services

Now that all of the dependencies are set up, you can start to configure the service and use the samples to test it. Three samples are available for download, and any one of these samples can be used to validate the deployment. The JavaScript sample comes pre-installed with the GMS Installation Package and is ready to use, so the instructions outlined here will focus on the JavaScript sample. To access the sample:

1. In your browser, open the Service Management UI by accessing either of the following URLs:
   - <GMS Local Host>:8080/genesys/home/login.jsp#/
   - <GMS Local Host>:8080/genesys/home/index.jsp#/
2. Click Lab > Samples.
3. Click the `Scenario` drop-down to display the list of scenarios supported. The first two scenarios are of type `builtin`, which means that GMS handles the service features by itself, although the actual interaction must be handled by an ORS workflow or URS strategy. The other scenarios depend on ORS-based Callback services and require advanced configuration. These instructions on this page will show you how to configure and test the `builtin` scenarios.

Prerequisite

You must have configured the dependencies.

Step 1: Resource Group – Add Access Number

Why:

GMS provides this access number to the user, and the user dials in to this access number.

How:

GMS Service Management UI

Procedure:

1. Go to the GMS Service Management UI > Tools > Resources.
2. Add the access number to the `DNIS` group.

Step 2: GMS Service – Create Service request-interaction

Why:

This service is responsible for receiving the GMS request and providing an access number to the user.

How:

GMS Service Management UI

Procedure:

1. Go to the GMS Service Management UI > Services > Configured Services.
2. Click `Add Service`.
3. Set `Configure Service = request-interaction`.
4. Set `Service Name = request-interaction`.
5. Click Save.

Step 3: GMS Service – Create Service match-interaction

Why:

This service helps to match a voice call with an existing GMS service responsible for providing the access number.

How:

GMS Service Management UI

Procedure:

1. Go to the GMS Service Management UI > Services > Configured Services.
2. Click `Add Service`.
3. Set `Configure Service = match-interaction`.
4. Set `Service Name = match-interaction`.
5. Click Save.

Step 4: GMS Service – Create Service request-chat

Why:

This service is responsible for receiving the GMS request and providing a URL to start the chat interaction.

How:

GMS Service Management UI

Procedure:

1. Go to the GMS Service Management UI > Services > Configured Services.
2. Click `Add Service`.
3. Set `Configure Service = request-chat`.
4. Set `Service Name = request-chat`.
5. Click Save.
6. Set the service property `_chat_endpoint = Environment:gms_builtin` (Note: For single tenant: `Resources:gms_builtin`.)

Step 5: Inbound SCXML Service - Voice

Why:

The inbound service matches the voice call with an existing GMS service. If a matching service is found, the GMS user data is attached to the interaction, and the call is routed to the agent.

How:

- Configuration Manager > Switches > SIP_Switch
- Configuration Manager > Scripts

Procedure:

1. Create a route point associated with the access number configured in the procedure Resource Group Add Access Number.
2. Set Annex > Orchestration section > `application = script:GMSInbound.Voice.GMSMatchBuiltin`.
3. Create an enhanced routing script `GMSInbound.Voice.GMSMatchBuiltin`.
4. Set Annex > Application section > `url = http://<gmshost:gmsport>/genesys/1/document/service_template/callback/src-gen/IPD_Voice_GMSMatch.scxml`.
5. Set Annex > `ApplicationParms/app_find_agent_timeout = 30`.
6. Set Annex > `ApplicationParms/app_match_gms_builtin = true`.
7. Set Annex > `ApplicationParms/app_match_target = <target>` (Example: `Customer_Service@stat_server.GA`).
8. Set Annex > `ApplicationParms/app_no_match_target = <target>` (Example: `All_Standard_Agents@stat_server.GA`).

9. Set Annex > `ApplicationParms/app_require_access_code = false`.
10. Set Annex > `ApplicationParms/app_require_ani = true`.
11. Set Annex > `ApplicationParms/app_treatment_waiting_for_agent = <blank>` (A blank value will force the service to use a packaged music file.).
12. Make sure that MSML capabilities are configured and working to play treatments. This step is required because this service includes play treatments, and has a dependency on Media Server.

Step 6: Inbound SCXML Service - Chat

Why:

This inbound service attaches the GMS user data to the interaction, and routes the interaction to the agent.

How:

- Configuration Manager > Chat Server
- Configuration Manager > Scripts

Procedure:

1. Go to Configuration Manager > Chat Server.
2. Create an end point that was specified in procedure GMS Service Create Service request chat (sub-step 6):
   - `gms_builtin = GMSInbound.Chat.QueueBuiltin`
3. Go to Configuration Manager > Scripts.
4. Create an interaction queue that you just specified, above.
   - Name: `GMSInbound.Chat.QueueBuiltin`
   - Annex > `Orchestration/application = script:GMSInbound.Chat.QueueBuiltin.Routing`
5. Create an interaction queue view.
   - Name: `GMSInbound.Chat.QueueBuiltin.View 1`
   - Annex > `View/Queue = GMSInbound.Chat.QueueBuiltin`
6. Create an Enhanced Routing Object that you just specified, above.
   - Name: `GMSInbound.Chat.QueueBuiltin.Routing`
   - Annex > `Application/url = http://<gms_host>:<gms_port>/genesys/1/document/service_template/callback/src-gen/IPD_Chat_QueueBuiltin.scxml`
   - Annex > `ApplicationParms/app_find_agent_timeout = 30`
   - Annex > `ApplicationParms/app_match_gms_builtin = true`
   - Annex > `ApplicationParms/app_match_target = <target>` (Example:

Customer_Service@Stat_Server.GA)

- Annex > ApplicationParms/app_no_match_target = <target> (Example: All_Standard_Agents@Stat_Server.GA)

Step 7: Interaction Workspace – Display GMS Attached Data

Why:

GMS attaches data to the call prior to routing it to the agent. This attached data is displayed to the agent when the call arrives at the agent desktop (Interaction Workspace), and helps the agent to understand the source of the call, as well as to understand the additional information sent from the customer's device when creating the Callback.

How:

Configuration Manager > Business Attributes

1. Create a new business GMSCaseData attribute of type Interaction Operational Attribute.
2. Create new attribute values:
   - first_name
   - last_name
   - location_lat
   - location_long
   - GMS_Call_Direction
   - GMS_MatchMethod_AccessNumber
   - GMS_MatchMethod_ANI
   - GMS_MatchResult
   - GMS_MatchReason
   - GMS_ServiceName
   - GMS_UserData
3. Set the following Application > InteractionWorkspace options:
   - interaction-workspace > interaction.case-data.format-business-attribute = GMSCaseData
   - interaction-workspace > toast.case-data.format-business-attribute = GMSCaseData

Next Steps

Test the GMS Builtin Services

Testing the GMS Builtin Services

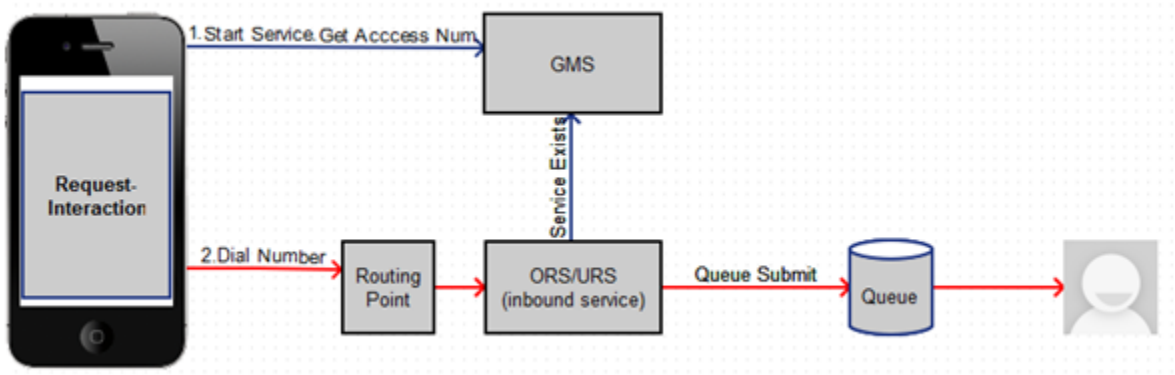
Now that you have configured the Builtin services, it's time to test them.

Prerequisites:

You must have completed the following:

1. Configured the dependencies.
2. Configured the Builtin services.

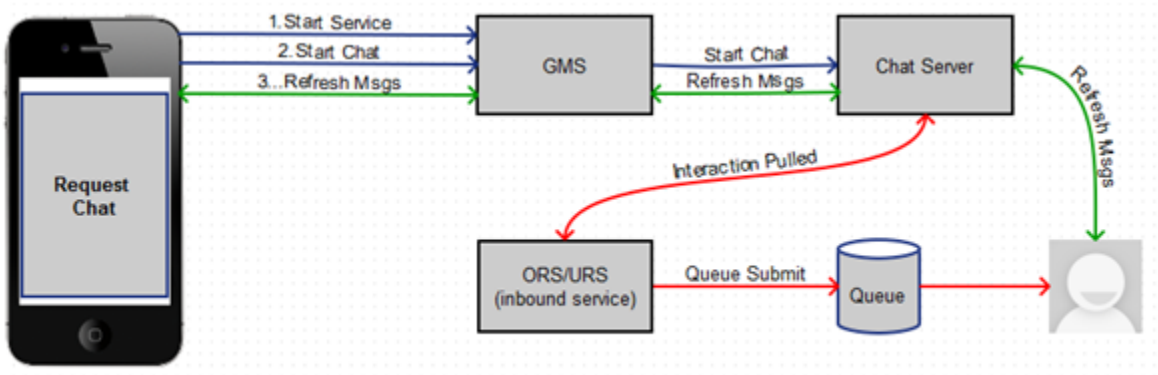
Scenario request-interaction Test Procedure



1. On the Agent Desktop:
   - Log in agent.
   - Make voice ready.
2. Using the Javascript sample: Service Management UI > Lab > Sample:
   - Log in agent and make voice ready.
   - Set `Contact# = <customer phone from which call will be dialed>`
   - Set `Scenario = REQUEST-INTERACTION`
   - Click `Connect`.
   - Dial displayed Number to Call.

3. Expected result:
   - Treatment is played.
   - Call is routed to agent.
   - Toast is displayed with attached data.
   - Call is connected to agent.
   - For a successful GMS call, `GMS_MatchResult = SUCCESS` is displayed in the agent desktop as attached data.

Scenario request-chat Test Procedure



1. Agent Desktop
   - Log in agent.
   - Make chat ready.
2. Using the Javascript sample: Service Management UI > Lab > Sample:
   - Set `Scenario = REQUEST-CHAT`
   - Click `Connect`.
3. Expected result:
   - GMS app displays chat tab.
   - Chat interaction is routed to agent.
   - Toast is displayed with attached data.
   - Chat is connected to agent.
   - GMS app shows agent has joined chat.
   - Agent desktop shows customer has joined chat.
   - On a successful GMS call `GMS_MatchResult = SUCCESS`
   - Customer and agent can now exchange messages.

Next Steps

Configure the ORS-based Services

Configuring the ORS-based Services

Now that the basic scenarios are working, let's get started with the ORS-based advanced scenarios.

Prerequisites

You must have completed the following:

1. Configured the dependencies.
2. Configured the Builtin services.
3. Tested the Builtin services.

Step 1: GMS Service – Samples

Why:

This service is responsible for receiving the GMS request from the sample application.

How:

GMS Service Management UI > Services > Configured Services

Procedure:

1. Click `Add Service`.
2. Set `Configure Service = callback`.
3. Set `Service Name = samples`.
4. Set `Common Default Configuration = samples`.
5. Click Save.
6. Set service property `_urs_server_url = http://<urshost:urshttplisteningport>`.
7. Set `_target = <routetarget>` Example: `Customer_Service@Stat_Server.GA`.
8. Set `_urs_virtual_queue = GMS_VQ_SIP_Switch`.
9. Set `_routing_point = 8999`.
10. Set service property `_chat_endpoint = Environment:gms_ors` (or `Resources:gms_ors` if single tenant).

Step 2: Inbound SCXML Service - Voice

Why:

This inbound service matches the voice call with an existing GMS service. If a matching service is found, it moves the interaction to the GMS service (ORS session), which attaches the GMS User Data, and routes the call to the agent.

How:

- Configuration Manager > Switches > SIP_Switch
- Configuration Manager > Scripts

Procedure:

1. Create a route point associated with the access number configured in the procedure Resource Group Add Access Number.
2. Set Annex > Orchestration section > `application = script:GMSInbound.Voice.GMSMatchORS`.
3. Create an enhanced routing script `GMSInbound.Voice.GMSMatchORS`.
4. Set Annex > Application section > `url = http://<gmshost:gmsport>/genesys/1/document/service_template/callback/src-gen/IPD_Voice_GMSMatch.scxml`
5. Set Annex > `ApplicationParms/app_find_agent_timeout = 30`
6. Set Annex > `ApplicationParms/app_match_gms_builtin = false`
7. Set Annex > `ApplicationParms/app_no_match_target = <target>` (Example: `All_Standard_Agents@stat_server.GA`)
8. Set Annex > `ApplicationParms/app_require_access_code = false`
9. Set Annex > `ApplicationParms/app_require_ani = true`
10. Set Annex > `ApplicationParms/app_treatment_waiting_for_agent = <blank>` (A blank value will force the service to use a packaged music file.)

Step 3: Inbound SCXML Service - Chat

Why:

This inbound service attaches the GMS user data to the interaction, and routes the interaction to the agent.

How:

- Configuration Managaer > Chat Server
- Configuration Manager > Scripts

Procedure:

1. Go to Configuration Manager > Chat Server.
2. Create an end point that was specified in the procedure GMS Service Create Service request chat (sub-step 6):
   - `gms_builtin = GMSInbound.Chat.QueueORS`
3. Go to Configuration Manager > Scripts.
4. Create interaction queue that you just specified, above.
   - Name: `GMSInbound.Chat.QueueORS`
   - Set Annex > Orchestration/application = script:GMSInbound.Chat.QueueORS.Routing
5. Create an interaction queue view.
   - Name: `GMSInbound.Chat.QueueORS.View 1`
   - Set Annex > View/Queue = GMSInbound.Chat.QueueORS
6. Create an Enhanced Routing Object that you just specified, above.
   - Name: `GMSInbound.Chat.QueueORS.Routing`
   - Set Annex > `Application/url = http://<gms_host>:<gms_port>/genesys/1/document/service_template/callback/src-gen/IPD_Chat_QueueORS.scxml`
   - Set Annex > `ApplicationParms/app_find_agent_timeout = 30`
   - Set Annex > `ApplicationParms/app_match_gms_builtin = false`
   - Set Annex > `ApplicationParms/app_no_match_target = <target>` (Example: `All_Standard_Agents@Stat_Server.GA`)

Next Steps

Test the ORS-based Services

Testing the ORS-based Services

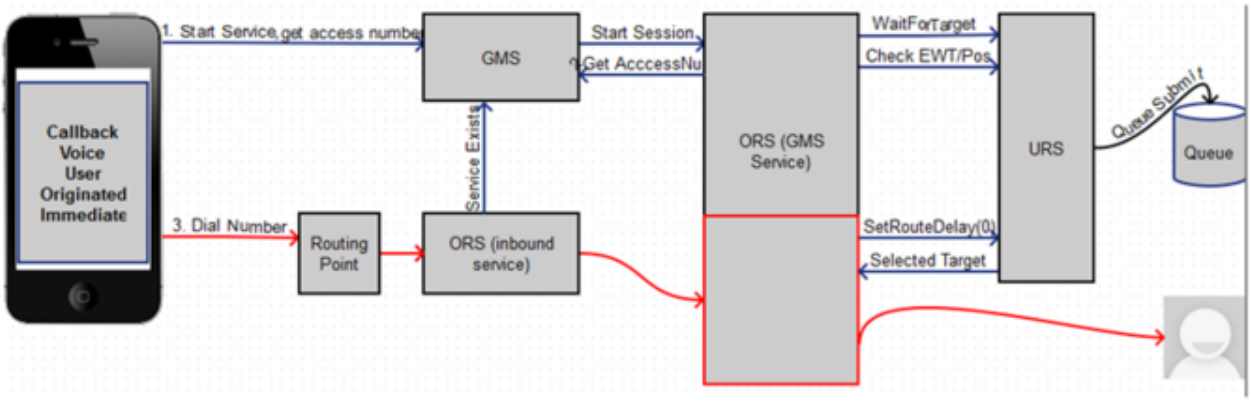
Now that you have configured the ORS-based services, it's time to test them.

Prerequisites

You must have completed the following:

1. Configured the dependencies.
2. Configured the Builtin services.
3. Tested the Builtin services.
4. Configured the ORS-based services.

Scenario VOICE-NOW-USERORIG Test Procedure

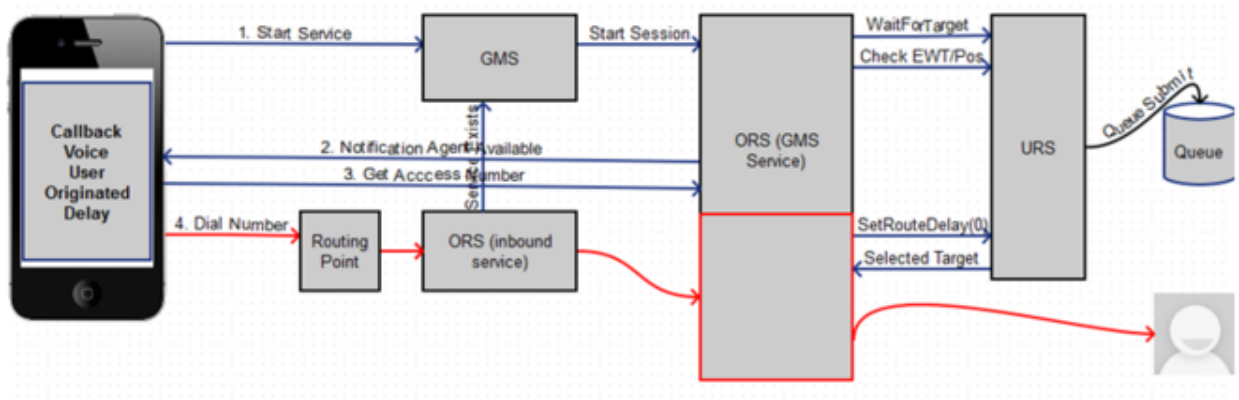


1. Agent Desktop:
   - Log in Agent.
   - Make voice ready.
2. Using JavaScript sample: GMS Service Management UI > Lab > Sample:
   - Set `Contact# = <customer phone from which call will be dialed>`.
   - Set `Scenario = VOICE-NOW-USERORIG`.
   - Click `Connect`.

- Dial displayed Number to Call.

3. Expected result:
   - Treatment is played.
   - Call is routed to Agent.
   - Toast is displayed with attached data.
   - Call is connected to Agent .
   - On a successful GMS call `GMS_MatchResult = SUCCESS`.

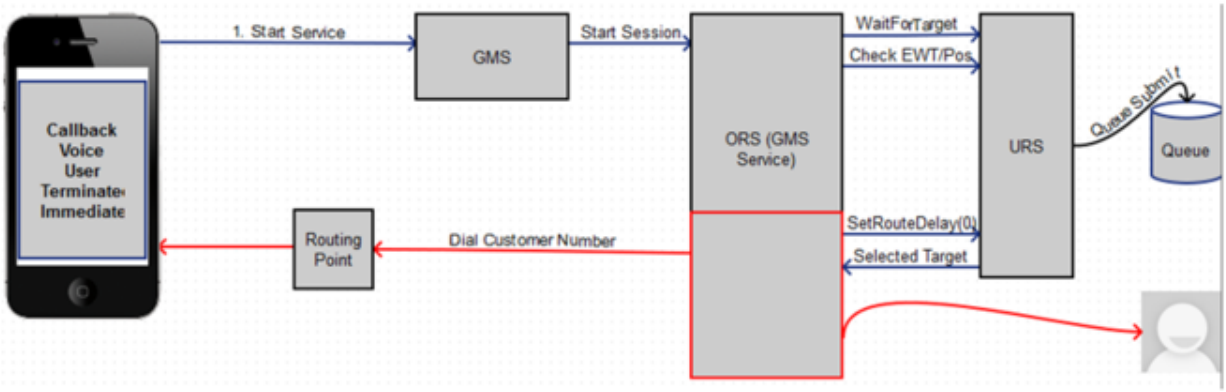
Scenario VOICE-WAIT-USERORIG Test Procedure



1. Agent Desktop:
   - Log in Agent.
   - Make voice ready.
2. Using JavaScript sample: GMS Service Management UI > Lab > Sample:
   - Set `Contact# = <customer phone from which call will be dialed>`.
   - Set `Scenario = VOICE-WAIT-USERORIG`.
   - Click `Connect`.
   - Click `OK` on the message.
   - Wait for Agent Available message.
   - Select `Yes, I am ready to talk`.
   - Dial displayed Number to Call.
3. Expected result:
   - Treatment is played.

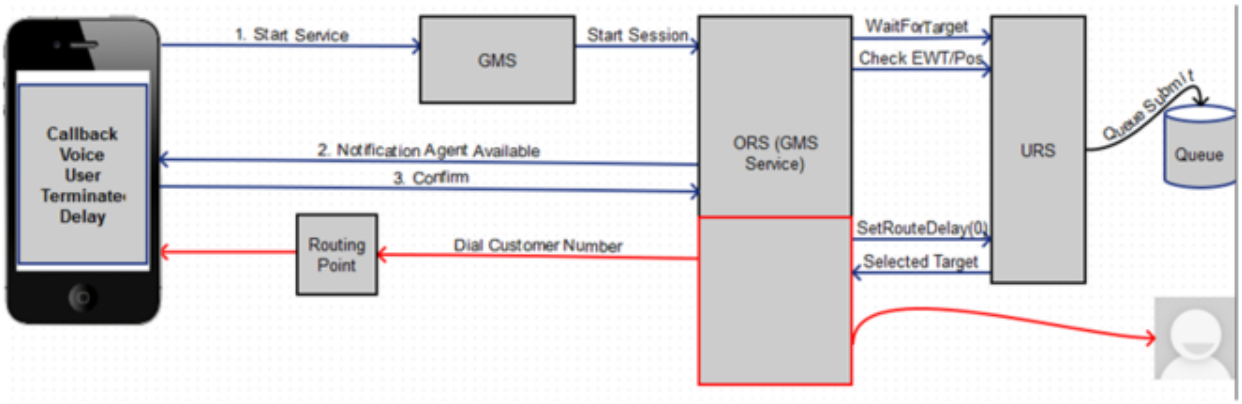
- Call is routed to Agent.
- Toast is displayed with attached data.
- Call is connected to Agent.
- On a successful GMS call `GMS_MatchResult = SUCCESS`.

Scenario VOICE-NOW-USERTERM Test Procedure



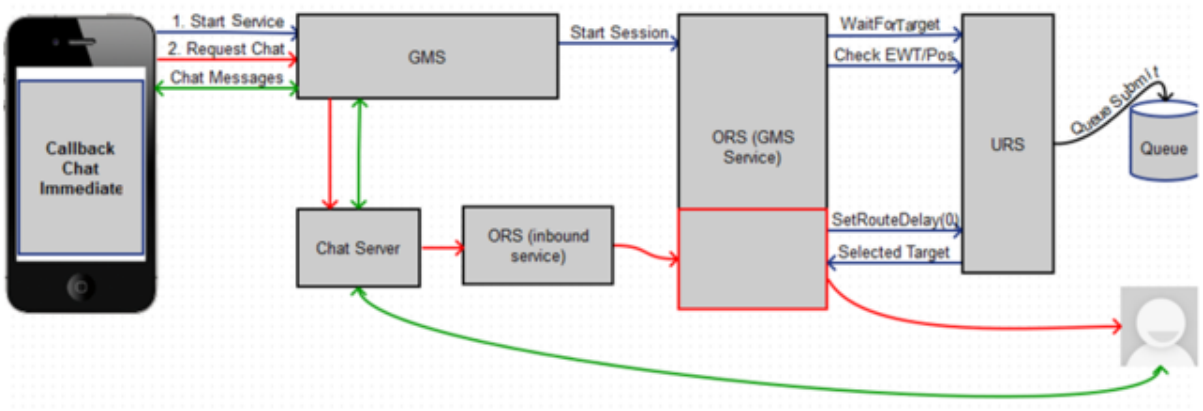
1. Agent Desktop:
   - Log in Agent.
   - Make voice ready.
2. Using Javascript sample: GMS Service Management UI > Lab > Sample:
   - Set `Contact# = <customer phone to which call will be dialed>`.
   - Set `Scenario = VOICE-NOW-USERTERM`.
   - Click `Connect`.
   - Message displays: `You will receive a call shortly`.
   - Click `OK`.
3. Expected result:
   - Call is received.
   - Treatment is played.
   - Call is routed to Agent.
   - Toast is displayed with attached data.
   - Call is connected to Agent.
   - On a successful GMS call `GMS_MatchResult = SUCCESS`.

Scenario VOICE-WAIT-USERTERM Test Procedure



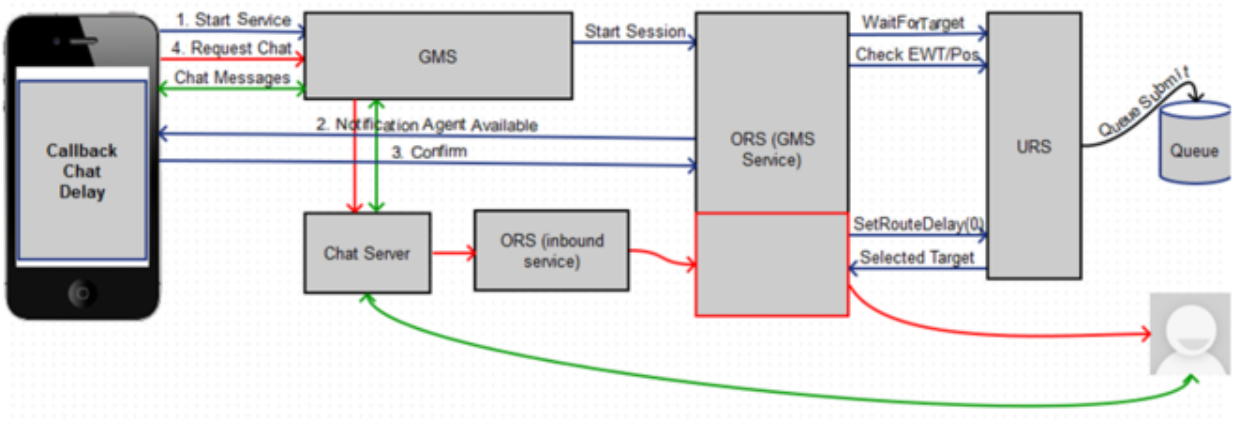
1. Agent Desktop:
   - Log in Agent.
   - Make voice ready.
2. Using Javascript sample: GMS Service Management UI > Lab > Sample:
   - Set `Contact# = <customer phone to which call will be dialed>`.
   - Set `Scenario = VOICE-WAIT-USERTERM`.
   - Click `Connect`.
   - Click `OK` on the message.
   - Wait for Agent available message.
   - Select `Yes, I am ready to talk`.
   - Message displays: `You will receive a call shortly`.
3. Expected result:
   - Call is received.
   - Treatment is played.
   - Call is routed to the Agent.
   - Toast is displayed with attached data.
   - Call is connected to the Agent.
   - On a successful GMS call `GMS_MatchResult = SUCCESS`.

Scenario CHAT-NOW Test Procedure



1. Agent Desktop:
   - Log in Agent.
   - Make chat ready.
2. Using Javascript sample: GMS Service Management UI > Lab > Sample:
   - Set `Scenario = CHAT-NOW`.
   - Click `Connect`.
3. Expected result:
   - GMS app displays chat tab.
   - Chat interaction is routed to the Agent.
   - Toast is displayed with attached data.
   - Chat is connected to the Agent.
   - GMS app shows agent has joined chat.
   - Agent Desktop shows Customer has joined chat.
   - On a successful GMS call `GMS_MatchResult = SUCCESS`.
   - Customer and Agent can now exchange messages.

Scenario CHAT-WAIT Test Procedure



1. Agent Desktop:
   - Log in Agent.
   - Make chat ready.
2. Using Javascript sample: GMS Service Management UI > Lab > Sample:
   - Set `Scenario = CHAT-WAIT`.
   - Click `Connect`.
   - Click OK on the message.
   - Wait for Agent Available message.
   - Select `Yes, I am ready to chat`.
3. Expected result:
   - GMS app displays chat tab.
   - Chat interaction is routed to Agent.
   - Toast is displayed with attached data.
   - Chat is connected to Agent.
   - GMS app shows agent has joined chat.
   - Agent Desktop shows customer has joined chat.
   - On a successful GMS call `GMS_MatchResult = SUCCESS`.
   - Customer and Agent can now exchange messages.

What's Next?

Congratulations - you have successfully tested your GMS deployment! You can now go ahead and configure additional Callback services as needed. Note that you can quickly configure a Callback service to one of the above scenarios by selecting the appropriate default configuration after you add a Callback service.