Genesys Mobile Services Deployment Guide

Genesys Mobile Engagement 8.1.2

12/29/2021

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:

- 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 — Provides a reference of the configuration options available for Genesys Mobile Services.
- 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.

> **Important**
>
> You can find help about the Service Management UI in the Service Management UI Help.

Planning Your Deployment

For the 8.1.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

- 4 GB RAM, or greater

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.

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 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)	8.1.3x	Optional, installed and running, with an HTTP port enabled in the related Application object.
Universal Routing Server (URS)	8.1.300.19	Mandatory, required for the GMS services.
Interaction Routing Designer (IRD)	8.1.300.22	Mandatory, required for strategies running on URS.
SIP Server	8.1.100.67	Mandatory, required for route point calls and incoming calls.

Genesys Component	Minimum Version Required	Comments
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	8.1.410.33	Used for Callback services, in order to play treatments and use Call Progress Detection (CPD) for outbound calls.

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?

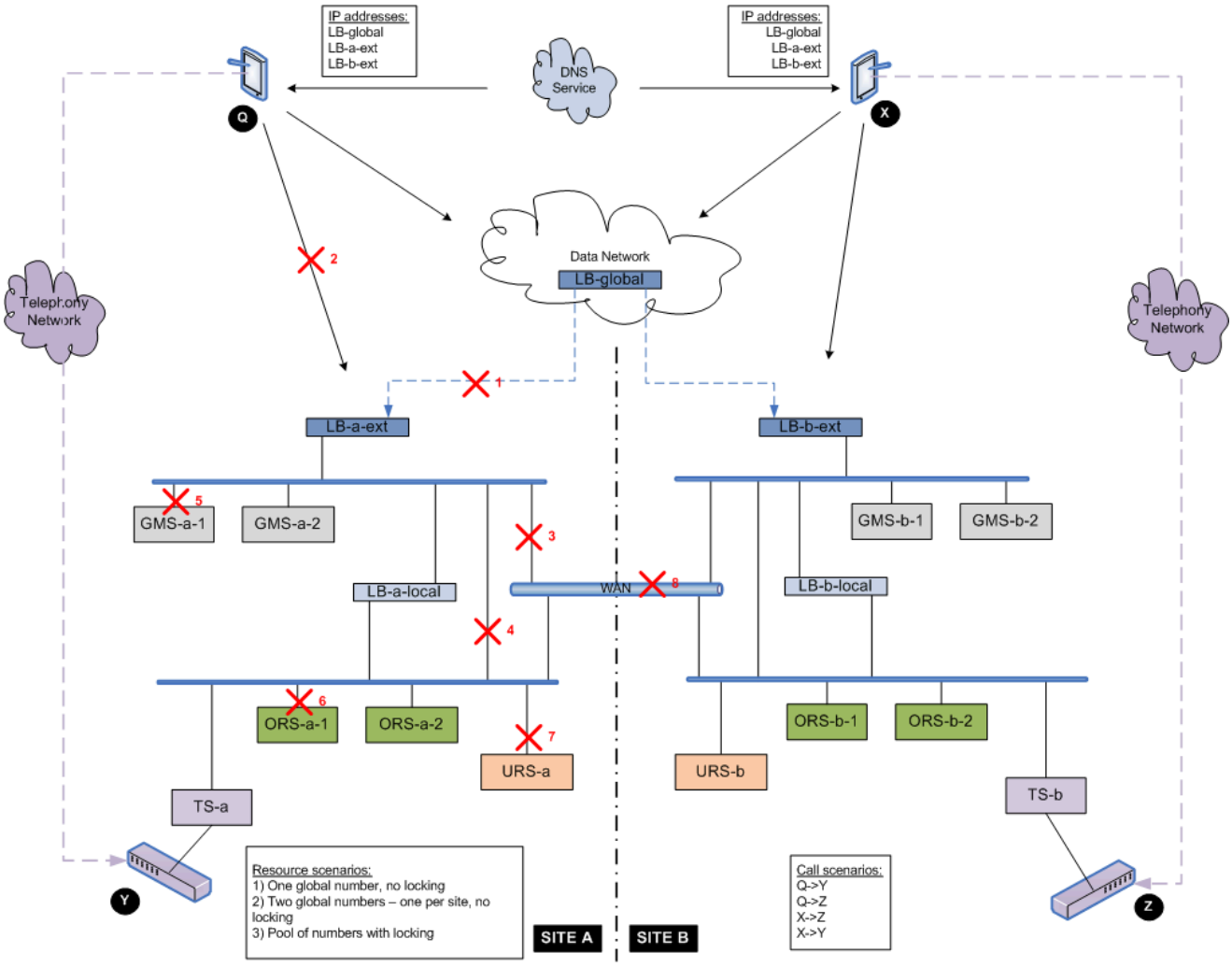
The general recommendation for user originated scenarios is to deploy an independent GMS cluster per site, and control call routing by ensuring that no intersecting pools of incoming phone numbers are configured. This way, if the client is requesting an access number through the data API call, it will 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_812.apd** - This template is for deploying all GMS's into the same cluster. The Cluster Application will contain shared configuration for GMS nodes.
- **GMS_812.apd** - This template is used to deploy GMS with default options. For a single node deployment, the GMS ApplicationCluster_812.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 8.1 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_812.apd
   - \Templates\GMS_812.apd.
5. Select the ApplicationCluster_812.apd file and click *Open*.
6. In the Properties dialog box, click *OK*.
7. Repeat these steps to also import the GMS_812.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_812). 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_812.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_812). 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_812.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_812_node1). Do not uses 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. 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_812.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_812). 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.
- Click *Permissions* and then select *Environment\Administrators.*
- In the *Log On As* group, select *This Account* and set *GMS Admin* for the account.

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

End

Next Steps: Configure ORS and Deploy DFM files.

ORS Configuration

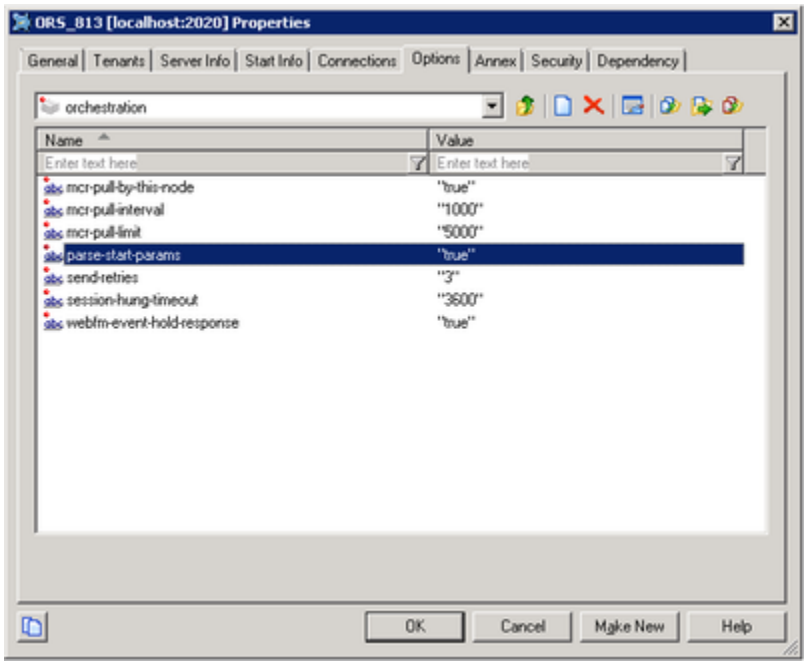
This page details the ORS configuration steps required before you can use your Genesys Mobile Services installation.

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 *true.*
7. 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. 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.

Note: You must restart Orchestration Server after deploying DFM files for the changes to take effect.

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 *dfm* section.
6. Create and configure one option for each DFM, using the option value to specify the file path. Details are provided in the table below.
7. Click *OK* to save your changes.
8. Restart Orchestration Server (ORS). ORS reads the DFM configuration on startup.

Important: At least one GMS node must be up and running in order for ORS to start, and to successfully read the DFM file. If there is only one GMS node in your deployment and it is not running, ORS will **not** start. In the case of multiple GMS's, point the DFM links to a load balancer.

End

List of DFM Options for Orchestration Server

Service	Option Name	Option Value
Storage	gsgStorage	http://<gms_host>:<gms_port>/genesys/dfm/storage.jsp
Notification	gsgNotification	http://<gms_host>:<gms_port>/genesys/dfm/notification.jsp
Genesys Mobile-Based Services	gsgBasedServices	http://<gms_host>:<gms_port>/genesys/dfm/services.jsp
Callback	gsgCallback	http://<gms_host>:<gms_port>/genesys/dfm/callback.jsp

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.

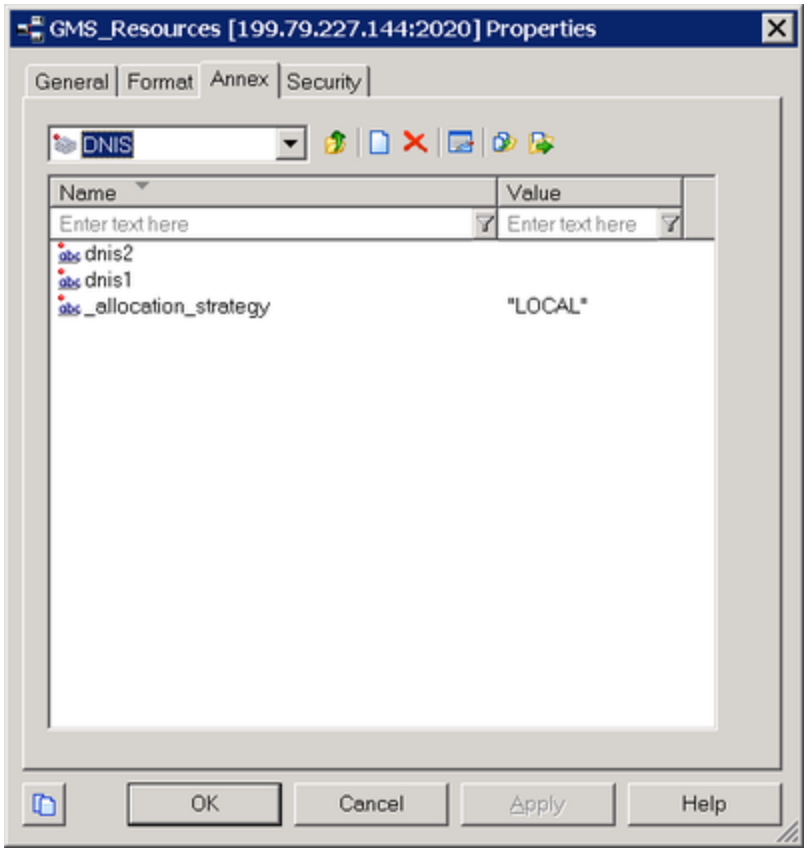
Note: Be sure that you have the *Show Annex tab in object properties* option selected from the *View* > *Options* menu before starting this procedure.

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.
7. Add and set an allocation strategy option for this group.

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

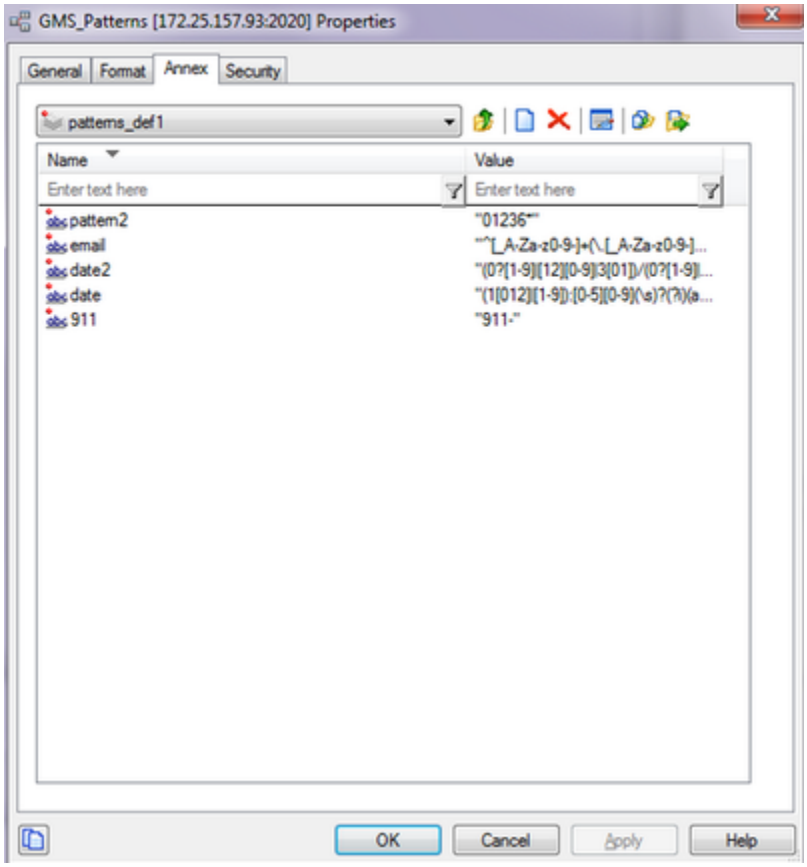
1. Start Configuration Manager.
2. Under the tenant you are working with, open the Transactions folder.
3. Right-click and select *New* > *Transaction.*
4. On the *General* tab, configure the following fields:
   - Name—This name must match the *patterns* > *pattern_list_name* option value from your 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.

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

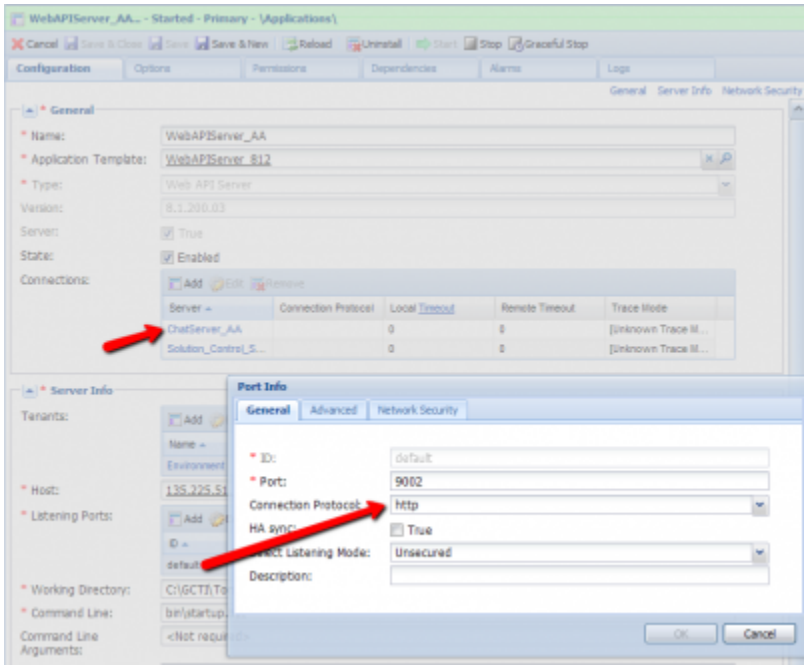
1. Open Genesys Administrator in a web browser.
2. Locate and view the Genesys Mobile Services Application object you previously created and configured.
3. Under the *General* section of the *Configuration* tab, add a connection to the Web Server API Application object that will be used with your Genesys 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
ixn_server_submit_queue	true	default	Queue to which the chat interaction placed. "default" implies, use the default queue specified in the Chat server options->endpoint:1. Any value specified here should be defined in the Chat server options->endpoints:1.

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.

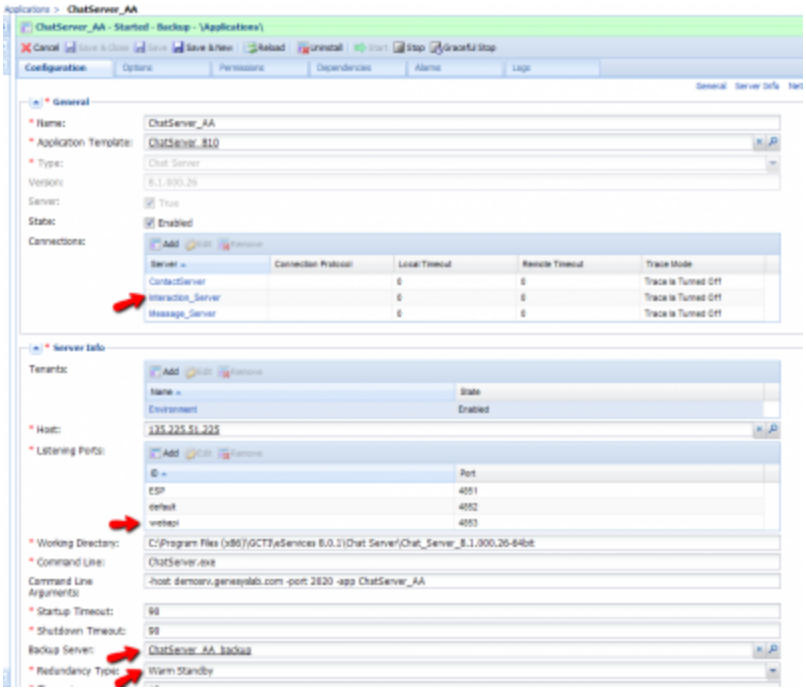
Use the following procedure to download a required chat-related file, and to update the Web API Server Application object that is being used by your GMS deployment:

Start

1. Download the `ChatLBServerInfo.jsp` file, and then add the file into the Web API Server directory.
   - 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.

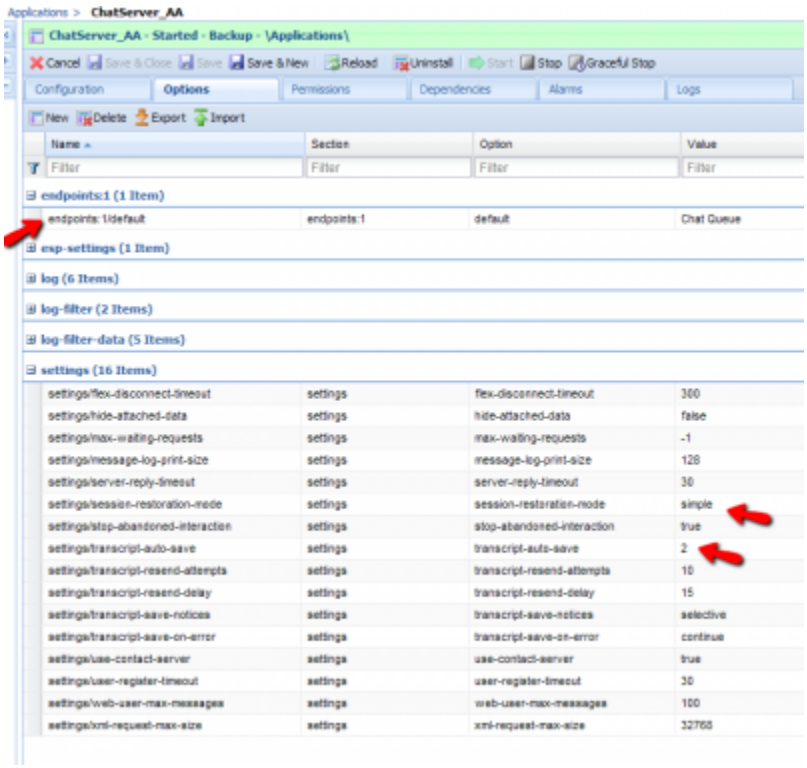
The detailed steps are provided below:

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
Security and Access Control	An outline of key security concerns and information about providing access control.
Load Balancer Configuration Examples	Recommendations for load balancing.
Configuring and Starting a GMS Cluster	Describes the process for initializing a GMS cluster.
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).
Transport Layer Security	Describes Transport Layer Security (TLS), which enables cryptographic and trusted communications between Genesys clients and servers.

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:

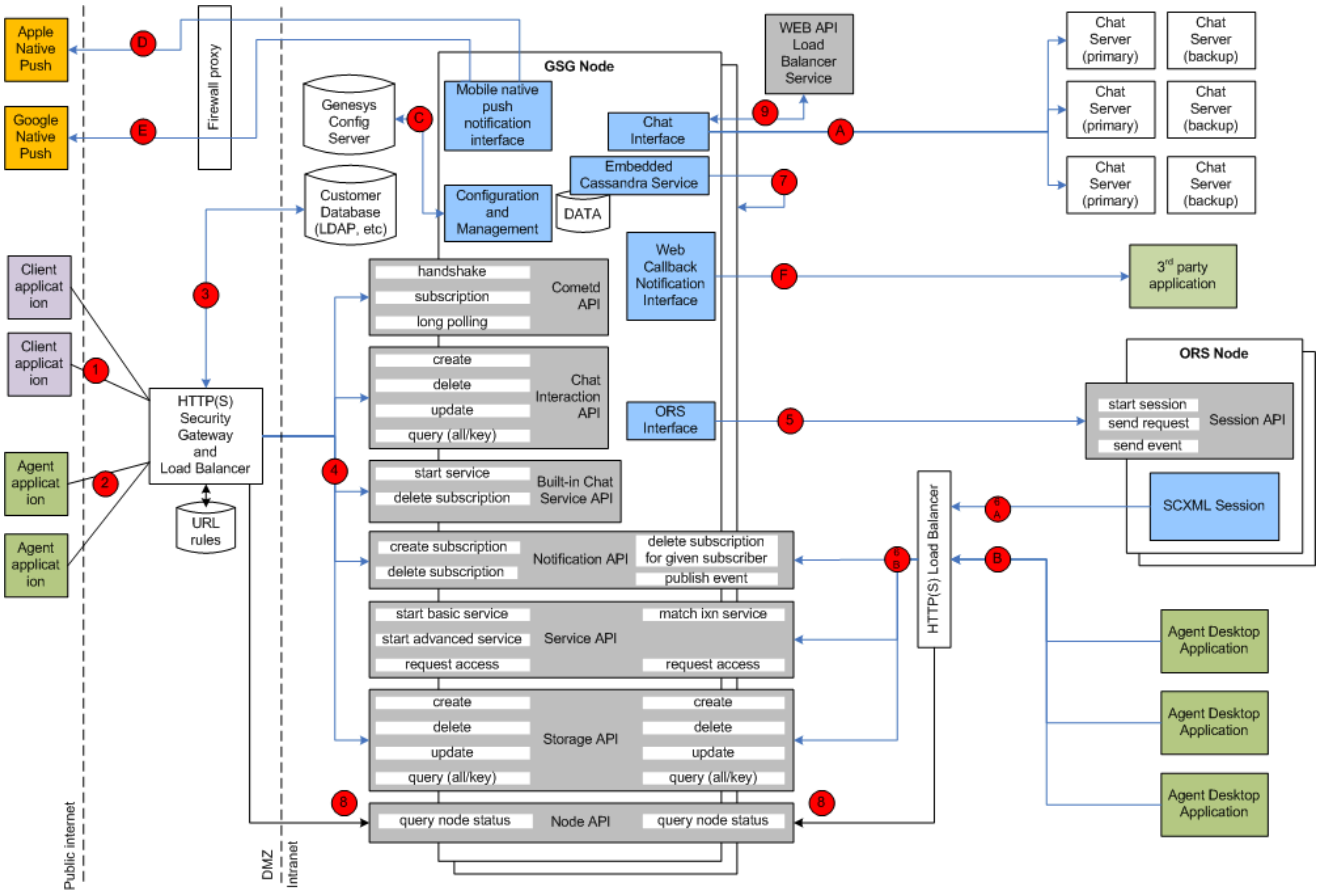
- client authentication (optional)
- TCP port and URL access control
- load balancing functionality, distributing the load between multiple Genesys Mobile Services nodes responsible for processing API requests
- HTTP connection encryption (SSL)

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 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 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.
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*nch.xml: *<parameter name="http_port" displayName="Jetty 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. See *Applications->Genesys Mobile Services->Connections->ORS(1..n)->Server Info->Host and Listening Ports->http*.	
Security and Load Balancing Gateway in Front of Genesys Mobile Services Nodes	(6A)	ORS node running SCXML session	Invoking Genesys Mobile Services APIs: Service (match-interaction), Notification	HTTP	Deployment specific. See 3rd party documentation.	Hard coded as part of the Genesys Mobile Services API URL inside SCXML	Security proxy should be configured to block access to the API URLs according

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
			and Storage APIs.			session. Could be configured in Configuration Manager and read be SCXML session	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>/launcher.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 and load balancing gateway can also be configured with the help of

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							Genesys Professional Services.
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 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.	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 for synchronization between Genesys Mobile Services nodes and client access should only be allowed from Genesys Mobile Services

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							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.
Genesys WEB API Server Node	(9)	Genesys Mobile Services node	Chat server load balancing and high availability API	HTTP	Default port: 9002. Configured in Configuration Manager. See GMS->Connections->WEB API Server->Server Info->Listening Ports->default port	Read dynamically from Configuration Manager configuration: GMS->Connections->WEB API Server. See also GMS->Options->chat/ chat_load_balancer_url_path in Configuration Manager.	No special security is required for this connection. Read only operation.
Genesys	(A)	Genesys	FlexChat	TCP/IP, TLS	Default	Read	Enable TLS

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
Chat Server - N+1 Primary/ Backup Pairs		Mobile Services node	protocol	encrypted if required	port: 4856. Configured in Configuration Manager. See *GMS->Connections->WEB API Server->Connections->Chat Server (primary)->Server Info->Listening Ports->webapi port*.	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.	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. Could be configured in Configuration Manager and read dynamically by desktop application.	Security proxy should be configured to block access to the API URLs according to the access rules below.
Genesys Configuration Server	(C)	Genesys Mobile Services node	Reading Genesys Mobile Services node configuration and	TCP/IP protected by TLS is required	Default: 2020. Configured in Configuration Manager configuration	Configuration is in *<Genesys Mobile Services installation directory>/startup.bat*	Enable TLS encryption if a higher level of protection is required.

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
			reacting on changes		file.	parameters **-app', '-host' and '-port**. See *launcher.xml* in the same directory for description of the parameters.	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 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	Google require SSL protected connection with client side certificates. Make sure your firewall allows

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
						Messaging	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}
{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*

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.

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>
```

Next Steps

- Configure load balancing for your server.

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 and Starting a GMS Cluster

Prerequisite

- GMS version 8.1.2x
- Red Hat Linux version 5.0 (32 bit and 64 bit), 4Gb RAM or Higher
- 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:

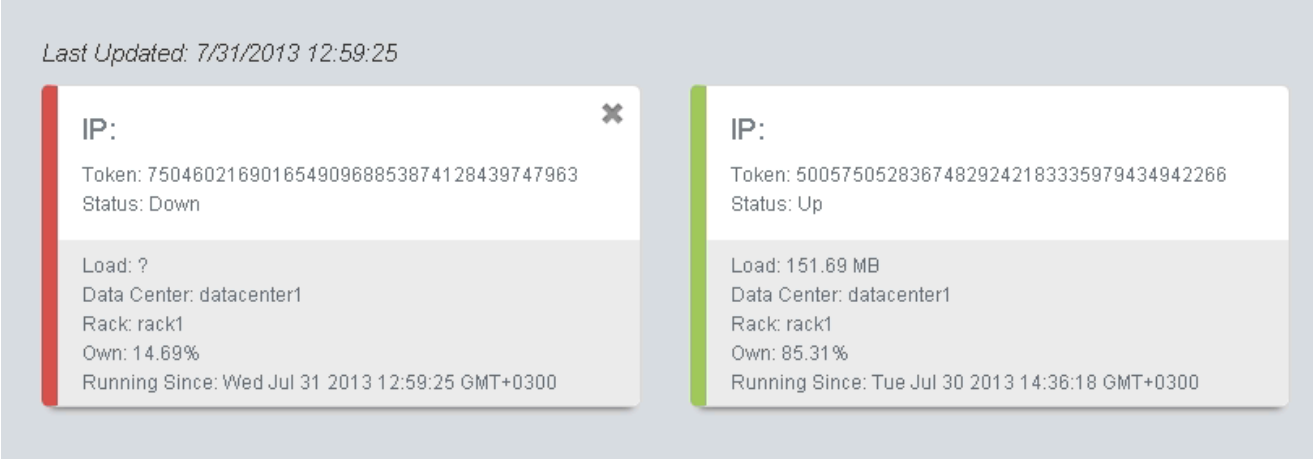
```
global
    daemon
    maxconn 256
defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms
frontend http-in
    bind *:8080
    default_backend cluster_gms
listen admin
    bind *:9090
    stats enable
backend cluster_gms
        balance roundrobin # 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
```

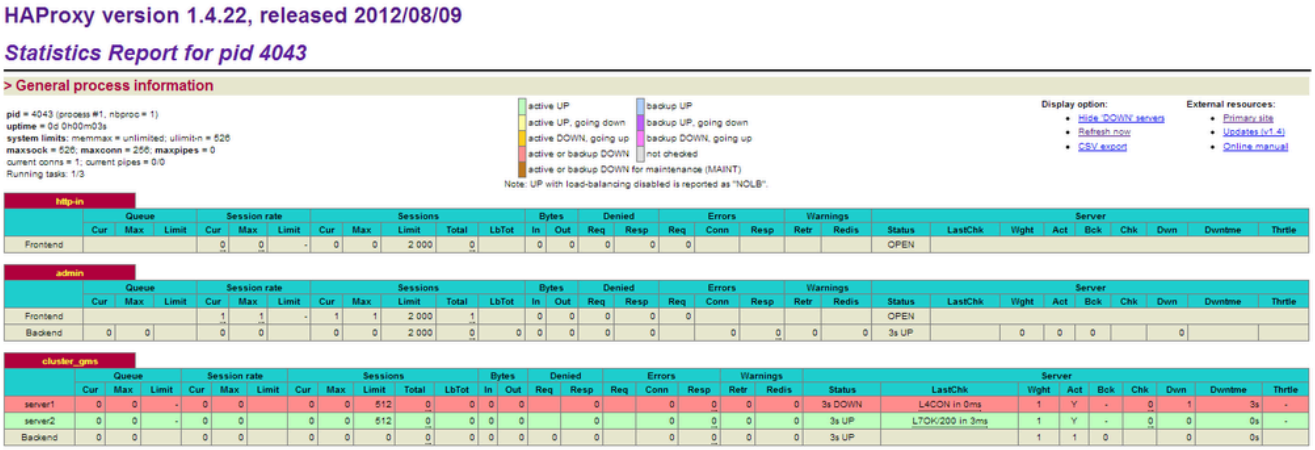
GMS Service Management UI

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.

Mobile Push Notifications

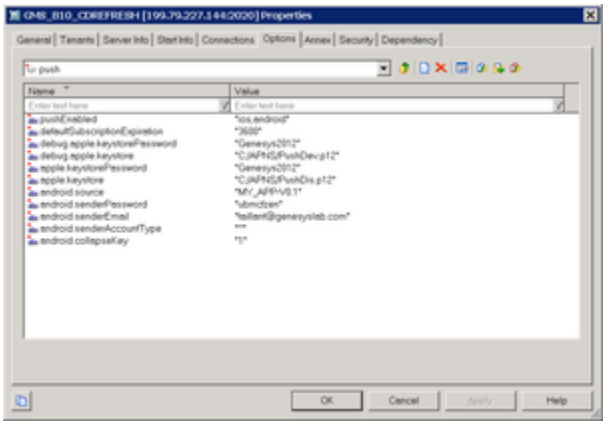
Native Push (Notification Service)

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.

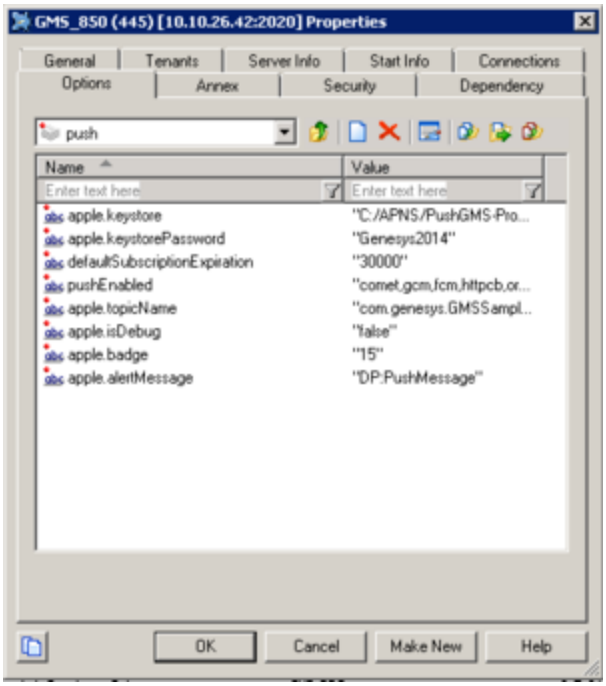
Some services send native push messages to the mobile device. For this to work, both general and device-specific settings need to be configured correctly in the *push* section of your Genesys Mobile Services Application object.

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

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.

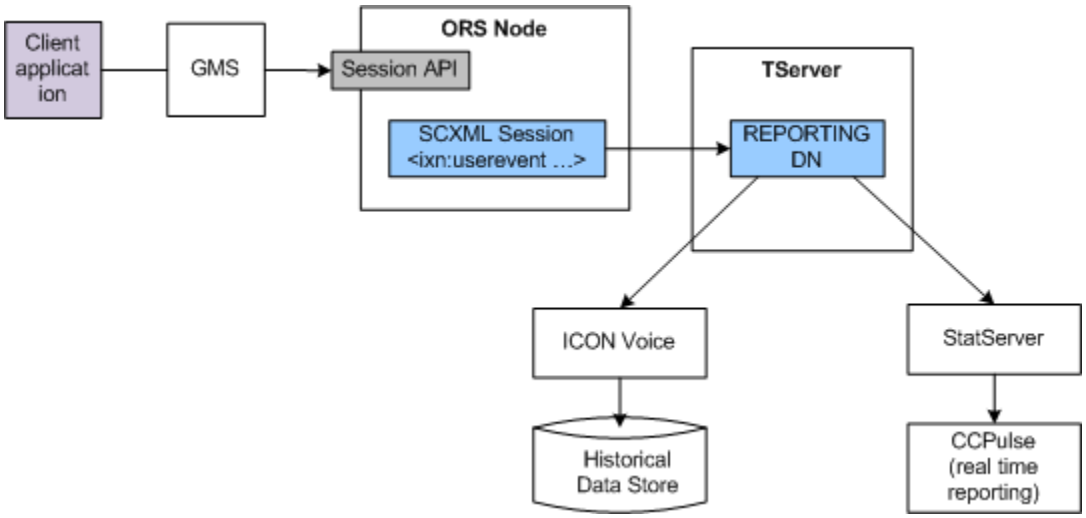
Custom Reporting

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

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).

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

gms Section

Changes take effect: Immediately.

Option name	Option type	Default value	Restriction on value	Description
http.connection_timeout	Integer	10	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.

Option name	Option type	Default value	Restriction on value	Description
http.socket_timeout	Integer	10	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	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	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	Max Range (0-65535)	All http client requests from gms to other servers will use a client socket port from 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.

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					
defaultSubscriptionExpiration	general	Integer	optional	any Integer>=30	Default subscription expiration (in seconds). If not set or assigned an incorrect value, the default value (30) will be used.
pushEnabled	provider	Collection<String>	mandatory		A comma-delimited list of strings that describe the enabled push types. Currently, only following push types are supported: **android** and/or **ios** and/or **httpcb** and/or **orscb**. Any other push type will be ignored. If an option value is not set then it will be handled

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					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 log entries.
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* dictionary (iOS-

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					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 be empty	Specified when initializing a

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					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	C2DM 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 become active

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					(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 returned the

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					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 **pushEnalbled** 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.

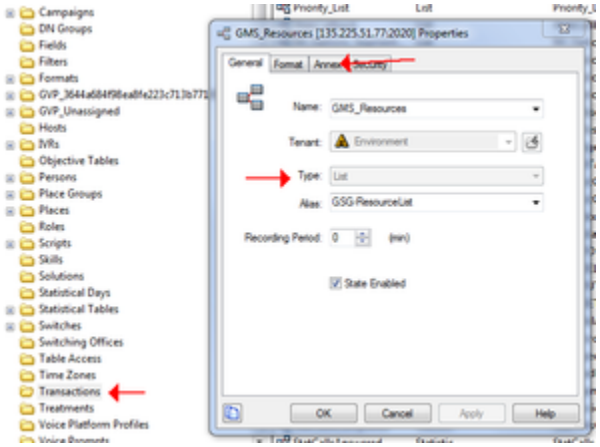
Option name	Option type	Default value	Restriction on value	Description
resources_list_name	String	GMS_Resources	Mandatory	Name of the Strategy

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
```

server Section

Changes take effect: Immediately.

Option name	Option type	Default value	Restriction on value	Description
node_id	Integer		Mandatory, two-digit number	Specifies a two-digit number that should be unique in the Genesys Mobile Services deployment. It is used in the generation of DTMF access tokens.

Option name	Option type	Default value	Restriction on value	Description
dateFormat	String		Optional	The string used to format dates. Syntax of the string should meet the expectations of java class java.text.SimpleDateFormat. See http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html 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's the value which 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	Server port listened for Rest API calls.
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

Option name	Option type	Default value	Restriction on value	Description
				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 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

Option name	Option type	Default value	Restriction on value	Description
				events. Example: `http://<host>:<port>`
_booking_expiration_time	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.

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.

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

Creating GMS Related 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 the web_port option is not equal to the jetty HTTP port.

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

Troubleshooting

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.