Genesys Mobile Services Deployment Guide

Genesys Mobile Engagement 8.1.1

12/30/2021

Table of Contents

Deploying Genesys Mobile Services in Your Environment

This deployment guide can be used to install Genesys Mobile Services on your system, configure basic settings, and verify the installation. It includes chapters with the following information:

- New In This Release — An overview of new features and improvements included with each release of Genesys Mobile Services.
- Planning information — Details related to planning and preparation for your Genesys Mobile Services installation, including prerequisites and links to related information. Genesys recommends reading this 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 on your computer.
- Configuration guidelines — An overview of the configuration options you will use to get Genesys Mobile Services running in your environment.

Next Steps

After you have successfully installed Genesys Mobile Services, you might want to do the following:

- Download the latest version of the Release Note (using links on the Genesys Mobile Services Product Page) to see the most recent news and updates about this product.
- Read the Genesys Mobile Services API Reference for detailed information about Genesys Mobile Services.
- Check the Genesys Mobile Services Developer's Guide to find details about samples that are distributed with your Genesys Mobile Services deployment.

New In This Release

Check out the new features that have been added in the latest releases of Genesys Mobile Services.

New in Release 8.1.1

Release 8.1.100.28

- Improved Chat Support - Genesys Mobile Services now provides the ability to create a chat interaction on any service.
- Internal Poke Message - Genesys Mobile Services now provides the ability to create an internal poke message to be sent via push notification by an agent to the mobile device.
- Resource Booking Timeout - Genesys Mobile Services now provides the ability to control the resource booking timeout.

Release 8.1.100.14

- CometD - Genesys Mobile Services now provides HTTP-based events using CometD.
- Chat Support - Genesys Mobile Services now includes a Chat API, which customer-facing applications can use to create and manage a chat session associated with contact center-related services when used with Genesys Chat Server.
- Improved OS Support - Genesys Mobile Services now provides support for the following operating systems:
  - Red Hat Enterprise Linux 64-bit
  - Windows Server 2008 64-bit
- Google Cloud Messaging - The push notification service now supports Google Cloud Messaging (GCM).
- Push Notification Localization - Push notification now supports localized messages. The language specified during subscription is used to retrieve content from a custom XML file that is specified by the *localizationFileLocation* option.
- Improved Genesys Mobile Services API Security - Includes the ability to hide sensitive data in logs, and take advantage of client-side port definition when connecting to Genesys Servers.
- Counters and Monitoring Support - This release provides support for displaying counters and monitoring Genesys Mobile Services nodes.

New in Release 8.1.0

Documentation Improvements

- Updated deployment information.
  - Deployment Prequisites - At least 4 GB RAM is required for this product to run properly.
  - Configuring - Updated to better reflect changes to Genesys Mobile Services.
- Service name updates have been fixed throughout documentation, including in call flow diagrams.
- Genesys Mobile Services Developer's Guide
  - Additional information provided about Genesys Mobile Services samples.
  - Samples now available to download from Sample Resources.

Release 8.1.000.30

- Enhanced OS Support:
  - Windows Server 2003, 32-bit
  - Windows Server 2008, 32-bit and 64-bit compatibility
- API Updates—API naming conventions have been adjusted to be more intuitive and consistent. (See table below for a list of service names that have been updated with this release.)
- Enhancements to Push Notification Services for iOS and Android—This release includes support for the sending different types of message content (such as images or audio) and adds OS-specific attributes that can be associated and passed with the notification message.
- Mobile Client Samples—Code samples are now available from the Genesys Mobile Services Developer's Guide.

Service Name Updates for Release 8.1.000.30

Build .23	Build .30
a2c-basic	request-interaction
a2c-match	match-interaction
a2c-reserve-accessinfo	request-access
a2c-advanced-sync	request-inbound-immediate
a2c-advanced-async	request-inbound-delay
a2c-advanced-poll	request-inbound-poll
a2c-advanced-async-location	request-inbound-delay-location
a2c-advanced-sync-enterprise-initiated	request-outbound-immediate
a2c-advanced-async-enterprise-initiated	request-outbound-delay
a2c-advanced-poll-enterprise-initiated	request-outbound-poll

Release 8.1.000.23

- Connect using Mobile Apps—Genesys Mobile Services allows mobile customers to connect with your contact center, automatically associating mobile contextual data with the phone call in order to improve the customer experience. Call timing can optionally be postponed until an agent is available to attend the call.
- Informed Access Number Selection—The phone number used to call your contact center can be chosen for each individual inbound call, based on mobile contextual data, such as customer location.
- Customer Location—Part of the mobile contextual data is customer location. For instance, customer location can be displayed on a map for the agent attending the call.

Planning

Introduction

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 Information 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.

Deployment Prerequisites

To work with Genesys Mobile Services, 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

- Windows

  - Windows Server 2003 (32-bit)
  - Windows Server 2008 (32-bit, 64-bit compatibility mode)
- Linux — Red Hat Enterprise Linux AS 5

Genesys Environment

- Orchestration Server (release 8.1.200.20 or later) installed and running, with an HTTP port enabled in the related Application object.

Note: For a full list of supported operating systems and databases, see the Supported Operating Environment Reference Manual.

Related Resources

- Genesys Mobile Services API Reference
- Genesys Mobile Services Developer's Guide
- Orchestration Server documentation

Installing

Installation Overview

Note: Before you begin with the installation process, be sure that your environment meets the minimum requirements specificed on Planning.

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

1. Create a Genesys Mobile Services configuration object
2. Install Genesys Services Gateway

You will repeat each of these tasks for each Genesys Mobile Services instance (or node) that you want to create in your Genesys environment.

Detailed steps for each task are available on this page.

Once the installation is complete, additional configuration is required before your Genesys Mobile Services deployment is ready to use. For more information refer to Next Steps.

Creating an Application Object in Configuration Manager

Before installing Genesys Mobile Services, use the template included with your installation package to create a Application object in Configuration Manager and provide some basic configuration details.

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.

Note: Configuration objects can also be created and configured in Genesys Administrator. Refer to the Framework 8.1 Genesys Administrator Help for information.

Start of Procedure

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 of Procedure

Importing the Genesys Mobile Services Application Template

Purpose: To import the Application Template associated with Genesys Mobile Services using Configuration Manager.

Start of Procedure

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. This template is included with your IP as *\Templates\GMS_810.apd*.
5. Select this file and click *Open*.
6. In the Properties dialog box, click *OK*.

End of Procedure

Creating/Configuring the Genesys Mobile Services Application Object

Purpose: To create and configure an Application object for Genesys Mobile Services.

Prerequisites: Import the Application Template.

Start of Procedure

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_810 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 (if desired).
   Note: The Application name should not contain spaces.
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 select the port and click *Edit Port* or you can configure a new port by clicking *Add Port*. Either action brings up 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. These fields are required to save the Application object, however.
10. Select the Connections tab and specify all the servers to which Genesys Mobile Services must connect:
    - Orchestration Server (ORS)
11. Click OK to save your configured Application object.

End of Procedure

Further configuration can take place after your Genesys Mobile Services installation is complete. For details about additional configuration options, refer to Configuring.

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 the Genesys Mobile Services in your environment.

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

Start of Procedure

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. Under *User*, enter the user name and password for logging on to Configuration Server.
5. Click *Next*. The Select Application screen appears.
6. Select the 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.

7. Click *Next*. The *Choose Destination Location* screen appears.
8. Under *Destination Folder*, keep the default value or browse for the desired installation location.
9. Click *Next*. The *Server Configuration Parameters* screen appears.
10. Specify whether this instance of Genesys Mobile Services is going to be the primary or backup server. 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. If you make this instance a Backup Server then you also need to specify the IP Address of the primary Genesys Mobile Services server before continuing.
11. Click *Next*. The Ready to Install screen appears.
12. Click *Install*. The Genesys Installation Wizard indicates it is performing the requested operation for Orchestration Server. When through, the Installation Complete screen appears.
13. Click *Finish* to complete your installation of Genesys Mobile Services.

End of Procedure

Next Steps

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

- Configure Genesys Mobile Services options that determines how services behave and how this product works with Orchestration Server. The configuration details steps provided here are required for Genesys Mobile Services to run correctly.
- Review the sample applications included with this installation to help understand how you can leverage Genesys Mobile Services in your own development.

You may also want to consult the Genesys Mobile Services API Reference to get a better idea of what services are available to you.

Configuring

Pages in this section detail configuration steps that must be followed before your can use your Genesys Mobile Services installation. Before your installation is ready to use, you should read each of the following pages and make the necessary updates to your Genesys environment.

Page	Summary
Basic Configuration	Basic configuration of services provided by your Genesys Mobile Services solution.
Chat Support	Configuration details for setting up and using the Chat API.
Configuration Options	Detailed look at the configuration options available in your Genesys Mobile Services Application object.
Security	An outline of key security concerns and information about providing access control.
Load Balancer Configuration Examples	Recommendations for load balancing.

Basic Configuration

This page details the basic configuration steps required before your can use your Genesys Mobile Services installation. For a more general look at the configuration options available, refer to Configuration Options. Additional documentation is also available that describes considerations and configuration for:

- security and access control for your Genesys Mobile Services solution.
- load balancing suggestions for your Apache server.

Basic Configuration Overview

Genesys Mobile Services provides a set of APIs or services that require configuration before the product can be used. Depending on the volatility of their configuration, these services can either be configured in Configuration Server (when the volatility is high) or have their configuration embedded directly into a .WAR file.

This page is limited to managing configuration options using 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:

- 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.

Additional Considerations

Some parts of Genesys Mobile Services are configured outside of Genesys Configuration Server. One example is the *cmd.properties* file inside of the web application archive, which contains some global configuration objects such as connection parameters for accessing Configuration Server. Another would be load balancing, which relies on web server configuration.

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. Steps required to create and populate this resource list are provided below.

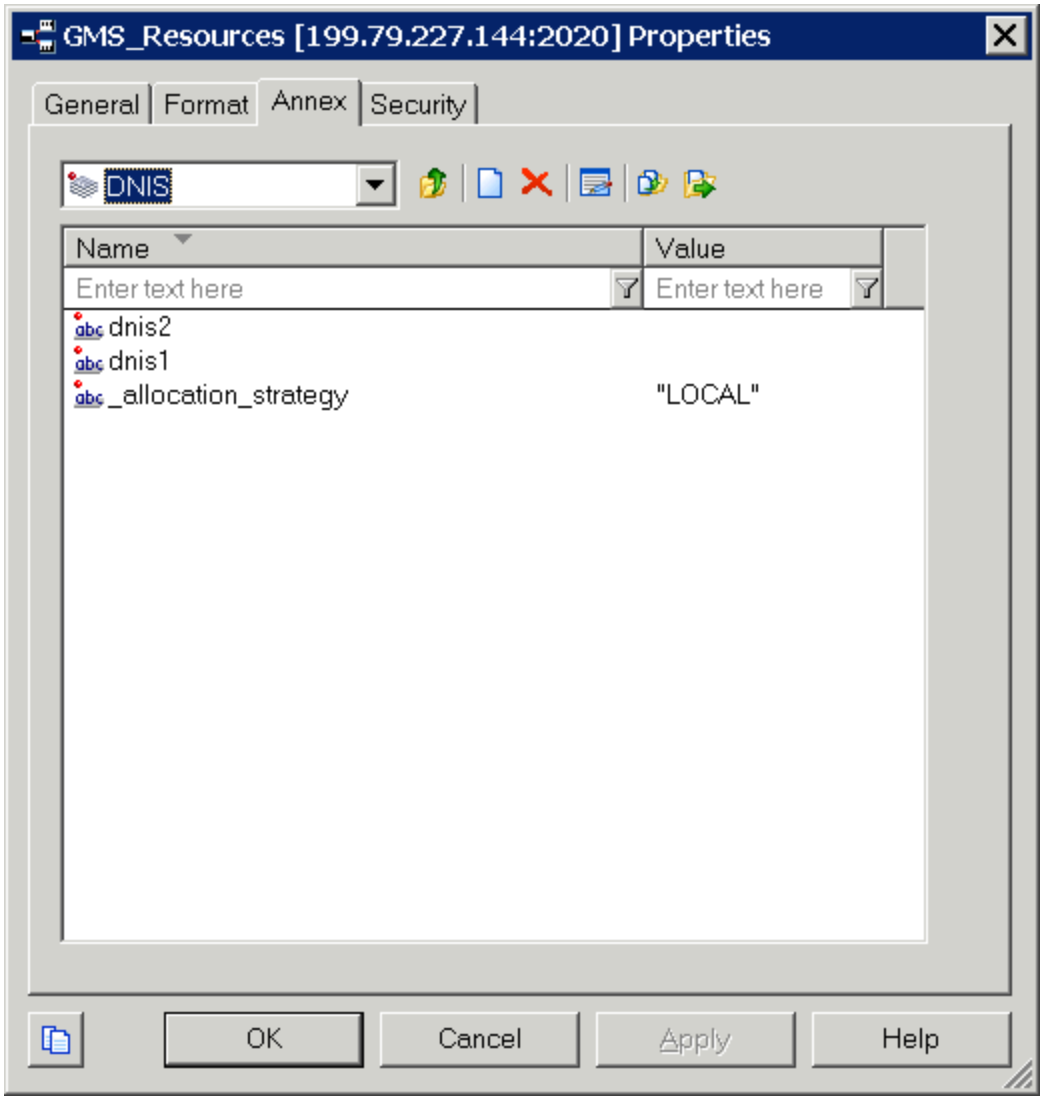
Note: Be sure that you have the *Show Annex tab in object properties* option selected from the *View > Options* menu before starting this procedure.

Start of Procedure

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 of Procedure

A sample resource list configuration is shown below.



Configuring Services for Genesys Mobile Services

To complete your deployment, the following services need to be configured in your Genesys Mobile Services Application object:

- Genesys Mobile Services-Based Services
- Orchestration Server-Based Services
- Native Push (Notification)

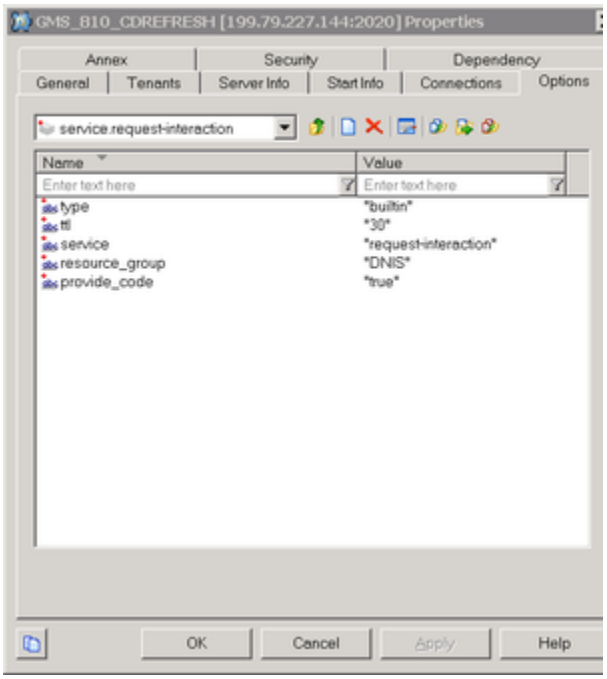
Genesys Mobile Services-Based Services

The following services need to be configured for this category:

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

Required options are outlined below, with some sample values and screenshots provided to help you get started. For more information about configuring these services, see the Overview of Services Provided by Genesys Mobile Services.

request-interaction

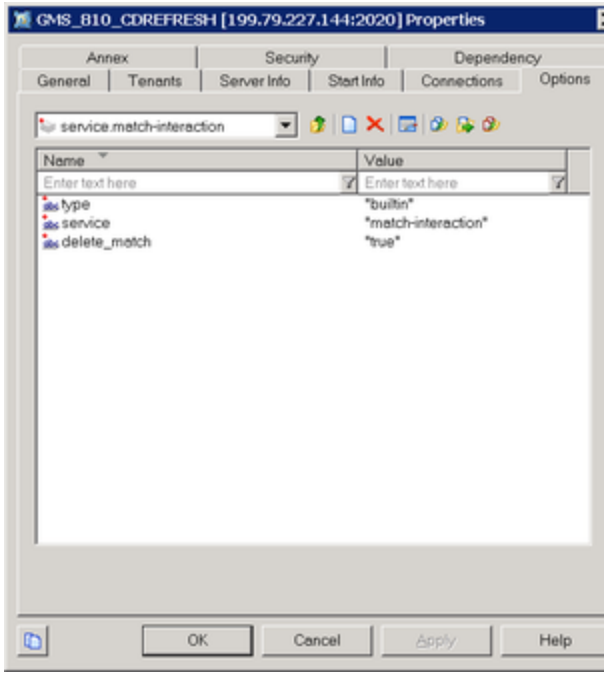


Sample request-interaction options

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

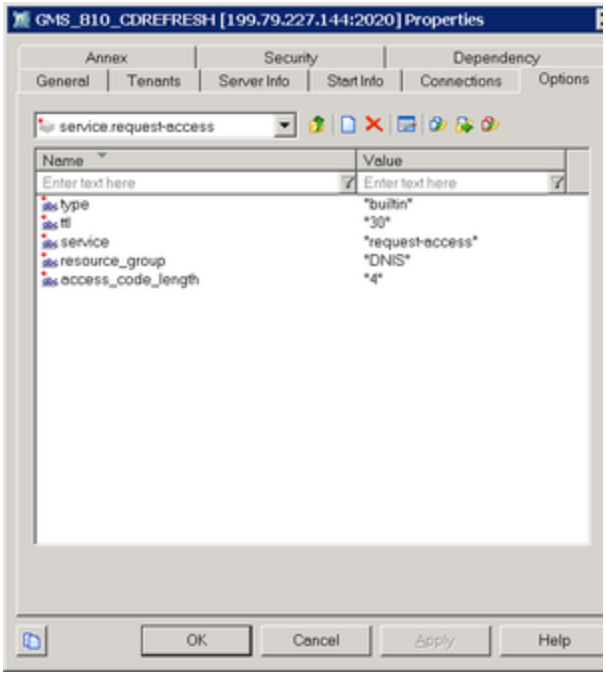


Sample match-interaction options

Required match-interaction Options

Option Name	Option Value
type	builtin
service	match-interaction
delete_match	true

request-access



Sample request-access options

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

Orchestration Server-Based Services

This page shows five sample services that can be configured:

- request-inbound-immediate
- request-inbound-delay
- request-inbound-poll
- request-outbound-immediate
- request-outbound-delay

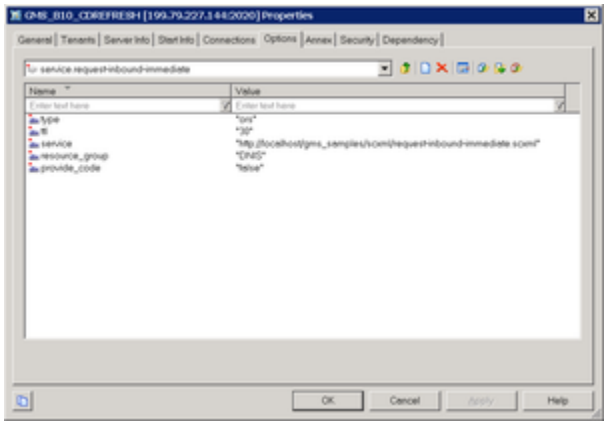
These services are based off of sample SCXML applications that are distributed with Genesys Mobile Services. These sample files can be downloaded from the Sample Resources page. Applications should be customized to fit the business logic for applicable use cases, and then made available on

an application server. The URL is then specified as the *service* parameter in the applicable Options section of your Genesys Mobile Services Application object.

Other services can be configured by creating similar sections with the service parameter using a different URL that points to the SCXML application hosted on the application server of your choice.

Dependency: Genesys Mobile Services-based services should be configured first since Orchestration Server-based services are dependent on the match-interaction and request-access services.

request-inbound-immediate

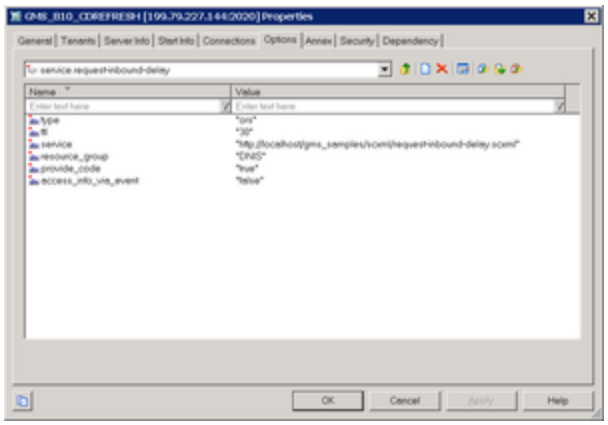


Sample request-inbound-immediate options

Required request-inbound-immediate Options

Option Name	Option Value
type	ors
ttl	30
service	http://<your server>/gms_samples/request-inbound-immediate.scxml
resource_group	(Use the section name created earlier under your GMS_Resources Transactions object.)
provide_code	true

request-inbound-delay

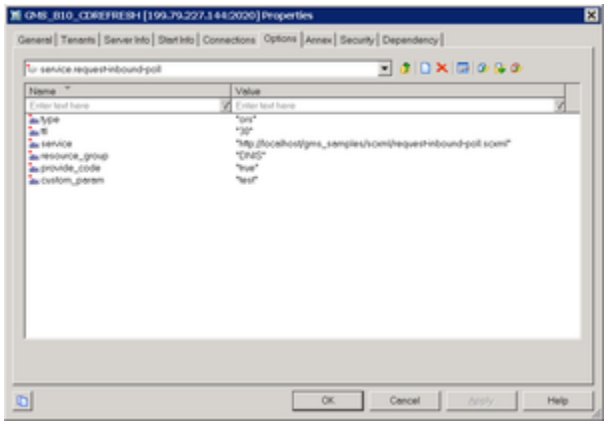


Sample request-inbound-delay options

Required request-inbound-delay Options

Option Name	Option Value
type	ors
ttl	30
service	http://<your server>/gms_samples/request-inbound-delay.scxml
resource_group	(Use the section name created earlier under your GMS_Resources Transactions object.)
provide_code	true

request-inbound-poll



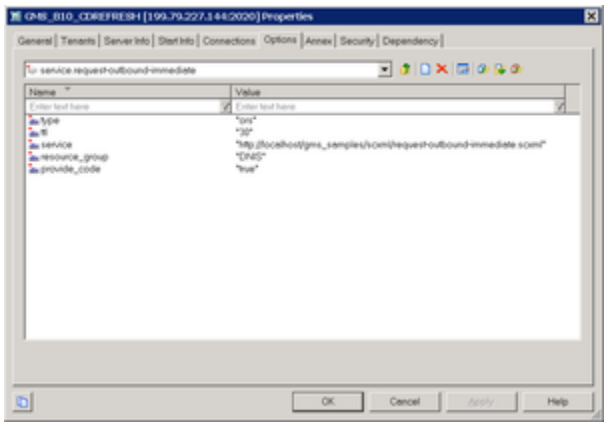
Sample request-inbound-poll options

Required request-inbound-poll Options

Option Name	Option Value
type	ors
ttl	30

Option Name	Option Value
service	http://<your server>/gms_samples/scxml/request-inbound-poll.scxml
resource_group	(Use the section name created earlier under your GMS_Resources Transactions object.)
provide_code	true

request-outbound-immediate

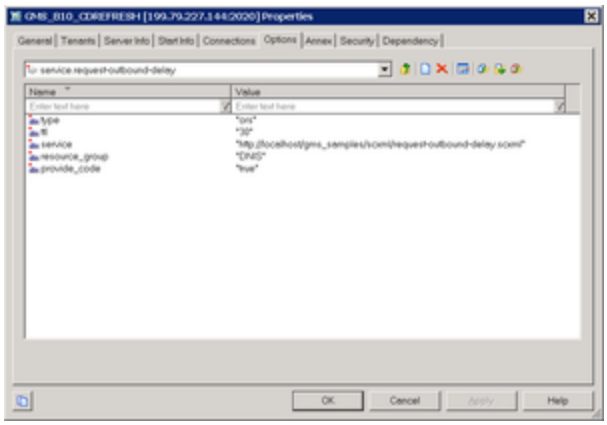


Sample request-outbound-immediate options

Required request-outbound-immediate Options

Option Name	Option Value
type	ors
ttl	30
service	http://<your server>/gms_samples/scxml/request-outbound-immediate.scxml
resource_group	(Use the section name created earlier under your GMS_Resources Transactions object.)
provide_code	true

request-outbound-delay



Sample request-outbound-delay options

Required request-outbound-delay Options

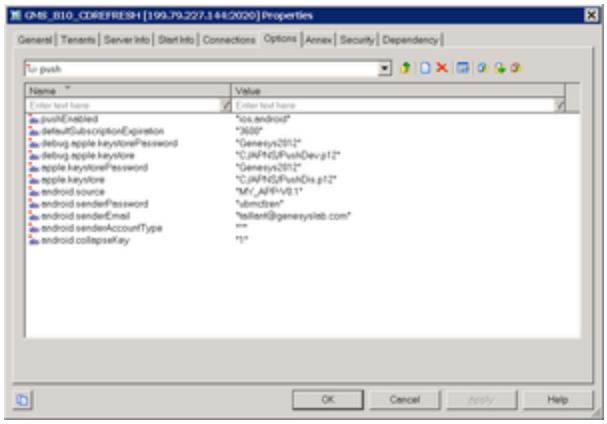
Option Name	Option Value
type	ors
ttl	30
service	http://<your server>/gms_samples/scxml/request-outbound-delay.scxml
resource_group	(Use the section name created earlier under your GMS_Resources Transactions object.)
provide_code	true

Native push (notification service)

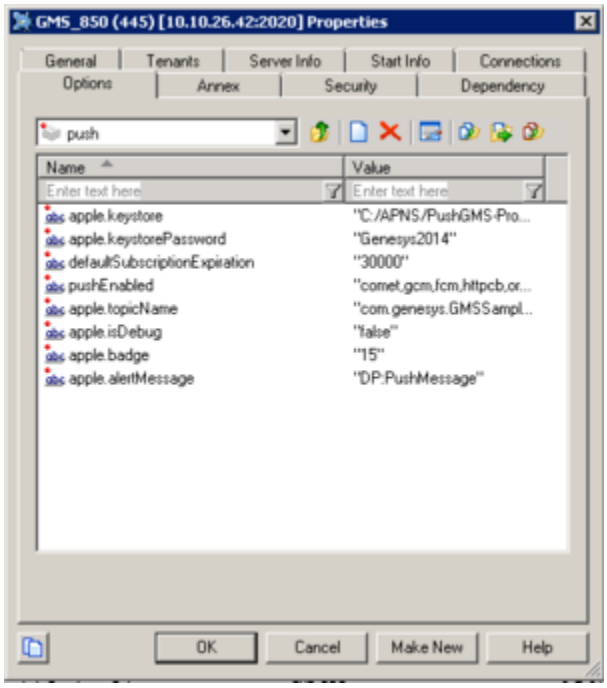
Some services, such as request-inbound-delay or request-outbound-delay, send native push messages to the mobile device. For this to work, both general and device-specific settings need to be configured correctly in the *push* section of your Genesys Mobile Services Application object.

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 C2DM Device Settings:**

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

Mandatory 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. For general information about the push notification service, refer to Push Notification Service.

Deploy DFM Files for Orchestration Server-based Services

Included with your installation are special configuration files for Orchestration Server called DFM. 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
- Genesys Mobile-Based Services
- Geo-coding (for location based services)

The latest DFM definition files are available to download from the Genesys Mobile Services Sample Resources page. Details about deploying these DFM in your environment are provided in Deploy DFM Files in Orchestration Server. 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 Orchestation Server-based services. Please refer to the Genesys Mobile Services API Reference for syntax of the requests.

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

Start of Procedure

1. Download and extract the latest DFM sample files from the Genesys Mobile Services Sample Resources

page.

2. Start Configuration Manager.
3. In Configuration Manager, select *Environment* > *Applications*.
4. Locate and open the Application object for your Orchestation Server. (**Note:** This should be the same Application object you created a connection to when configuring your Genesys Mobile Services Application object.)
5. Select the *Options* tab.
6. Open the *dfm* section.
7. Create and configure one option for each DFM, using the option value to specify the file path. Details are provided in the table below.
8. Click *OK* to save your changes.
9. Restart Orchestration Server.
10. Restart Genesys Mobile Services.

End of Procedure

List of DFM Options for Orchestration Server

Service	Option Name	Option Value
Storage	gsgStorage	file://<*extraction path*>\ors\drm\cfg_GSG_storage.txt
Notification	gsgNotification	file://<*extraction path*>\ors\drm\cfg_GSG_notification.txt
Genesys Mobile-Based Services	gsgBasedServices	file://<*extraction path*>\ors\drm\cfg_GSG_base_services.txt
Geo-coding	geocoding	file://<*extraction path*>\ors\drm\cfg_YahooPlaceFinder.txt

Next Steps

With basic configuration complete, you are ready to consider more advanced configuration details.

- Understand and implement additional configuration steps to provide security and access control for your Genesys Mobile Services solution.
- Configure load balancing for your server.
- Review the sample applications included with this installation to help understand how you can leverage Genesys Mobile Services in your own development.
- Read the API Reference to get a better grasp on what type of services are available with Genesys Mobile Services.

Chat Support

Introduced in release: **8.1.100.14**
Internal poke introduced in release: **8.1.100.28**
chat_endpoint option introduced in release: **8.1.100.28**

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

Configuration Overview

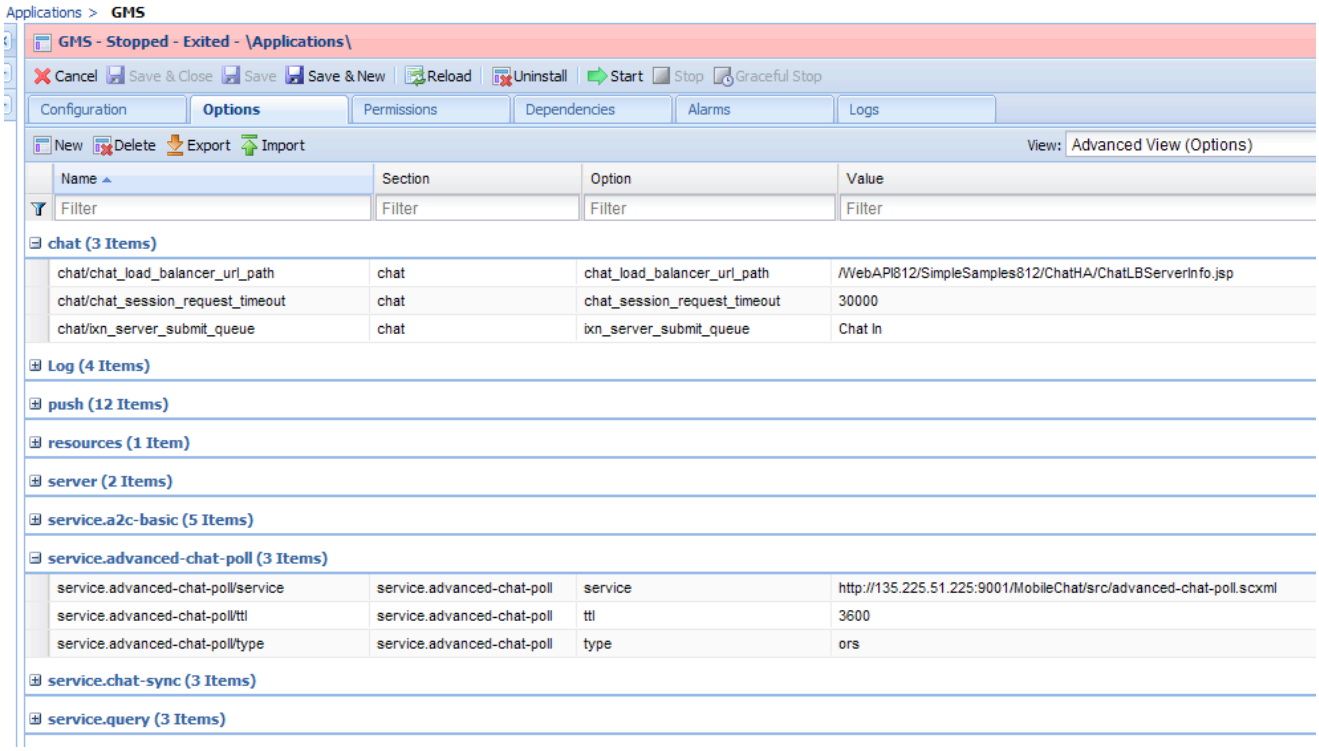
Prerequisite: Before beginning the steps described here, you should have completed the basic configuration process. To use the Chat API with your Genesys Mobile Services deployment, you must specify configuration details in the Application objects for the following objects:

- 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:



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, include the mandatory configuration options described in the table below.

Required Genesys Mobile Services Options

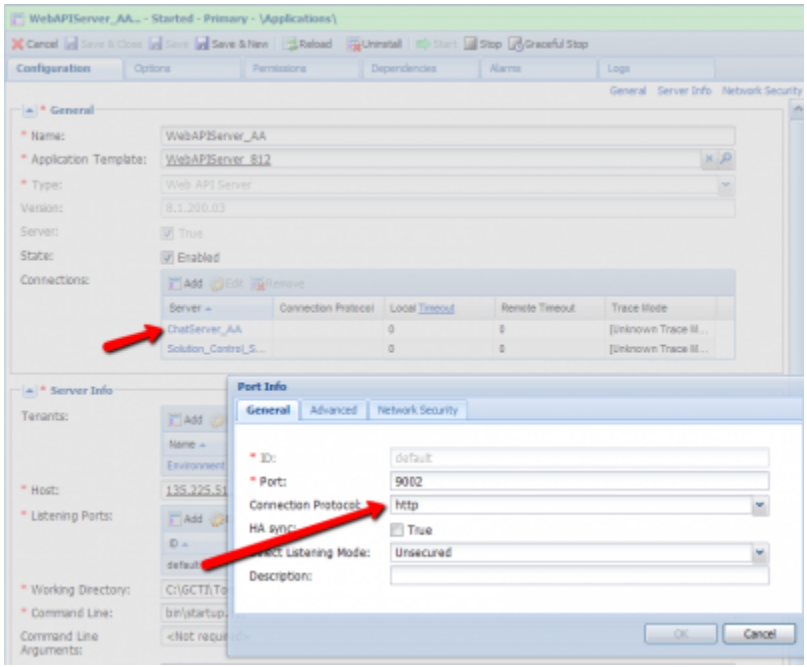
Section: chat			
Option Name	**Required**	**Option Value**	**Description**
chat_load_balancer_url_path	true	WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp	Url to the load balancer (WebAPI) for Chat servers
chat_session_request_timeout	true	30000	Duration after which the chat interaction gets 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

Section: chat			
			defined in the Chat server options->endpoints:1.
Section: service.request-chat-poll			
Option Name	**Required**	**Option Value**	**Description**
service	true	<url to SCXML application request-chat-poll.scxml>	URL to the scxml application to be fetched by ORS
ttl	true	3600	Duration after which service will time out and exit
type	true	ors	Should always be ors since this is an advanced service
chat_endpoint	false	<endpoint defined in chat server>	The endpoint configured on the Chat server on which the new chat interactions from GMS will arrive. Any value specified here should also be defined in the Chat server options > endpoints:1. When this value is not specified, GMS will use the value of "ixn_server_submit_queue" configured in options > chat. The endpoint should be associated with a queue configured to execute an ORS workflow (inbound_chat.scxml). This workflow is responsible to forward the chat interaction to the GMS service for advanced routing.
internal_api_url_base	false	http://<gmshost>:<gmsport>	Base url used to generate and attach the poke url to be used by agent desktop. Required if poke feature is required.
default_poke_message	false	<your poke message>	Default poke message to be sent on an internal-poke request without the poke_message parameter. Required if poke feature is used.

The following configuration options can be used to test the poke capability using the Interaction Workspace test plugin, which is available as a download: Genesys Mobile Services Interaction Workspace Test Plugin

Section: service.request-chat-poll			
Option Name	**Required**	**Option Value**	**Description**
iws_plugin_url	false	<url to the plugin.jsp>	URL to the JSP page to be fetched by Interaction Workspace. Plugin.jsp is provided in the samples for test purposes.
iws_plugin_view_name	false	<name of the view in Interaction Workspace>	Name of the view (tab) that displays the plugin page.

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. The Web API Server Application object being used by your Genesys Mobile Services deployment can be updated using the following procedure:

Start of Procedure

1. Open Genesys Administrator in a web browser.

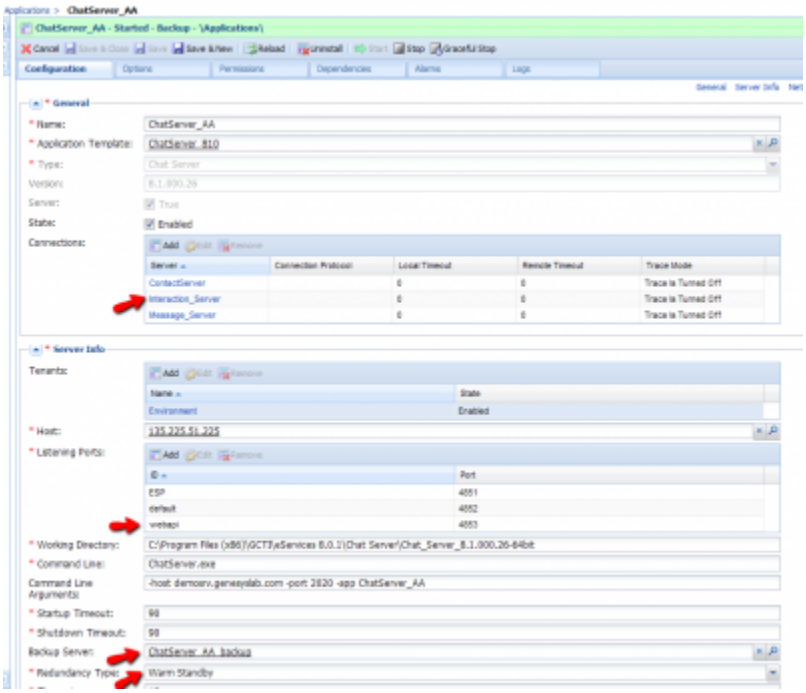
2. Locate and view the Web Server API Application object associated with your Genesys Mobile Services deployment.
3. View the *Configuration* tab.
4. In the *General* section, find the *Connections* table and click *Add*.
5. Locate and select the Chat Server Application object that you want to use.
6.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

End of Procedure

Note:

- Download ChatLBServerInfo.jsp for Single Tenant
- Download ChatLBServerInfo.jsp for Multi-Tenant

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.

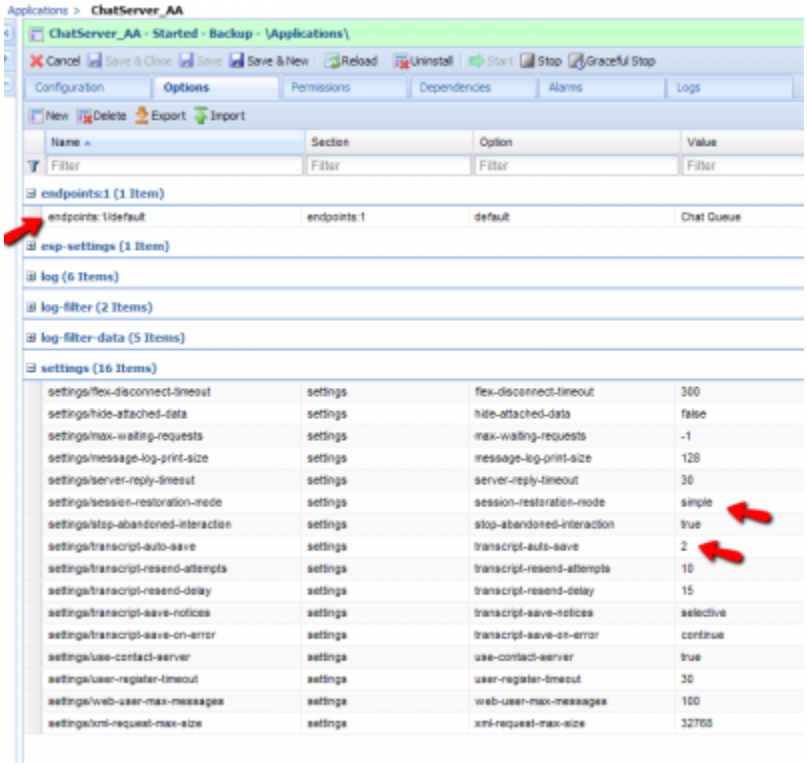
Detailed steps are provided below:

Start of Procedure

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 of Procedure

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 of Procedure

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 of Procedure

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

Configuration Options

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	2	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	2	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).
apple.isDebug	provider	Boolean	optional		Specifies whether to use production or debug servers for Apple Push notification.
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.

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
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 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	GSH formatting	Boolean	optional		If included and true, indicates that the message should not be

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

Option name	Affinity	Option type	Necessity	Restriction on value	Notes
					(in seconds) to wait before Google C2DM service can be accessed again if the request returned the 503 code (Service unavailable). Please 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 Genesys Mobile Services 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

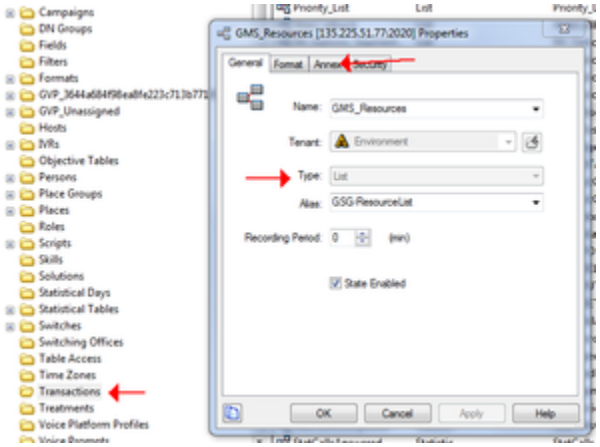
Changes take effect: Immediately.

Option name	Option type	Default value	Restriction on value	Description
resources_list_name	String	GMS_Resources	Mandatory	Name of the Strategy configuration object (of type List) which holds configuration details of resources and resource groups.
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.



Gsg list.png

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

Option name	Option type	Default value	Restriction on value	Description
				tokens.
dateFormat	String		Optional	The string used to format dates. Syntax of the string should meet the expectations of java class java.text.SimpleDateFormat. See 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

Option name	Option type	Default value	Restriction on value	Description
				• For Orchestration Server-based services: ors
service	String		Mandatory	• For Genesys Mobile Services-based services: The name of the matching service. • For Orchestration Server-based services: The URL of the service's SCXML application.
ors	String		Optional	The IP address of the ORS instance this service will use. Overrides any ORS connections, if they are present. **Note:** Only used for Orchestration Server-based services.
_booking_expiration_timeout	Integer		Optional Valid values: Lower limit is 5 seconds and upper limit is 1800 seconds (30 minutes).	This option is specific to the service.request-interaction and service.request-access services, and applies only to LOCAL and CLUSTER allocation strategies. This option allows you to set a different value per service for the booking expiration timeout. This value can also be passed through the request-access URI parameter. Note that the value passed through the request-

Option name	Option type	Default value	Restriction on value	Description
				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.

Security

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 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:

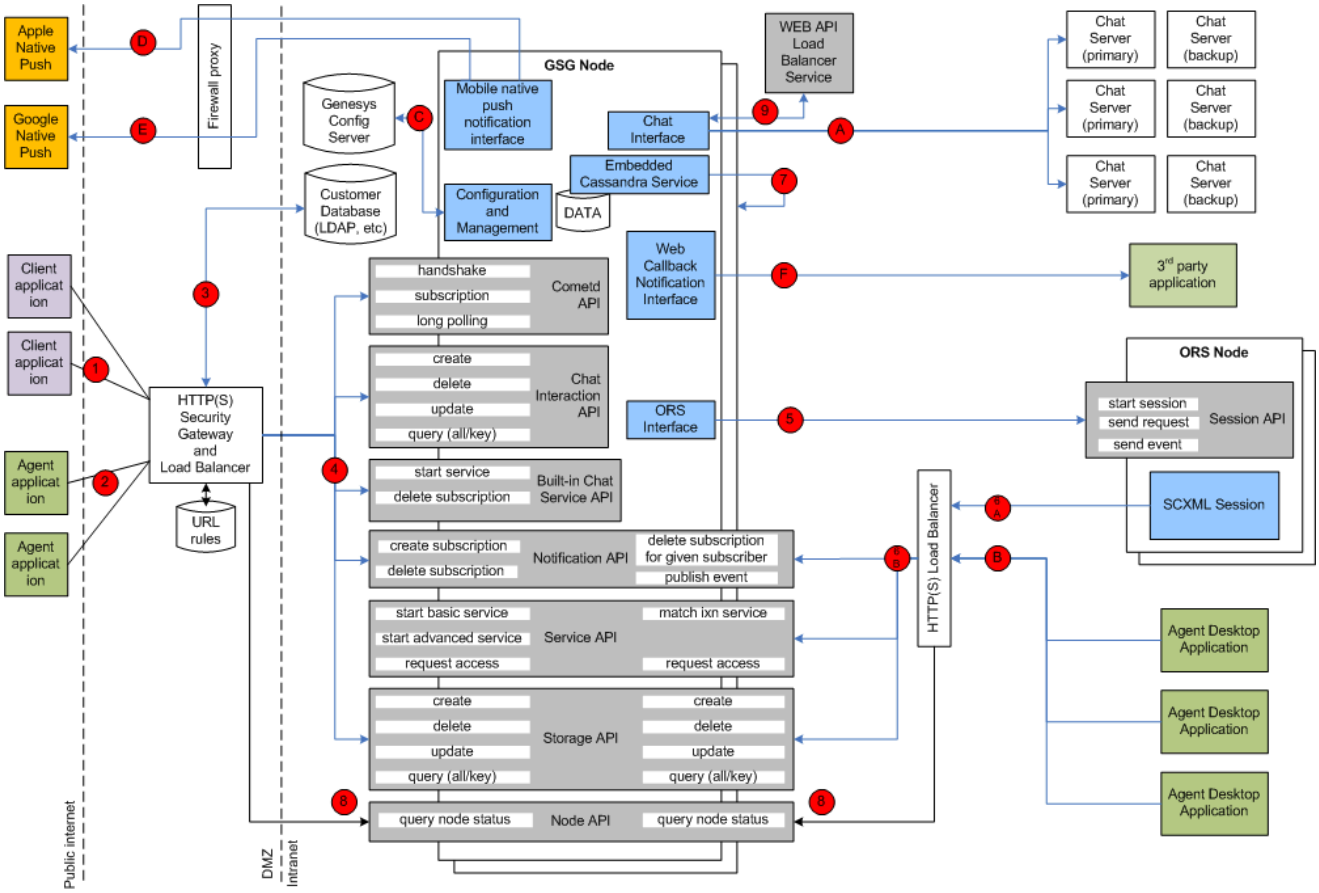
- client authentication (optional)
- TCP port and URL access control
- load balancing functionality, distributing the load between multiple Genesys Mobile Services nodes responsible for processing API requests
- HTTP connection encryption (SSL)

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 normal 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 change/ configure this port. See also GMS-	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/ digest/ oAuth/... client authentication.

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
					>Options->/server/external_url_base configuration option in Configuration Manager		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 against Configuration Server, or other

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							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 according to the access rules

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							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 balancing gateway can also be

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							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>/launch.xml*: *<parameter name="http_port" displayName="Http 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->yourapp->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 and Storage APIs.	HTTP	Deployment specific. See 3rd party documentation.	Hard coded as part of the Genesys Mobile Services API URL inside SCXML session. Could be configured in	Security proxy should be configured to block access to the API URLs according to the access rules below.

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
						Configuration Manager and read be SCXML session	
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 Genesys Professional Services.
Cassandra	(7)	Local and	Cassandra	TCP/IP	Defaults:	Uses local	Most of

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
Instance Embedded into Genesys Mobile Services Node		remote Genesys Mobile Services nodes accessing distributed Cassandra storage	client API		• 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.	connection to embedded Cassandra instance.	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 nodes. Enable internode encryption in

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							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 Chat Server - N+1 Primary/ Backup	(A)	Genesys Mobile Services node	FlexChat protocol	TCP/IP, TLS encrypted if required	Default port: 4856. Configured in Configuration Manager.	Read dynamically before each chat API call (refresh/	Enable TLS encryption if a higher level of protection is

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
Pairs					See *GMS->Connections->WEB API Server->Connections->Chat Server (primary)->Server Info->Listening Ports->webapi port*.	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.	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 reacting on changes	TCP/IP protected by TLS is required	Default: 2020. Configured in Configuration Manager configuration file.	Configuration is in *<Genesys Mobile Services installation directory>/startScript.bat* parameters **-app', '-host' and '-port**. See	Enable TLS encryption if a higher level of protection is required. This is a Platform SDK-based connection supporting

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
						launcher.xml in the same directory for description of the parameters.	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 Messaging	Google require SSL protected connection with client side certificates. Make sure your firewall allows access to the *https://android.googleapis gcm/send* URL from

Server Component	Connection #	Client Type	API/ Function	Transport	Server: Port, Configuration	Client Configuration	Security Recommendations
							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:
{base url:port}//genesys/cometd

{base url:port}/genesys/{version}/service/{service name}
{base url:port}/genesys/{version}/service/{id}
{base url:port}/genesys/{version}/service/{id}/request

{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}

{base url:port}/genesys/{version}/notification/subscription
{base url:port}/genesys/{version}/notification/subscription/{id}

{base url:port}/genesys/{version}/service/chat-sync
```

```
{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/*
Block:
{base url:port}/genesys/{version}/service/match-interaction
{base url:port}/genesys/{version}/notification/publish
{base url:port}/genesys/{version}/node
```

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

With advanced configuration complete, you are ready to verify your deployment and start working

with Genesys Mobile Services!

- Configure load balancing for your server.
- Review the sample applications included with this installation to help understand how you can leverage Genesys Mobile Services in your own development.
- Read the API Reference to get a better grasp on what type of services are available with Genesys Mobile Services.

Apache Load Balancer

The following is an example of how to configure Apache load balancer that can be positioned in front of GSG 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 if 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

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>
```

Starting and Stopping

Overview

You can start Genesys Mobile Services in any of the following ways:

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

Starting and Stopping Genesys Mobile Services Applications Using Solution Control Interface

Prerequisites

- Genesys Mobile Services is installed. See Deployment Guide.

Start

1. From the Applications view in SCI, select Genesys Mobile Services Application object on the list pane.
2. Click the appropriate button (`Start` or `Stop`) 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 8.1 Solution Control Interface Help*.

Starting and Stopping Genesys Mobile Services Applications Using Genesys Administrator

Prerequisites

- Genesys Mobile Services is installed. See Deployment Guide.

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

Create Genesys Mobile Services Related Alarms

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

1. 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)
2. No more resources Alarm (EventId 2001): This alarm is raised by GMS when no more resources are available in GMS (LOCAL or CLUSTER strategy).

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

Web API Resources

Web API Resources Overview

Files to be added into the Web API Server directory.

Downloadable Files

- Genesys Mobile Services Web API Resources