SIP Feature Server Administration Guide

SIP Feature Server 8.1.2

8/28/2024

Table of Contents

SIP Feature Server Administration Guide	**4**
Users and DNs	**6**
Provisioning users	7
Find Me Follow Me	10
Provisioning user groups	15
Provisioning DNs	16
Devices	**18**
Voicemail	**31**
Provisioning mailboxes	33
Voicemail notifications	35
Voicemail profiles	40
Voicemail Forwarding	47
Voicemail Opt Out	50
Dial plan	**52**
Creating Partitions	53
Creating Calling Profiles	55
Creating Call Forwarding Profiles	58
Editing Dial Plan Settings	59
Dial Plan Administration	61
Maintenance	**63**
Starting Feature Server	64
Stopping Feature Server	65
Upgrading Feature Server	66
Re-initialize Feature Server backend	70
Enable / Disable Voicemail Deposits	76
Check and Refresh Mailbox Counters	78
Remove Metadata of Expired Voicemail Messages	80
Managing Feature Server Cassandra backend	81
Python Scripts	82
Configuration Database Synchronization	113
Retrieve provisioned and unprovisioned devices	115
Set up mailbox and user time zones	116
Scheduled maintenance tasks	117
GDPR Compliance	119
Appendix	**125**

Appendix: Backing up and restoring embedded Cassandra data 126
Appendix: Performing maintenance operations on embedded Cassandra 128
Appendix: Feature Server Maintenance Python Scripts for Embedded Cassandra 134
Appendix: Remove a node from Feature Server deployed with Embedded Cassandra 136

SIP Feature Server Administration Guide

SIP Feature Server enables you to provision users and manage devices, mailboxes, and voicemail. You can also set up the dial plan that governs call disposition.

SIP Feature Server can be administered using Genesys Administrator Extension (GAX) with Feature Server plugin installed. For supported versions of GAX and Feature Server plugins that can be used with a particular Feature Server version, refer SIP Feature Server Release Notes.

Some tasks, such as user and user group creation, can occur only in Genesys Administrator.

If your environment administers the dial plan through SIP Server, the GAX interface does not display the dial plan.

Voice prompts for the Telephone User Interface (TUI) are available in 13 language variants: English (UK, US, and AU), Spanish (Spain and Latin America), German, French, Italian, Japanese, Brazilian Portuguese, Russian, Chinese, and Korean.

Users and DNs

Provision users and manage devices.

Provision users

Find Me Follow Me

Provision user groups

Provision DNs

Devices

Manage SIP devices

Voicemail

Enable voicemail and set up mailboxes and voicemail profiles.

Set up mailboxes

Manage voicemail notifications

Manage voicemail profiles

Dial Plan

Administer the dial plan.

Create partitions

Create calling profiles

Create forwarding profiles

Edit dial plan settings

Maintenance

Maintain your Feature Server installations.

Start Feature Server

Stop Feature Server

Upgrade Feature Server

Cassandra Database Maintenance

Maintain your Cassandra database.

Back up and restore Cassandra data

Maintain your Cassandra database

Remove a Cassandra cluster node

Reimport Cassandra data

Important

SIP Feature Server's UI has been deprecated from version 8.1.201.83 dated 09/14/16, and is not supported any further. Therefore, all administrative tasks must be performed using GAX.

Users and DNs

SIP Feature Server enables you to:

- provision users and user groups
- set options such as roles and call preferences
- manage DN settings

Provisioning users

You create users and perform most provisioning in Genesys Administrator, then assign roles, mailbox access, a calling profile, a time zone, and voicemail notification preferences in Genesys Administration Extension (GAX).

> **Important**
>
> Do not assign any user to more than 10 mailboxes, including individual and group mailboxes. Note that mailbox assignments are among the tasks you perform in GA.

To provision a user:

Log into GAX as an administrator (*GAX IP address*:*port*/gax). Under **Operations > SIP Voicemail & Call Settings**, select **Users**. Search for and click the user name of the user that you want to provision, or select **Bulk Assignment** to assign calling profiles, voicemail profiles, or forwarding profiles to multiple users simultaneously.

General

1. Select one or more Feature Server roles:
   - *User* (default) grants the user access to voicemail.
   - *Administrator* grants the user the ability to log in as administrator and perform all the tasks available in the Feature Server GAX interface.
   - *Group Mailbox Administrator* grants the user the ability to log in as administrator and manage group mailboxes in GAX. Only users who are Group Mailbox Administrators can change greetings and passwords for group mailboxes.
2. Select a User Mailbox Access profile:
   - *Phone + Web View + Web Playback* enables the user to access voicemail over the phone and to view and play voicemail through GAX.
   - *Phone + Web View* enables the user to access voicemail over the phone and to view, but **not** play, voicemail through GAX.
   - *Phone Only* enables the user to access voicemail over the phone, without web access.
3. To activate voicemail access, select a Calling Profile other than Not Set. If the menu is empty, you can Create a Calling Profile or let the default calling profile apply to the user.
4. Select a voicemail profile.
5. Select a forwarding profile.
6. Select a time zone for message playback. When a user specifies a time zone in their user profile, the value overrides this setting. The user time zone also takes precedence over the default time zones for the application, switch, and mailbox, unless:

- the user time zone is set to Default and the mailbox time zone is not set to Default, or
- the user uses the telephone UI to log into a group mailbox anonymously, without first logging into a personal mailbox.

  In both cases, the mailbox time zone takes precedence.

> **Tip**
> Click the values in the Agent Logins, DNs, and Mailboxes tables to view user assignment details.

Call Settings

Click **Call Settings** to set the default values for this user. You can apply or change these values only when the user has a DN or agent login assigned. Save your changes before you leave this tab.

Setting	Values (default value in bold)	Description
Reject Call On Not Ready	**System (Off)**, Off, On	Rejects call when a user is not ready on a device.
Call Waiting	**System (On)**, Off, On	Does not reject a call when the user or device is already in a call.
Forward All Calls	**Off**, Forward All Calls To + *phone number*, Find Me Follow Me	Forwards all incoming calls to the specified number or to one or more of the destinations specified in the specified Find Me Follow Me rules. **Important:** To forward calls to voicemail, the number specified must be the number configured as the VoIP DN (of service type `Voicemail`) for the associated switch. See Configure SIP Server for Feature Server.
Forwarding On No Answer	**System (Off)**, Off, On + *phone number*	After the No Answer Timeout value elapses, forwards calls to the specified number.
No Answer Timeout	**System (30)**, 5 to 60 seconds (in 5-second intervals)	Specifies the length of time, in seconds, that Feature Server waits for the user to answer a ringing call.
Forwarding On Busy	**System (Off)**, Off, On + *phone number*	When the user is on a call, forwards calls to the specified number.

Note: User settings have a higher priority than dial plan settings.

Email Notifications

1. Click **Email Notifications** to set the values for this user. These values apply only if the associated voicemail profile has email notifications enabled.

2. Select **On** or **Off** to enable or disable notifications by email.
3. If you have enabled email notifications, in **Email To** type an email address to which you want the notifications sent. Use the standard address format: *name*@*domain*.
4. Save your changes.

Web Service Notifications

1. Click **Web Service Notifications** to set the values for this user. These values apply only if the associated voicemail profile has web notifications enabled.
2. Select **On** or **Off** to enable or disable notifications by web service.
3. If you have enabled web service notifications, in **Phone Number** type a phone number to identify the destination of the notifications. Use only digits.
4. Save your changes.

Bulk Upload of User ID and User Mailbox Password

This option will be enabled in user interface only when the **user-login** = `true`. To assign a user ID and user mailbox password to a user, provide the details mentioned in the following table as a CSV file.

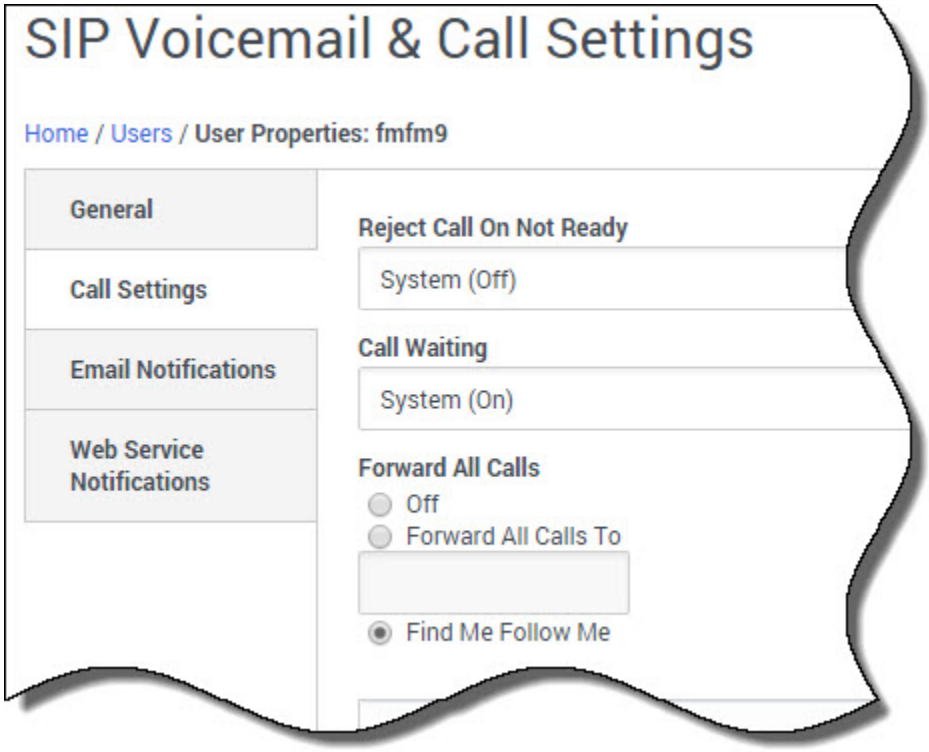
Field	Field Description	Default/Optional Values
Username (M)	The name of the user to be assigned with.	N/A
User Id (M)	A 6-digit unique number to log on through TUI.	N/A
User Mailbox Password (O)	A numeric value of 4 digits (depending on the SIP Feature Server configuration).	N/A

Find Me Follow Me

Find Me Follow Me (FMFM) is an industry-standard method for customizing call forwarding. Users and administrators can specify multiple forwarding destinations that can vary over time and day of the week. Depending on the forwarding profile set for each user, destinations can include some combination of internal extensions, external phone numbers, and voicemail.

Find Me Follow Me requires some configuration steps.

Selecting Find Me Follow Me

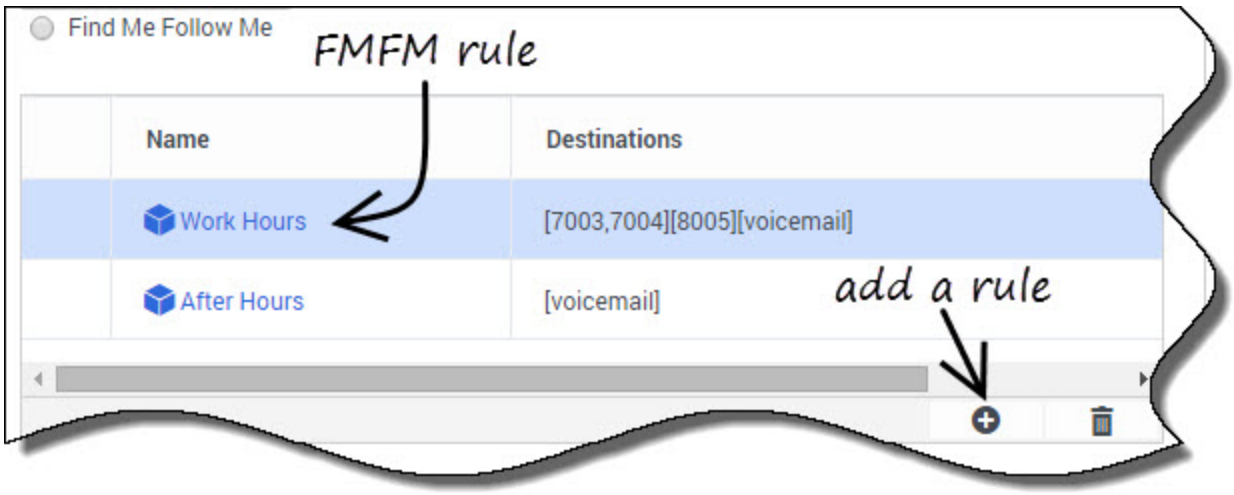


To set up Find Me Follow Me on the **Call Settings** tab of the **User Properties** page, select **Find Me Follow Me** under **Forward All Calls**.

If you can't see the Find Me Follow Me option, the user's forwarding profile doesn't allow it.

Back to top

Selecting FMFM rules



The FMFM rule table lists all the rules that apply to the user. Here, you see the two default rules, **Work Hours** and **After Hours**, and the destinations for each.

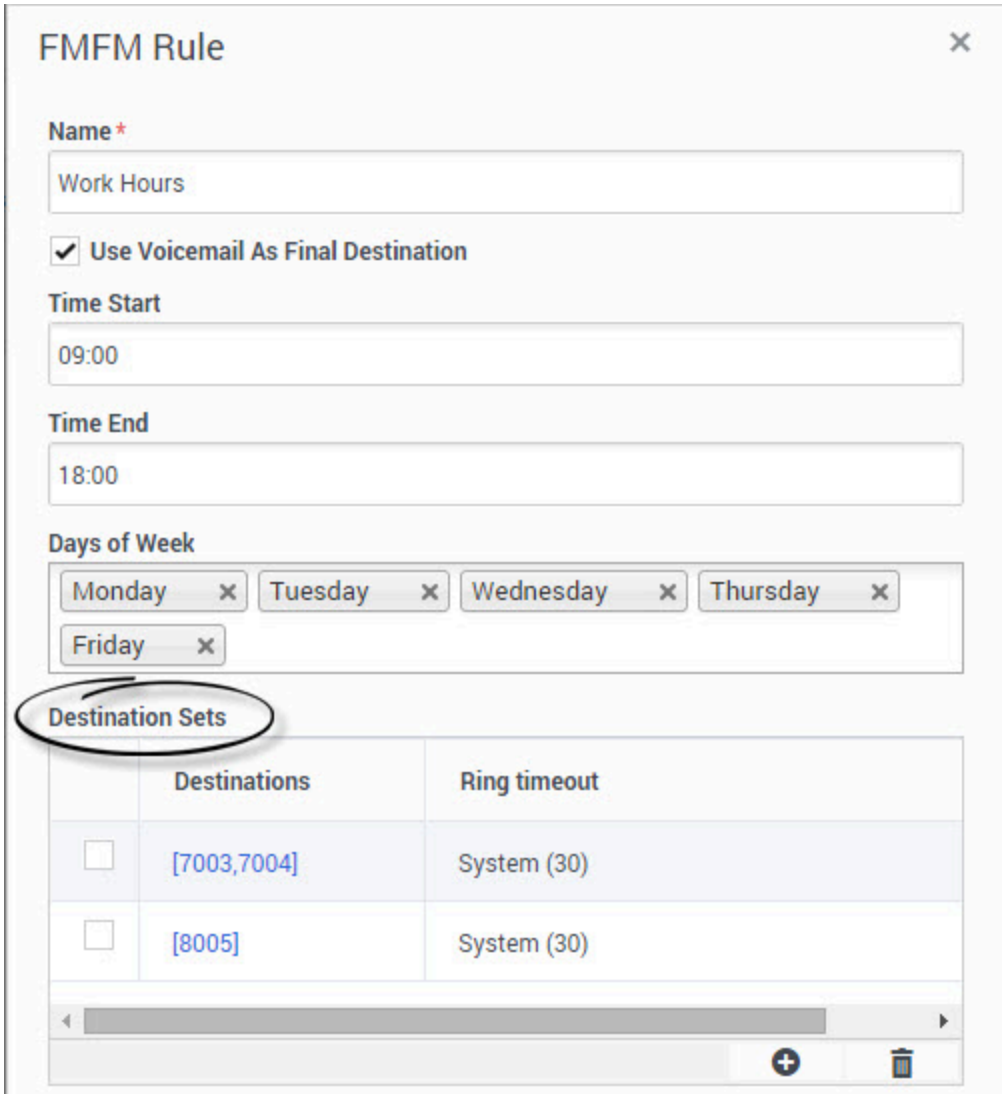
You can edit or remove these rules, with two exceptions:

- The **After Hours** rule acts as the default rule for all times not covered by other rules, so you cannot delete it or edit its times or days. You can, however, change destinations, and all other rules take priority.
- The **Work Hours** rule acts as the default rule for work hours. You cannot delete it but you can change times, days, and destinations, so you can effectively inactivate this rule by removing all days.

You can also create your own rules from scratch.

Back to top

Editing or creating an FMFM rule



The FMFM Rule page opens when you select an existing rule to edit or decide to add a new one.

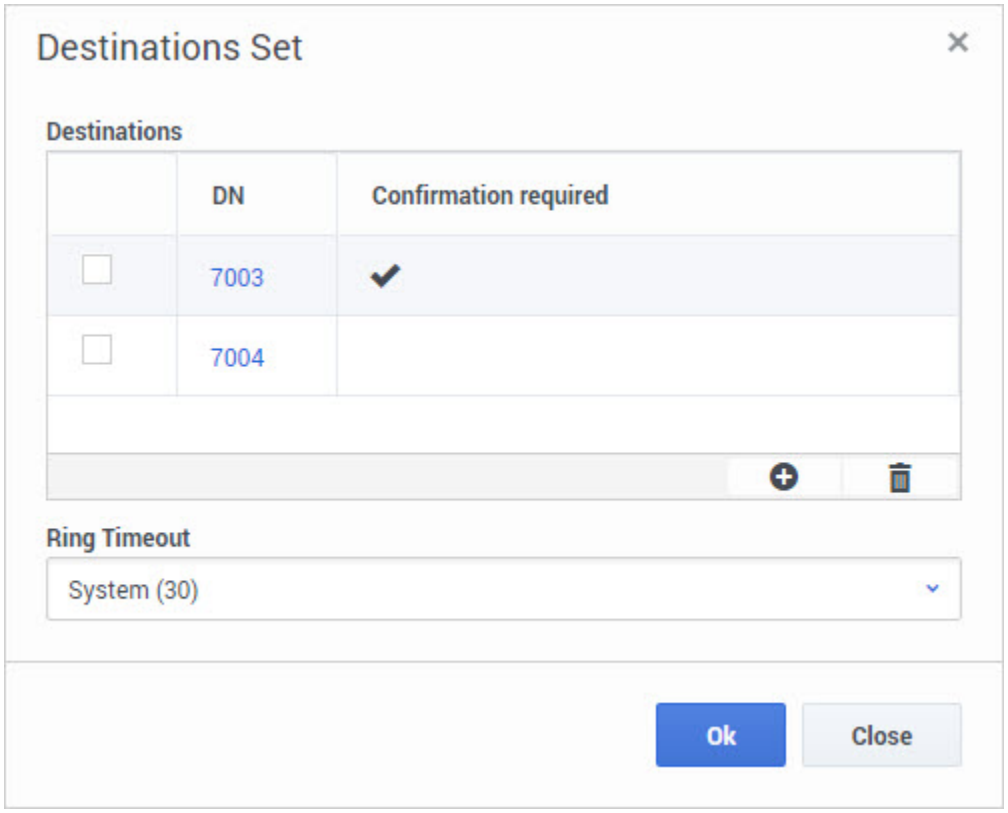
In this example, when the user gets a call during work hours, lines 7003 and 7004 ring simultaneously, because a single destination set includes them both. If no one answers after 30 seconds, line 8005 rings. If no one answers after another 30 seconds, the call transfers to voicemail because the rule has **Use Voicemail As Final Destination** checked.

The default value of **Time End** is End of the day, which is equivalent to a moment immediately before midnight.

One rule at a time: Feature Server ignores all forwarding settings applied to destinations. In this case, if the owners of 7003 and 7004 have set their calls to go to 7777, the call still goes to 8005.

Back to top

Destination sets

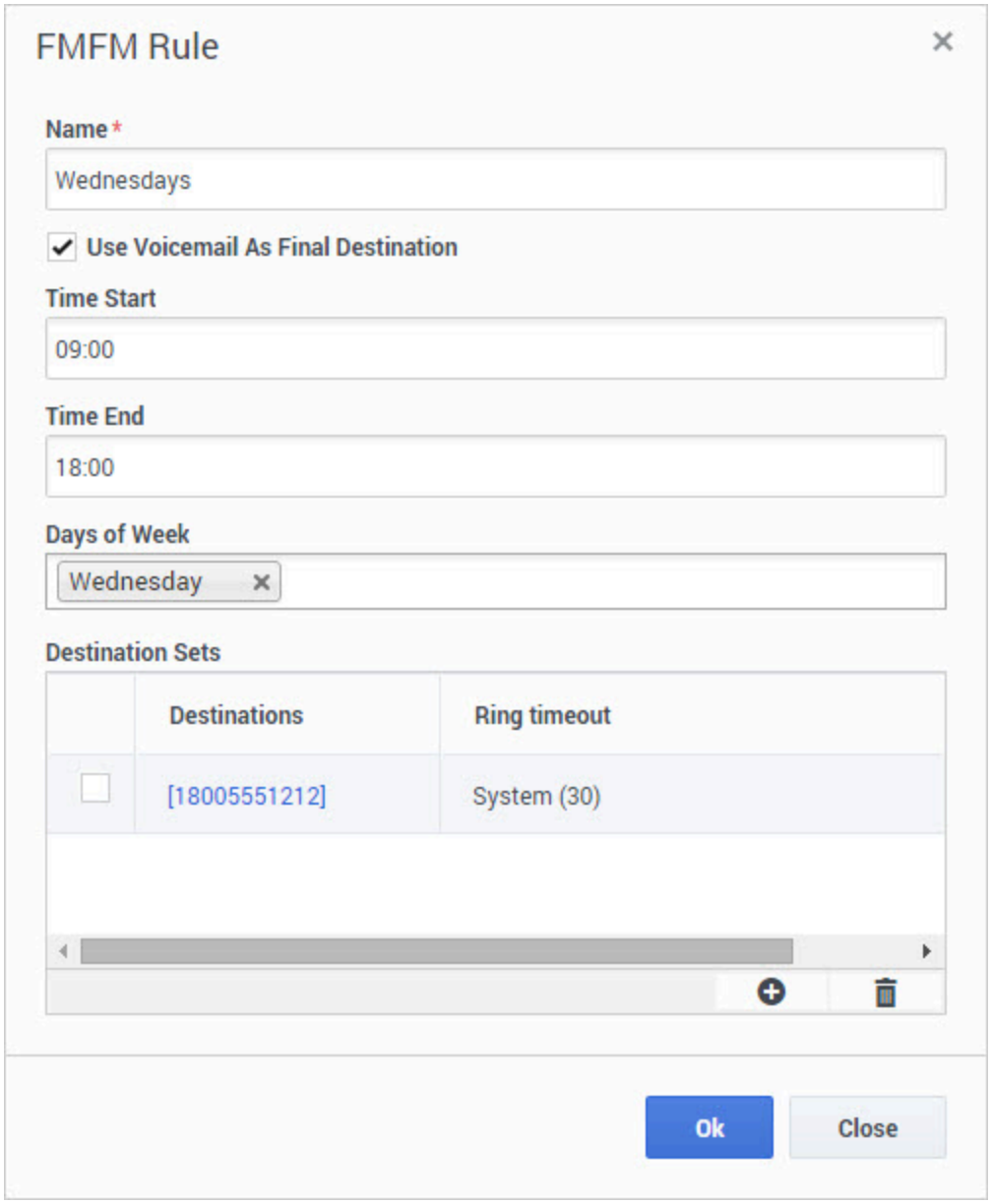


All destinations in a set ring simultaneously. If you want to ensure that a human is handling a call, **Confirmation required** requires the person who answers the phone to enter a digit specified by the spoken prompt, usually zero.

Destinations limited: You might not be able to forward calls to internal destinations at other company sites. If the forwarding profile allows it, you can try to use the external version of that number: 800-555-7003, for example, rather than 7003.

Back to top

Rule example: Wednesdays



If the user typically works away from the office on Wednesdays, you could create a Wednesday-only rule. First, delete Wednesday from the Days of Week in the Work Hours rule (to avoid a conflict). Then create a new rule that uses the same start and end times as the Work Hours rule, but set the Day of Week to Wednesday and the user's mobile phone as the primary destination.

Provisioning user groups

You create users groups and perform most provisioning in Genesys Administrator. In GAX, you can only assign a voicemail profile and set voicemail notification preferences.

To provision a user group:

1. Log into GAX as an administrator (*GAX IP address*:*port*/gax). Under **Operations > SIP Voicemail & Call Settings**, select **User Groups**. Search for and click the user name of the user group that you want to provision.
2. Select a voicemail profile.

Email Notifications

1. Click **Email Notifications** to set the values for this user group. These values apply only if the associated voicemail profile has email notifications enabled.
2. Select **On** or **Off** to enable or disable notifications by email.
3. If you have enabled email notifications, in **Email To** type the email address to which you want the notifications sent. Use the standard address format: *name*@*domain*.
4. Save your changes.

Web Service Notifications

1. Click **Web Service Notifications** to set the values for this user group. These values apply only if the associated voicemail profile has web notifications enabled.
2. Select **On** or **Off** to enable or disable notifications by web service.
3. If you have enabled web service notifications, in **Phone Number** type a phone number to identify the destination of the notifications. Use only digits. If your notification message does not include the user phone number, this field does not appear.
4. Save your changes.

Provisioning DNs

You cannot create DNs in the GAX plug-in. To create DNs, see DNs.

To provision a DN:

1. Log into Genesys Administration Extension as an administrator (*GAX IP address*:*port*/gax). Under **Operations > SIP Voicemail & Call Settings**, select **DNs**. Search for and click the DN that you want to provision or select **Bulk Assignment** to assign calling profiles to multiple DNs simultaneously. The DN Properties window displays:
   - Number of the DN
   - Switch to which the DN belongs
   - Mailbox Number, if any
   - Assigned To, which is the object, if any, to which the DN is assigned
   - Logged In, which is the name of any user who is currently logged into the DN
2. Optionally, set a password for the DN. This password controls device authentication.
3. To activate voicemail access, select a Calling Profile other than Not Set. If the menu is empty, you can Create a Calling Profile or let the default calling profile apply to the device.
4. To provision Softswitch DNs (optional): SIP server uses Softswitch DNs to establish calls with remote agents. Feature Server is provisioned to assign and use a calling profile for VoIP Service DNs with service-type=softswitch. Perform the required steps given in the Calling Profile for VOIP DN with service type softswitch section to assign a calling profile for VOIP DN with service type softswitch. When a remote agent calls, Feature Server chooses a calling profile based on the following priority sequence:
   1. Whether a calling profile is assigned to the agent. Otherwise, proceed to the next step.
   2. Whether a calling profile is assigned to the extension DN. Otherwise, proceed to the next step.
   3. Whether a calling profile is assigned to the VoIP Service DN with service-type=softswitch. Otherwise, proceed to the next step.
   4. Use internal caller's calling profile.
5. To provision Trunk DNs (optional): Feature server allows you to assign calling profiles to Trunk DNs similar to assigning calling profiles to Extension DNs.
   While processing inbound calls, Feature Server chooses the calling profile based on the following priority:
   1. Whether a calling profile is assigned to Trunk DN. Otherwise, proceeds to the next step.
   2. Use external caller calling profile.
6. Select the **Call Settings** tab to configure the DN call settings. Note that these settings do not apply when the DN is assigned, because the assignee settings override them.

Setting	Values (default value in bold)	Description
Forward All Calls To	**Off**, On + *phone number*	Immediately forwards all calls to the specified number. **Important:** To forward calls to voicemail, the number specified must be the

Setting	Values (default value in bold)	Description
		number configured as the VoIP DN (of service type `Voicemail`) for the associated switch. See Configure SIP Server for Feature Server.
Forwarding On No Answer	**System (Off)**, Off, On + *phone number*	After the No Answer Timeout value elapses, forwards calls to the specified number.
Forwarding On Busy	**System (Off)**, Off, On + *phone number*	When the user is on a call, forwards calls to the specified number.

7. Click Save changes.
8. To assign a mailbox to a DN, under **Configuration > System > Configuration Manager > Switching > DNs**, select a DN and create an option with **Section** `TServer`, **Key** `gvm_mailbox`, and a **Value** that is the mailbox number that you want to assign to the DN.

Bulk Assignment of Calling Profiles to DN

To assign calling profiles to DN, create a CSV file with the following fields.

Field	Field Description
DN number (M)	The number which uniquely identifies the DN.
Calling Profile ID (M)	The ID of the calling profile to be assigned.

Sample CSV file

```
100001,4b61f3a0-5332-41a2-94a7-362c47df6570
20000,56f310dd-ea86-48d8-9d21-c6d15842c296
```

Devices

You manage SIP desk phones from Polycom, AudioCodes, Genesys, and Yealink using the SIP Device Management area of Genesys Administrator Extension (GAX).

Device management supports dynamic model configurations through which phone models running on the certified firmware are supported by device management dynamically.

For the list of certified firmware, refer Genesys Supported Media Interfaces Guide.

The following are tested and recommended models:

- Polycom: SPIP_3xx, SPIP_4xx, SPIP_5xx, SPIP_6xx,VVX_3xx, VVX_4xx,VVX_5xx, VVX_6xx, VVX_15xx
- AudioCodes: 4xxHD, 4xx
- Genesys: 4xxHD, 4xx
- Yealink: SIP-TxxP

To add a model other than the models listed above, configure the model name in the **[dm]** section of the Master Feature Server Application object.
For example:

Option Name	Option Value
yealink	SIP-T48G
polycom	SSIP_7000

Important

- If the newly-added model has a UA Header Pattern that is different from the default supported matcher patterns, then you must configure the custom UA Headers in the **[dm]** section of all Feature Server Application objects.
- If you remove the model names that are configured in the **[dm]** section of the Master Feature Server Application object, the existing device with the corresponding model is supported as long as the device is present in Device Management.

Before you begin, verify that you have implemented device management.

Feature Server supports devices behind Session Border Controllers (SBCs) and firewalls. For these phones, you must set the configuration option **sip-preserve-contact** to `true`. For a single phone, set the option for its extension DN. If all phones on a site are behind an SBC or firewall, you can set the option at the application level.

Agents can change their ACD Agent State on desktop phones made by Polycom, AudioCodes/Genesys

(420HD model with firmware version 2.2.2 or higher only), and Yealink.

<tabber> Device profiles=

Device profiles are typically a collection of settings tied to a specific switch, enabling you to assign common settings to multiple devices. Create at least one profile for each SIP switch.

To create and manage device profiles:

1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Administration > SIP Device Management**, select **Profiles**.
2. To create a new profile, select **New**. To edit an existing profile, select it from the list. To create a new profile based on an existing profile, select a profile from the list (by clicking anywhere in the row other than the check box), then click **Clone** on the profile page.
3. Enter the profile details.

Tab	Field	Value
General	Profile Name	Because profiles are switch-specific, use the switch name as part of the profile name.
	Feature Server Application	Select the Feature Server application instance for the site this profile applies to.
	Default	Optionally, set this profile as the default profile when you use Interactive Voice Response (IVR) to provision new phones associated with the switch. If you set no default profile, IVR-provisioned phones use the most recent profile created.
SIP Server	Address and Port	Set the IP address (or FQDN) and port number of the SIP Server (or SBC address and port, for devices behind an SBC) associated with this profile.
	Transport	Select the transport protocol of your choice.
	Register	Specify whether phones need to register with the SIP Server.
	Registration Timeout	The duration, in seconds, of the registration, which is automatically renewed if the phone remains in service.
Outbound Proxy	Address, Port, Transport	Set the IP address (or FQDN) and port number of the Proxy Server (or SBC address and port, for devices behind an SBC) used in outbound calls. Select a transport type for outbound calls.

Tab	Field	Value
Voicemail	Voicemail Access Number	The phone number that a user dials to access voicemail.
Date and Time	NTP Server Address	Set the IP address (or FQDN) of your preferred NTP Server. Port number is not required.
	Time Zone	Select the time zone for all phones using this profile.
	Update Interval	The frequency, in hours, of synchronization of all phones using this profile. The default value is 24 hours, the maximum is 72.
Corporate Directory	LDAP Server Address and Port	Set the IP address (or FQDN) and port number of your LDAP server.
	Username and Password	The credentials needed to log into the LDAP server.
	Base Domain Name	The top level of the LDAP directory tree.
	Display Name	The display format of the returned search result.
	Name and Number Attributes	The name and number attributes of the LDAP records to be returned.
	Name and Number Filters	The search criteria for name and number lookups, respectively.
Call Settings		These options control a user's ability to set Do Not Disturb and call forwarding using the functions built into the phone itself. Even with these options set to `Disable`, your users can still control similar functions from within Feature Server. See Setting up your user profile.
	Do Not Disturb	Enables users to set their phones to Do Not Disturb.
	Call Forwarding	Enables users to set their phones to forward calls.
Logging	Syslog Server Address and Port	Set the IP address (or FQDN) and port number of your syslog server.
Security	Directory Path for Trusted Certificate	Set the path to the trusted certificate required for https and for secure communication between the phones and SIP Server.
License	Vendor	Select Polycom. Only Polycom phones that use LDAP, and have

Tab	Field	Value
		firmware below version 4.x, require a license.
	Directory Path For License File	Use the specified format to set the path to the license file provided by the phone vendor.
Business Continuity	Peer Switch	If your environment uses Business Continuity, select the peer switch of the secondary SIP Server.
	Peer SIP Server Address and Port	Set the IP address (or FQDN) and port number of the secondary SIP Server (or SBC address and port, for devices behind an SBC).
	Registration Mode	`Dual` is the default. You must change the value to `Single` when the ACD feature is enabled.
	Registration Timeout	The duration, in seconds, of the registration, which is automatically renewed if the phone remains in service.
	\Retransmission Timer	Modify this timer for quicker retransmission of SIP INVITE messages to a peer SIP server when the preferred SIP Server is down. Valid values: `20-200` milliseconds. Default: `50`. Enabled only in dual registration mode, and supported only in AudioCodes/ Genesys phones firmware version 2.2.8 or higher DONE.
Custom Configuration	Vendor	Select the vendor associated with the custom configuration.
	Configuration File	Upload a custom configuration file to set additional parameters for phones using this profile. The options set in this file are supplemental only, because the values explicitly set for the profile override the values for the same options in the configuration file. Whenever the custom files for the profile and the device contain the same parameter, the device values override the profile values. You must use the vendor-specific format for any parameters you enter in this file. Feature Server does not validate these parameters but does ignore all improperly formatted parameters.

Tab	Field	Value
	Override Profile configuration	To override the profile configurations in GAX, in the **Custom Configuration** tab of a profile, select **Override**. Important The order of precedence for the configuration file parameters sent to the device is as follows: 1. Device configuration 2. Profile custom configuration 3. Profile configuration
ACD (Automatic Call Distribution)	ACD	You can enable or disable the ability of all devices assigned to this profile to control agent login and logout access to the ACD queue, and allow agents to change their state to Ready, Not Ready, or After Call Work. You can also override this value for individual devices. For more information, see the **Devices** tab in this page. Enabling ACD in a profile automatically enables ACD for the first line in each device. To enable ACD for a different line, you must manually disable ACD on the first line and then enable ACD on the other line. See the notes following this table. Agent Login and State Update on SIP Phones details the related SIP Server functionality.
	Reason Codes	Not Ready reason codes let agents specify a reason (such as Lunch or Away) for setting themselves to Not Ready. You can create up to 50 codes of up to 5 digits each. **Important:** To ensure that your devices and Agent Desktop use the same Not Ready reason codes, you must use the same codes both here and in Workspace Desktop Edition (formerly Interaction Workspace).
Supplementary Services (Supported only in AudioCodes/Genesys phones firmware version 2.2.8 or higher)	Default Ringing Device	You can configure the phone to ring on the speaker, the headset, both speaker and headset, or not to ring at all.

Tab	Field	Value
	Hands Free Speaker Phone Mode	Enabled by default. When disabled, pressing the speaker button has no effect.
	Supervisor Listen In	Disabled by default. Enable to allow Supervisors to access an agent's handset (in Mute only mode), to listen in on a conversation that the agent is conducting on headphones with the customer.
	Agent Greeting	Disabled by default. Enable to allow Agents to record personal voice greetings directly on their phones, which are played to the customer and the agent when the agent attends the call.
	Override Device Configurations	By default, supplementary services configured in the device level takes higher priority (the disabled setting). Enable to give higher priority to the supplementary services configured in the profile level.

Important Notes about the ACD Feature (Agents can change their ACD Agent State on desktop phones)

- Not supported during bulk uploading.
- AudioCodes/Genesys phones require model 420HD and firmware version 2.2.2 or higher.
- Supported only on first line of AudioCodes/Genesys phones.
- For Business Continuity deployments:
  - Supported only in single registration mode.
  - For Polycom phones, configure single registration using an Fully Qualified Domain Name (FQDN) that resolves to 2 addresses that point to 2 separate SIP Servers (primary and peer). The FQDN must be configured in the SIP Server address field of the profile associated to the device, and the peer server's FQDN must *not* be configured in the profile.

|-| Devices=

Managing existing devices

1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Administration > SIP Device Management**, select **Devices**.
2. Select the devices you want to manage. To narrow your device list, you can search on the DN name, use **Advanced Search** to search on other attributes, or click **List Provisioned/List Unprovisioned** to toggle between the two lists.

> **Important**
> Advanced search displays the list of devices based on the provisioned or unprovisioned page. The MAC Address filter in Advanced search provides search results from both the provisioned and unprovisioned pages. combined.

3. Click **More** and select one of these actions:
   - **Associate Profile** associates the devices with an available profile, which you select from the **Select Profile** window.
   - **Resync** pushes settings to the devices.
   - **Restart** restarts the devices.
   - **Disable** disables the devices, effectively preventing their use.
   - **Enable** reactivates the devices.
   - **Clear Alerts** clears existing alerts from the device.

Deleting devices

1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Administration > SIP Device Management**, select **Devices**.
2. Select the devices you want to delete.
3. Click **Delete** to delete the selected devices.

Adding and modifying multiple devices

1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Administration > SIP Device Management**, select **Devices**.
2. To add and/or modify multiple devices simultaneously, click **Bulk Upload**.
3. Following the instructions in the **Bulk Upload** window, create and select a CSV file. Note these restrictions:
   - Limit each device to a maximum of four phone lines.
   - Limit each CSV file to a maximum of 5000 devices.
4. Select **Overwrite** to replace the settings of any existing devices that the CSV file includes. If you do not select overwrite, the upload ignores the CSV file's values for any existing devices.
5. Select **Resync** to resync the device once overwriting the devices are completed.
   Note that Resync option is enabled for the first line by default.
6. Click **OK**.

Adding and modifying individual devices

1. Log into GAX as an administrator (*GAX IP address*:*port*/gax). Under **Administration > SIP Device Management**, select **Devices**.
2. To create a new device, select **New**. To edit an existing device, select it from the list. To create a new device based on an existing device, select a device from the list (by clicking anywhere in the row other than the check box), then click **Clone** on the device page.
3. Enter the device details.

Tab	Field	Value
General	MAC Address	Type the device's unique MAC address.
	Vendor and Model	Select a supported device vendor and model.
	Profile Name	Type or select an appropriate device profile to associate with the device.
Logging	Logging	Enable or disable logging for the device. If enabled, **Click to view the device logs** stored in the **Syslog Directory Path** that you specify in **Settings > Logging**.
	Log Level	Select a logging level. The default is DEBUG.
License	Directory Path For License File	Use the specified format to set the path to the license file.
Custom Configuration	Configuration File	Upload a custom configuration file to set additional parameters for this device.
	Override Device configuration	Select the **Override** check box in the **Custom Configuration** tab of a device, to enable the parameters configured in the file to take precedence over the profile and the device configurations in GAX UI. Important When both profile and device configurations are set to Override, then the order of the precedence for configuration file parameters sent to the device is as follows: 1. Device custom configuration 2. Device configuration 3. Profile custom configuration

Tab	Field	Value
		4. Profile configuration
Lines	Lines	Add, remove, or modify lines for the device. Here you assign a DN to a device, and can specify a display name, typically the name of the user assigned to the line. For one line, check or clear the ACD check box to enable or disable the ability of the device to log agents into the ACD queue, log agents out, or allow agents to change their state to Ready, Not Ready, or AfterCallWork. This value overrides the ACD value set for the device profile. Enabling ACD in a profile automatically enables ACD for the first line in each device. To enable ACD for a different line, you must manually disable ACD on the first line and then enable ACD on the other line. Agent Login and State Update on SIP Phones details the related SIP Server functionality.
Supplementary Services (Supported only in AudioCodes/Genesys phones firmware version 2.2.8 or higher)	Default Ringing Device	You can configure the phone to ring on the speaker, the headset, both speaker and headset, or not to ring at all.
	Hands Free Speaker Phone Mode	Enabled by default. When disabled, pressing the speaker button has no effect.
	Supervisor Listen In	Disabled by default. Enable to allow Supervisors to access an agent's handset (in Mute only mode), to listen in on a conversation that the agent is conducting on headphones with the customer.
	Agent Greeting	Disabled by default. Enable to allow Agents to record personal voice greetings directly on their phones, which are played to the customer and the agent when the agent attends the call.

Assigning extensions (DNs) to devices

As detailed in the previous task, you can assign an extension to a device by specifying a DN for a line.

If enabled, you can also assign an extension by accessing an IVR (Interactive Voice Response) menu

from the device itself:

1. Connect the device to the network.
2. Lift the phone handset, which automatically dials the IVR number.
3. Follow the IVR prompts to assign an extension to the device.

> **Important**
> DNs must be unique across all sites (switches) in your Feature Server environment.

DTMF Tones Generation

AudioCodes/Genesys phones can generate DTMF tones when generation is requested by the agent through the agent desktop.

To configure SIP Server to remotely control DTMF generation on the SIP phone: In the TServer section of the DN object, configure: `sip-cti-control=dtmf`.

|-| Firmware=

Upgrading firmware

To upgrade (or downgrade) device firmware:

1. Create a shared directory in the file system, which must have read permission to the user under which Feature Server is running.
2. Copy the firmware files received from vendor to the shared directory and use this path to upgrade the firmware of the devices in GAX UI.
3. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Administration > SIP Device Management**, select **Firmware**.
4. Select one or more devices to upgrade or downgrade. To narrow your device list, you can search on the DN name, use **Advanced Search** to search on other attributes, or click **List Provisioned/List Unprovisioned** to toggle between the two lists.
5. Click **Upgrade**.
6. In the **Upgrade Firmware** window, use the specified format to set the path to the upgrade file.
7. Optionally, click **Schedule Upgrade** to set a specific date and time for the upgrade. Otherwise, upgrades begin immediately.
8. Click **OK**.

|-| Settings=

To modify settings that apply across device profiles, devices, and firmware:

1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Administration > SIP Device**

Management, select **Settings**.

2. Enter the settings.

Tab	Field	Value
General	Notification Delay	To avoid overwhelming the server with numerous simultaneous NOTIFY and HTTP requests during a reboot or resync of multiple phones, you can set a notification delay, in milliseconds. Default: 0. Valid values: 0 – 1000.
IVR (Interactive Voice Response)	Enable IVR Provisioning	Enable an administrator to use an IVR system to assign an extension to a device.
	IVR Admin Passcode	Type an integer of no more than six digits.
	IVR Number	The number to be dialed to trigger the IVR provisioning system. This number must be the same as the number specified during the IVR provisioning deployment. See Implement device management.
	BC Associations	SIP Business Continuity deployment supports IVR-based device provisioning only for mirrored DNs that are associated with a Disaster Recovery profile. In the **DN List** field, enter a comma-separated list of DN ranges or single DNs (or a mix). For example: 7000-8000,9001. Select the appropriate DR profile in the **DR Profile** field, to associate your list with it. **Notes:** • Validation is not performed for DN ranges in the list; the DNs that you enter are accepted as-is. • Changes to an association are applied only to new devices that are not yet provisioned; the changes do not affect existing, provisioned devices. • If a particular DN has more than one association, only the first matching association is used during disaster recovery. • IVR Provisioning switches to

Tab	Field	Value
		the Peer site automatically if the Preferred site fails.
Firmware	Max Simultaneous Upgrades	Set the maximum number of simultaneous upgrades allowable on each Feature Server. Default: 5. Valid values: 1-100. Recommended value: 20. You can calculate maximum number of simultaneous upgrades according to this formula: `Maximum number of simultaneous upgrades = number of active Feature Server instances * maximum-simultaneous-upgrades` For example, in the case of an active-active FS, the maximum number of simultaneous upgrades is 10 (2 * 5 = 10).
	Status Reset Timer	Set the time, in minutes, after which the firmware state resets from Completed to Idle. Default: 60. Valid values: 1-1440.
	Firmware Upgrade Timeout	Set the maximum allowed completion time of a firmware upgrade, in minutes. Default: 15. Valid values: 1-300. **Polycom VVX phones only:** When upgrading from Polycom firmware 5.0.0 to 5.0.1, set the timeout to 30.
	Maximum Bandwidth	The maximum network bandwidth, in Mbps, allotted to firmware upgrades. Default value = 0, Maximum value = 8192. You can determine a sufficient maximum bandwidth based on the amount of time it takes devices to download firmware, according to this formula: `Total time needed to download firmware (seconds) = Firmware size (MB) / Maximum bandwidth (Mbps) * Number of devices to be upgraded.` For example, to download firmware to 100 devices with 25 MB of firmware in 25 seconds, you would need to allot 100 Mbps of bandwidth ((25/100)*100 = 25). Note that the installation of firmware varies by vendor, so total upgrade time varies accordingly.

Tab	Field	Value
	Firmware Request Timeout	Set the maximum time, in minutes, after a device receives a notification from Feature Server that a new firmware is available before the device is expected to request firmware. Default value: 2 Valid values: `1-300`
Logging	Syslog Directory Path	Use the specified format to set the path to the device logs. If you use the recommended NXLOG logging server, point to the **\log_deposit** directory created during the device management implementation.

Voicemail

Voicemail administration in Feature Server consists of three main areas:

- Mailbox provisioning sets the characteristics of individual voice mailboxes, including activation, password reset, and time zone.
- Notifications inform users, through email or web services, of voicemail deposits.
- Profiles enable you to assign voicemail retention limits and notification settings to user groups or specific collections of users.

[+] Supported languages

You can deposit, read, or forward voicemail messages in the following languages.

- English (United States) ('en-US')
- English (UK) ('en-GB')
- English (Australia) ('en-AU')
- Brazilian Portuguese ('pt-BR')
- Simplified Chinese (Mandarin) ('zh-CN')
- Japanese ('ja')
- Spanish (Mexico) ('es-MX')
- Spanish (Spain) ('es')
- German ('de')
- Italian ('it')
- French (France) ('fr')
- Russian ('ru')
- Korean ('ko')

Voicemail configuration

To set up voicemail deposit:

- Log into Genesys Administrator Extension as an administrator (*GAX IP address*:*port*/gax). Under **Operations > SIP Voicemail & Call Settings > Voicemail**, select **Settings**.

General

- To enable voicemail deposit, select **Yes**. To disable voicemail deposit, select **No**.

Setting the value to No prevents administrators and users from changing the value for a specific mailbox.

Notification Defaults

Feature Server uses these values for SMTP and HTTP notifications when the corresponding field is not set in the Notifications tab of the Voicemail Profile associated with the user or user group. To set the default values, see Notification defaults.

Next, you provision mailboxes and set up voicemail profiles.

Provisioning mailboxes

Mailbox provisioning signifies configuring the characteristics of an individual voice mailbox that includes activation, password reset, language, and time zone settings.

To provision mailboxes:

1. Log into GAX as an administrator (*GAX IP address*:*port*/gax). Under **Operations > SIP Voicemail & Call Settings**, select **Mailboxes**. You can take either of these actions:
2. Search for and click the mailbox that you want to provision.
3. You can configure these settings for the selected mailbox:

Setting	Values (default value in bold)	Description
Status	**Active**, Locked	If the mailbox owner tries to log in unsuccessfully four times, they are locked out for 10 minutes. You can override the lock by selecting Active.
Mailbox Password	**System**, *user-selected*	Press Reset to reset the password to the system (default) value. Default value is a mailbox number.
Max Messages	**10**, 1 to *n*	Select the second radio button and type a value to set a new maximum number of messages. Select System to restore the system (default) value.
Optout Phone	**System (Not Set)**, *any phone number or routing point*	When set, enables a caller to transfer out of voicemail to the specified destination at any time during a call. Select the second radio button and type a value to set a new optout phone. Select System to restore the value to the number in parentheses, which is the value set at the application or switch level for the configuration option `voicemail-optout-destination`. For more details, refer to Voicemail Opt Out.
Time Zone	**System**, *time zone from menu*	Select a time zone from the menu to set a new time zone for all mailboxes that use the system (default) time zone. Select System to restore the system value.
Language	**System (English(United States))**, *language from menu*	Select a language from the menu to set a new language. Select System to restore the system value.
Assigned	n/a	The Assigned table lists the objects

Setting	Values (default value in bold)	Description
		(directory numbers, agents, users, user groups) to which this mailbox has been assigned. For some objects you can click the object name to view the object.
Messages	Unread/Read(Unread high-priority messages/Read high-priority messages)	Press **Delete All** to delete all normal and high-priority messages.
Voicemail Deposit Enabled	**Yes**, No	Setting Yes forwards unanswered calls to voicemail under various conditions, depending on the options set. Setting No plays the following message to the caller: **Sorry the service is temporarily not available goodbye**.

You can effectively deactivate a mailbox by setting its Status to Locked (which prevents user access), and by setting Voicemail Deposit Enabled to No (which prevents callers from depositing voicemail).

To include a disclaimer, press `Disclaimer` and upload an audio disclaimer message or other message to be played during every call, before message deposit. If you see the `Disclaimer is disabled` message, you must open Genesys Administrator and set the **play-disclaimer** option to `true` under the `Options` tab of the application or the governing switch, in the `VoicemailServer (Application object)` section or the `VoicemailServer (switch)` section. The value set at the application level overrides any value set at the switch level, so to enable a disclaimer for any switch, you must set the option to True for both application and switch. See Configuration options.

Disclaimer file types:

- For the United States and Japan, use CCITT uLaw (Mono) 64Kbps wav disclaimer files.
- For all other nations, use CCITT aLaw (Mono) 64Kbps wav disclaimer files.

Voicemail notifications

The feature supports three types of voicemail notifications:

- Message Waiting Indicator (MWI).
- Email (SMTP) operates through your standard SMTP email server.
- Web (HTTP) enables you to invoke web services to send notifications.

MWI, obviously, helps only when users are near their phones. SMTP and HTTP can notify practically anyone with internet access, and also include message information such as the caller's name, the time of the call, and the message priority.

For email and the web, you configure the notifications as part of a voicemail profile, though you also set default values for most fields.

> **Important**
> Users who log in dynamically on a phone that is not assigned to them do not receive email or web notifications for that phone's mailbox. Otherwise, users might receive notifications of voicemails left for someone else.

Enabling notifications

After you enable notifications in a voicemail profile, you can enable or disable either type of notification for each user or user group to which you've assigned the profile. Settings at the profile level also determine whether users can control notifications themselves. See Voicemail profiles for details.

SMTP and HTTP setup

To use SMTP to generate notifications, you must identify your SMTP server and security protocol by setting the SMTP configuration options in Genesys Administrator.

To use HTTP to generate notifications, you must set up your own web services. Then you identify the web service URL to Feature Server, which you do in the Notification defaults detailed below and in your Voicemail Profiles.

Notification defaults

You set notification defaults to serve as values that the system uses if the voicemail profile doesn't contain a value for a given field.

> **Important**
>
> These default values aren't enough to enable notifications. You can enable SMTP and HTTP notifications **only** through a voicemail profile.

To set notification defaults:

- Log into GAX as an administrator (*GAX IP address*:*port*/gax). Under **Operations > SIP Voicemail & Call Settings > Voicemail**, select **Settings**. Click **Notification Defaults** and complete the settings that apply to your chosen notification method (SMTP or HTTP):

Applies to	Setting	Values (default value in bold)	Description
SMTP	Email From Address	*user@domain*	The email address from which you want to send notifications. If you are using the TLS or SSL protocol, you must type the same address that you specified as the username in the SMTP configuration options section in Genesys Administrator.
SMTP	Email Subject	**Genesys Voicemail Notification: New Message from <CallerID>**, *any subject line*	The Subject line of the notification email. It can contain any of the parameter tokens available in the Email Message Body.
SMTP	Email Message Body	**Mailbox <MailboxID> has a new message from <CallerID>**, *any text*	The body of the notification email. It can contain any of the following parameter tokens, which are replaced by actual values in the delivered message. To insert a parameter, type **<** and select from the list, or type the tag name surrounded by brackets *<tagname>*. The message also includes any static text you type.

Applies to	Setting	Values (default value in bold)	Description
			• CallerID is the phone number of the caller • MailboxID is the mailbox that contains the message • MsgPriority is the message priority set by the caller, if enabled; emails that announce messages marked as Urgent are labeled High Importance (!). • MsgReceivedDate is the date on which the caller left the message, formatted according to the mailbox language • UserEmail is the email address of the recipient • UserPhone is the phone number of the recipient • VoicemailAccessURL is the URL that the recipient can click to retrieve their message online • VoicemailAccessNumber is the phone number that the user can dial to listen to their message
SMTP and HTTP	FS GAX URL	**none**, *GAX URL*	The Feature Server Genesys Administrator Extension URL forms the root of the Voicemail Access URL that the user clicks to go directly to the message. Use the format: `http://GAX server host:GAX server port/gax/?login#!`

Applies to	Setting	Values (default value in bold)	Description
HTTP	Web Service URL	**none**, *Web Service URL*	The URL of the web service you set up to handle HTTP notifications. Use the format: `http://host:port`
HTTP	Web Service Message Parameters	**CallerID=<CallerID>** **MailboxID=<MailboxID>** **MessagePriority=<MsgPriority>** **MessageReceivedDate=<MsgReceivedDate>** **UserEmail=<UserEmail>** **UserPhone=<UserPhone>** **AccessUrl=<VoicemailAccessURL>** **AccessNumber=<VoicemailAccessNumber>**	The parameter tokens available for the web notification message, formatted as key-value pairs (such as *caller=<CallerID>*). Other text is ignored. To insert a parameter, type **<** and select from the list, or type the tag name surrounded by brackets *<tagname>*. • CallerID (<CallerID> is the phone number of the caller • MailboxID (<MailboxID>) is the mailbox that contains the message • MailboxLanguage (<MailboxLanguage>) is the language specified for the mailbox, and governs the language of the included tokens • MessagePriority (<MsgPriority>) is the message priority (normal or urgent) set by the caller, if enabled • MessageReceivedDate (<MsgReceivedDate>) is the date on which the caller left the message • UserEmail (<UserEmail>) is the email address of the recipient

Applies to	Setting	Values (default value in bold)	Description
			• UserPhone (<UserPhone>) is the phone number of the recipient • VoicemailAccessURL (<VoicemailAccessURL>) is the URL that the recipient can click to retrieve their message online • VoicemailAccessNumber (<VoicemailAccessNumber>) is the phone number that the user can dial to listen to their message
HTTP	Web Service Name	**External Notification Service**, *any text*	The name you give to identify your web service; for example, Voicemail Notification Service. The user sees this name as a tab in their user profile.

Voicemail profiles

Voicemail profiles determine how long Feature Server keeps voicemails for a user or user group before deletion. Voicemail profiles use a Class of Service model to enable the quick assignment of voicemail notifications and retention limits to user groups or specific collections of users.

You can create profiles that set retention limits of 1 to 10,000 days, or use **`No Limits`** to set voicemails not to expire. You can effectively disable voicemail profiles by keeping the default Retention Limit value of the System Profile, **`No Limits`**, and assigning no other profile to your users.

> **Important**
> Remember not to assign a mailbox to multiple users or user groups.

The System Profile applies only when none of the users or user groups assigned to a mailbox has an assigned profile.

A new retention limit value in an assigned profile applies to all subsequent voicemail deposits in mailboxes associated with users and user groups the profile is assigned to. To apply a new value or a newly assigned profile to previously deposited voicemails, you run the Apply Message Retention Limits script. You can also run the script when you assign a new profile to a user or user group.

Managing voicemail profiles

To create and manage voicemail profiles:

1. Log into GAX as an administrator (***`GAX IP address:port`***`/gax`). Under **Operations > SIP Voicemail & Call Settings > Voicemail**, select **Profiles**.
2. To create a new profile, select **New**. To edit an existing profile, select it from the list.
   The Retention Limit column displays the limit for each profile, in number of days. The Email Notification and Http Notification columns display a check mark for the profiles that have those notification protocols enabled.
3. Uniquely name or rename the profile to identify the group or type of users to associate with it. For example: **`Tier 1 agents`** or **`Branch Office Wealth Managers`**. These names do not need to map directly to your defined user groups, because you can assign profiles to any individual or collection of users.
4. Set a Retention Limit, in number of days from **`1`** to **`10000`**. Select **No Limits** to retain voicemails until the user deletes them manually.
5. Optionally, set up email or web notifications.
6. Assign profiles to users and user groups.
7. If you have assigned new profiles or reassigned profiles and want to apply them retroactively, run the Apply Message Retention Limits script.

Assigning profiles

You can assign voicemail profiles directly to individual users or user groups, and to collections of users through Bulk Assignment.

Assign to...	Procedure
A single user	1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Operations > SIP Voicemail & Call Settings**, select **User**. Search for and click the user name of the user that you want to provision. 2. Select a Voicemail Profile from the menu.
A user group	1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Operations > SIP Voicemail & Call Settings**, select **User Group**. Search for and click the user name of the user group that you want to provision. 2. Select a Voicemail Profile from the menu.
Multiple individual users	1. Log into GAX as an administrator (***GAX IP address*:*port*/gax**). Under **Operations > SIP Voicemail & Call Settings**, select **User**. 2. Select **Voicemail Profile** from the **Bulk Assignment** menu. 3. Create and upload a csv file as instructed in the Bulk Assignment window.

Applying retention limits to existing voicemails

To apply new or changed retention limits to existing voicemails, you must run the **applyMessageRetentionLimits.py** python script. The script can run in two modes: information mode, which only reports on mailboxes and their associated voicemail profiles, and execution mode, which applies the new or changed retention limits. Information mode enables you to resolve conflicts (where one mailbox is associated with multiple users or user groups with different profiles) before running in the execution mode.

If you previously installed SIP Feature Server 8.1.201.18, you might see two similarly named retention limit scripts in the **python** folder. The correct script to use is **applyMessageRetentionLimits.py**.

> **Important**
>
> Because the retention limit for a mailbox depends on the associated profile of the longest duration, running this script can cause the immediate deletion of voicemails. For example, if you change a user's profile, lowering their retention limit from 100 to 30 days and resetting the longest retention limit of anyone associated with their mailbox to 30 days, running the script immediately deletes any voicemails older than 30 days from the mailbox.

See Python Scripts for information on how to deploy and run the **applyMessageRetentionLimits.py** script.

Sample command line

A sample command line to run the script:

```
java -jar <jython-version>.jar applyMessageRetentionLimits.py script input parameters
```

where the script input parameters are:

- **-H** Identifies the Cassandra host name (default: **localhost**)
- **-p** Identifies the Cassandra port (default: **9160**)
- **-o** Names the output file (default: **result.log**); changing this value enables you to store multiple log files
- **-e** Activates the execution mode
- **-y** Overwrites an existing output file of the name specified by the **-o** parameter (optional, but when you omit this parameter, the script does not successfully complete if an output file already exists)

Notifications

You set up Voicemail notifications in the Notifications tab.

To configure email and web notifications for a selected or new profile:

1. To enable email (SMTP) notifications, set **Email Notification Enabled** to **Yes**.
2. To allow users and user group administrators to turn email notifications on or off for themselves, and to specify the recipient email address, set **Email Notification Allow User Setup** to **Yes**. A value of **No** means that only administrators can control user and user group settings.
3. To enable web (HTTP) notifications, set **Web Service Notification Enabled** to **Yes**.
4. To allow users and user group administrators to turn web notifications on or off for themselves, and to specify the recipient phone number, set **Web Service Notification Allow User Setup** to **Yes**. A value of **No** means that only administrators can control user and user group settings.
5. Complete the notification settings that apply to your notification method, email (SMTP) or web (HTTP).

Applies to	Setting	Values (default value in bold)	Description
SMTP	Email From Address	*user@domain*	The email address from which you want to send notifications; if you are using the TLS or SSL protocol, you must type the same address that you specified as the username in the SMTP configuration options section in Genesys Administrator.
SMTP	Email Subject	**Genesys Voicemail Notification: New Message from <CallerID>**, *any subject line*	The Subject line of the notification email. It can contain any of the parameter tokens available in the Email Message Body.
SMTP	Email Message Body	**Mailbox <MailboxID> has a new message from <CallerID>**, *any text*	The body of the notification email. It can contain any of the following parameter tokens. To insert a parameter, type **<** and select from the list, or type the tag name surrounded by brackets *<tagname>*. The message also includes any static text you type. • CallerID is the phone number of the caller • MailboxID is the mailbox that contains the message • MsgPriority is the message priority set by the caller, if enabled • MsgReceivedDate is the date on which the caller left the message • UserEmail is the email address of the recipient • UserPhone is the phone number of the recipient

Applies to	Setting	Values (default value in bold)	Description
			• VoicemailAccessURL is the URL that the recipient can click to retrieve their message online • VoicemailAccessNumber is the phone number that the user can dial to listen to their message
HTTP	Web Service URL	**none**, *Web Service URL*	The URL of the web service you set up to handle HTTP notifications. Use the format: `http://host:port`
HTTP	Web Service Message Parameters	**CallerID=<CallerID>** **MailboxID=<MailboxID>** **MessagePriority=<MsgPriority>** **MessageReceivedDate=<MsgReceivedDate>** **UserEmail=<UserEmail>** **UserPhone=<UserPhone>** **AccessUrl=<VoicemailAccessURL>** **AccessNumber=<VoicemailAccessNumber>**	The parameter tokens available for the web notification message, formatted as key-value pairs (such as *caller=<CallerID>*). Other text is ignored. To insert a parameter, type **<** and select from the list, or type the tag name surrounded by brackets *<tagname>*. • CallerID (<CallerID> is the phone number of the caller • MailboxID (<MailboxID>) is the mailbox that contains the message • MailboxLanguage (<MailboxLanguage>) is the language specified for the mailbox, and governs the language of the included tokens • MessagePriority (<MsgPriority>) is the message priority

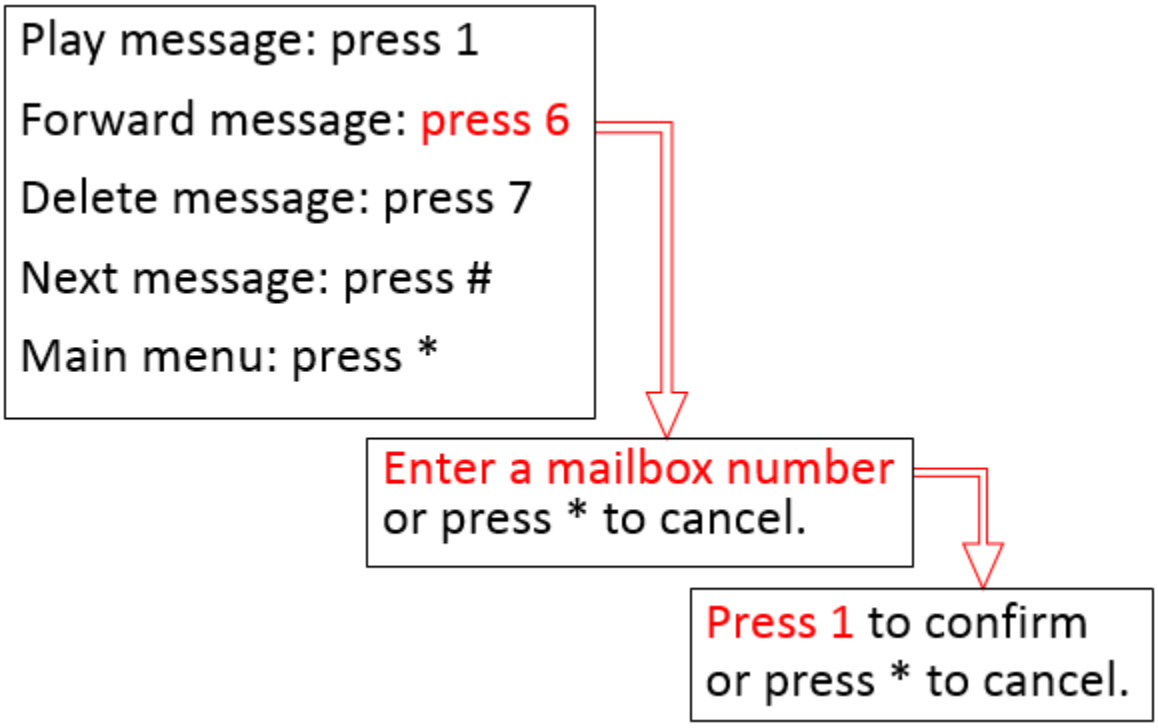
Applies to	Setting	Values (default value in bold)	Description
			(normal or urgent) set by the caller, if enabled • MessageReceivedDate (<MsgReceivedDate>) is the date on which the caller left the message • UserEmail (<UserEmail>) is the email address of the recipient • UserPhone (<UserPhone>) is the phone number of the recipient • VoicemailAccessURL (<VoicemailAccessURL>) is the URL that the recipient can click to retrieve their message online • VoicemailAccessNumber (<VoicemailAccessNumber>) is the phone number that the user can dial to listen to their message
HTTP	Web Service Name	**External Notification Service**, *any text*	The name you give to identify your web service; for example, Voicemail Notification Service. The user sees this name as a tab in their user profile.
SMTP and HTTP	FS GAX URL	**none**, *GAX URL*	The Feature Server Genesys Administrator Extension URL forms the root of the Voicemail Access URL that the user clicks to go directly to the message. Use the format: `http://GAX server host:GAX server port/gax/?login#!`
SMTP and HTTP	Voicemail Access Number	*any number*	The phone number that users dial to access

Applies to	Setting	Values (default value in bold)	Description
			voicemail. Sets the VoicemailAccessNumber parameter token that you can include in the email or web message body.

Voicemail Forwarding

You can use your telephone to forward voicemail messages left in your mailbox to any mailbox, when the option Voicemail Forwarding Enabled is set to yes.

How to Forward Voicemail



Begin at the message review menu on your telephone, after listening to a voicemail.

Press 6.

Enter a mailbox number.

Press 1.

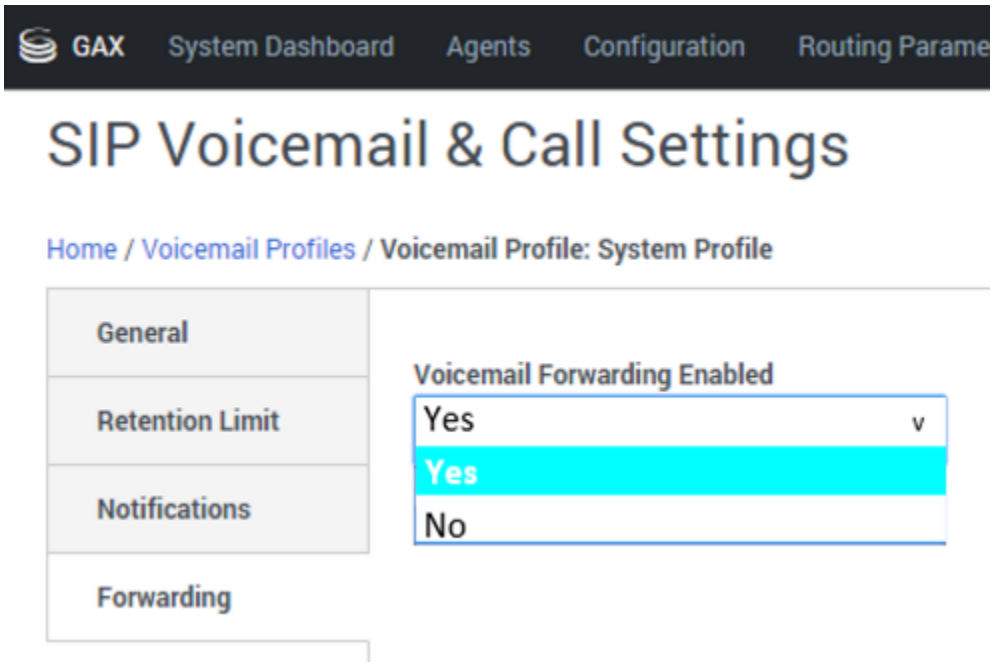
The voicemail is forwarded.

Notes

- Press * to cancel at any step.
- You cannot forward an expired voicemail.

- Forwarding a voicemail resets its retention limit in its destination mailbox.

Configuring Voicemail Forwarding



Use the Feature Server GAX plugin to configure voicemail forwarding.

Log in to GAX and go to

Voicemail Profiles › System Profile › Forwarding

Choose **Yes** to enable.
(The default is **No** -- disabled.)

Voicemail Profile Settings

The Voicemail Profile setting `Voicemail Forwarding Enabled` determines the state of this feature's functionality.

- Feature Server consults the Voicemail Profile of the User or User Group that the mailbox is assigned to.
- If the mailbox is not assigned, or assigned to multiple users or user groups with enable/disable settings that disagree, then Feature Server consults the System Voicemail Profile.

The above rules also determine which Voicemail Profile applies to the destination mailbox, and the retention limit setting in that Voicemail Profile is applied to the voicemail received.

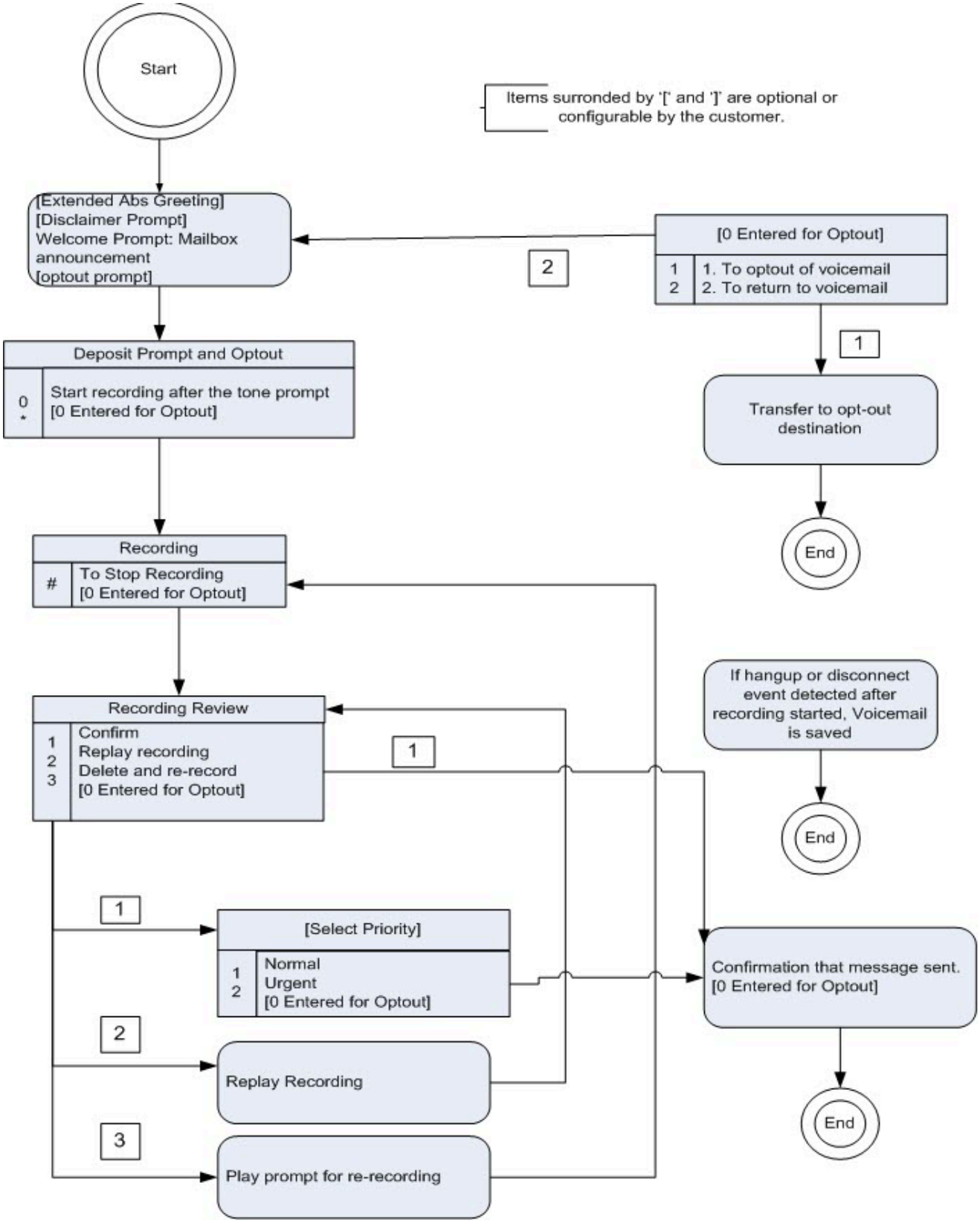
Voicemail Opt Out

The Opt Out feature provides an option for the user to exit the voicemail system and return to a configured destination on the mailbox, application, or switch. An Opt Out prompt is played after the optional disclaimer (like ‘Press zero at anytime to opt out of voicemail’).

Press ‘0’ to opt out of voicemail. The SIP Feature Server provides two options:

- Press 1 - to opt out of the voicemail and transfer to destination.
- Press 2 – to return to voicemail for depositing voicemail.

When the user selects option 1, the transfer out of voicemail to the configured destination is initiated. When the user selects option 2, the main menu of the voicemail is played. The following diagram illustrates the Telephone User Interface of the of the voicemail opt out feature:


Start
Items surronded by '[' and ']' are optional or configurable by the customer.
[Extended Abs Greeting]
[Disclaimer Prompt]
Welcome Prompt: Mailbox announcement
[optout prompt]
[0 Entered for Optout]
1 1. To optout of voicemail
2 2. To return to voicemail
2
1
Deposit Prompt and Optout
0 * Start recording after the tone prompt [0 Entered for Optout]
Transfer to opt-out destination
End
Recording
To Stop Recording [0 Entered for Optout]
If hangup or disconnect event detected after recording started, Voicemail is saved
Recording Review
1 Confirm
2 Replay recording
3 Delete and re-record
[0 Entered for Optout]
1
End
1
[Select Priority]
1 Normal
2 Urgent
[0 Entered for Optout]
Confirmation that message sent. [0 Entered for Optout]
2
Replay Recording
End
3
Play prompt for re-recording

Dial plan

The dial plan governs the disposition of inbound and outbound calls. If your Feature Server installation allows it, you can set up the dial plan in Genesys Administrator Extension. Otherwise, you must use the existing SIP Server dial plan functionality. See the Dial-Plan Rule section in the Framework 8.1 SIP Server Deployment Guide.

The Feature Server dial plan consists of:

- Partitions, known as "dial plan rules" in SIP Server 8.1.0 and before, which are the low-level building blocks of a dial plan, specifying criteria such as dialing patterns and effective times.
- Calling profiles, which consist of one or more partitions.
- Forwarding profiles, which set global call forwarding options.
- Dial plan settings, which set calling profiles and options for internal and external users and outbound calling.

Where multiple outcomes are possible, as in calling profiles and outbound routing rules, the application selects the closest match. If a caller, for example, dials 911, the application uses the rule with the pattern 911 rather than the pattern 9XX.

Call forwarding

Feature Server includes various options for call forwarding:

- No forwarding: a call rings until the recipient answers it or it times out.
- Forward calls to a single number.
- Forward calls to multiple destinations that can vary over time and day (known as Find Me Follow Me).
- Forward unanswered calls to a different number.
- Forward calls to a different number when all phones are busy.

Call forwarding consists primarily of two layers of options:

- Forwarding profiles set general forwarding options that you can apply to groups of users.
- User call settings and DN call settings control forwarding for users and phone extensions.

Disabling the Feature Server Dial Plan

If you are using the SIP Server dial plan rather than the SIP Feature Server dial plan, or you want to temporarily disable the Feature Server dial plan for some reason, you can set the configuration option **dialplan > active** to `false`, which takes effect after a restart (the default value is `true`). Setting this value to `false`:

- disables the Feature Server dial plan, and
- hides dial plan-related settings in the Feature Server GAX interface

Creating Partitions

If your Feature Server installation allows it, you can create and edit partitions in GAX. Otherwise, any dial plan settings you make in GAX are *not* used.

To create or edit a partition:

1. Log into Genesys Administration Extension as an administrator (*GAX IP address*:*port*/gax).
2. Under **Operations > SIP Voicemail & Call Settings > Dial Plan > Partitions**, click **New** to create a partition, or click the name of the partition that you want to edit.
3. Name or rename the partition. See the table below for examples.
4. To inactivate an existing partition without deleting it, uncheck **Active**.
5. Check **Block** to set this partition to bar calls that use the pattern specified in the rules.
6. Optionally, select a time zone. To default to the time zone of the call center, user, or device, select `Not Set`. To find time zone details, see List of tz database time zones.
7. Optionally, click the Time Start and Time End fields to select a time during which this partition is effective. Leave the Time Start at 00:00 and the Time End field empty to select the entire day.
8. Optionally, click the Days of Week field to select the specific days during which this partition is effective. Leave the field empty to select all days.
9. Type a Rule, using the patterns below as examples. Create a new partition for each pattern. To build your own patterns, see the Dial-Plan Rule section in the Framework 8.1 SIP Server Deployment Guide (note that the Dial Plan Parameters section does not apply to SIP Feature Server).

Name	Pattern
allow any call	.=>${DIGITS}
voicemail	5555=>gcti::voicemail (where 5555 is the number that users dial to access their voicemail)

10. Click **Save changes**.
11. Create calling profiles.

Exporting Partitions

1. Select the Partitions that must be exported.
2. Click **Export**.

A JSON file `callingpartitions.json` will be created with partitions configuration.

> **Important**
> The exported JSON file should not be modified.

Importing Partitions

1. Click **Import**.
2. Select the JSON file to be imported and click **Upload**.

Creating Calling Profiles

If your Feature Server installation allows it, you can create and edit calling profiles in GAX. Otherwise, any dial plan settings you make in GAX are *not* used.

To create or edit a calling profile:

1. Ensure that you have created the required partitions.
2. Log into GAX as an administrator (*GAX IP address*:*port*/gax).
3. Under **Operations > SIP Voicemail & Call Settings > Dial Plan > Calling Profiles**, click **New** to create a calling profile, or click the name of the profile that you want to edit.
4. Name or rename the calling profile.
5. To add a partition to the profile, click the plus (+) icon. Note that the order of the partitions matters only if the partitions have the same weight (an equal number of alphanumeric characters). Otherwise, the profile checks the least specific partitions first.
6. To delete a partition from the profile, select the partition and click the trash icon.
7. Create additional calling profiles as needed. To configure voicemail, create at least one voicemail-specific calling profile (such as `voicemail-profile-id`) and add your `default-partition` and `voicemail` partitions to it.
8. Edit dial plan settings.

Exporting Calling Profiles

1. Select the Calling Profiles that must be exported.
2. Click **Export**.

A JSON file **callingprofiles.json** will be created with partitions configuration.

> **Important**
> The exported JSON file should not be modified.

Importing Calling Profiles

1. Click **Import**.

2. Select the JSON file to be imported, and click **Upload**.

Bulk Upload of Calling Profile

To create multiple calling profiles with partitions, the following details must be provided as a CSV file. This table contains the field details that must be included in each row of the CSV file:

Field	Field Description	Default/Optional Values
calling profile name (M)	The name of the calling profile to be created.	
partition name (M)	The name of the partition the calling profile uses.	
partition rule (M)	The rules configured under the partition.	
partition active (O)	Inactivate/activate an existing partition.	**TRUE**, FALSE
partition block (O)	Allows partition to bar calls that use the pattern specified in rules.	**FALSE**, TRUE
partition time zone (O)	Select the required time zone	**Not set**, Any time zone name
partition time start (O)	Start time from which this partition is effective.	**00:00**, Any valid hours and minutes
partition time end (O)	End time till which this partition is effective	**Empty**, Any valid hours and minutes greater than start time.
partition days of week (O)	Select the specific day on which this partition is effective.	**Empty**, Sun, Mon, Tue, Wed, Thu, Fri, Sat

Sample CSV File

- To create a calling profile with multiple partitions, you must add one row per partition in the CSV file.
- The following CSV file will create two calling profiles and two partitions each.

```
"local calling profile ","Default Partition","1000=>1001"
"local calling profile","Voicemail","5555=>gcti::voicemail","TRUE","FALSE","Not
Set","00:00","05:19","Wed"
```

Important

- You cannot create multiple days of week in partitions using bulk upload of calling profiles.
- When you perform the bulk upload process for large number of calling profiles (for example, 1000), the bulk upload report will not be generated. However, the calling profiles and partitions will be created.

Creating Call Forwarding Profiles

Use forwarding profiles to set general forwarding options that you can apply to groups of users.

To create or edit a forwarding profile:

1. Log into GAX as an administrator (*GAX IP address*:*port*/gax).
2. Under **Operations > SIP Voicemail & Call Settings > Dial Plan > Forwarding Profiles**, click **New** to create a forwarding profile, or click the name of the profile that you want to edit.
3. Enter the profile details. The **Profile ID** is set automatically when you save a new profile.

Field	Value
Profile Name	Name the profile to distinguish it from other profiles.
Call Settings Enabled	No disables call forwarding. All calls ring through according to the dial plan. Users cannot view any call forwarding options.
Use External Destinations	Yes enables call forwarding to any phone number. No restricts destinations to recognized internal phone numbers only.
Maximum Number of Destinations	Sets the maximum number of destination sets available in Find Me Follow Me forwarding.
Find Me Follow Me Enabled	Yes enables calls to be forwarded to multiple destinations according to the Find Me Follow Me rules set for each user.

Editing Dial Plan Settings

If your Feature Server installation allows it, you can edit dial plan settings in Genesys Administration Extension (GAX). Otherwise, any dial plan settings you make in GAX are *not* used.

To edit the top-level dial plan settings:

1. Log into Genesys Administration Extension as an administrator (*GAX IP address*:*port*/gax).
2. Under **Operations > SIP Voicemail & Call Settings > Dial Plan**, select a switch.
3. Ensure that you have created the required calling profiles.
4. Select a System Internal Calling Profile to govern what happens when a call arrives from an internal user or DN (including internal remote users). Note that call settings assigned at the user or DN level take precedence over the value assigned here. The default Calling Profile is defined on the Switch level and cannot be directly associated with the user. If a Calling Profile is not set for a user or a device, then the dial plan that corresponds to the switch used in the dial plan is applied.
5. Select an External Caller Calling Profile to govern what happens when a call arrives from an external source.
6. Configure Outbound Routing Rules to specify the physical route to take for external destinations.
   - Click **New** to add a routing rule, or click a rule name to edit the rule. Name and define the rule, which is a pattern that must match the outbound dialed number; for example, `91XXXXXXXXXX=>${DIGITS}` means that all 12-digit numbers (beginning with 91) dial the route groups that you specify in the next substep. Note that you can perform digit manipulation at the same time as trunk selection. If multiple trunk DNs share the same prefix, SIP Server performs load balancing.
   - Click in the Groups field to assign one or more route groups to the routing rule. Select a group from the menu, or type the group name. Add groups in order of priority: the server attempts to route the call through the first route group that you add, then tries the next group until it succeeds.

     Groups that are defined in Outbound Routing Rules must be matched to prefix options that are configured on Trunk DNs. Read about prefix option configuration in the section "Agent Login–Level and DN-Level Options" of *Chapter 7: SIP Server Configuration Options* in the SIP Server Deployment Guide.
7. Click **Call Settings** to set the default values for this switch:

Setting	Values (default value in bold)	Description
Reject Call On Not Ready	**Off**, On	Rejects call when a user is not ready on a device. **Note:** When this option is set to 0n, any DN affected by this option can only receive a call if there is a user logged into that DN and Agent's Status for that user is set to **Ready**. Calls get rejected if they arrive on DNs that are not associated with any users or the user isn't logged in and not in the **Ready** state.
Call Waiting	**On**, Off	Does not reject a call when the user or device is already in a call.
No Answer Timeout	**30 sec**, 5 to 60 seconds (in 5-second intervals)	Specifies the length of time, in seconds, that Feature Server waits

Setting	Values (default value in bold)	Description
		for the user to answer a ringing call.
Forwarding On No Answer	**Off**, On + *phone number*	After the No Answer Timeout value elapses, forwards calls to the specified number.
Forwarding On Busy	**Off**, On + *phone number*	When the user is on a call, forwards calls to the specified number.

Note: User settings have a higher priority than dial plan settings.

8. Click **Save changes**.

Dial Plan Administration

The Dial Plan Administration feature enables you to display or hide the Dial Plan section in the **SIP Voicemail and Call Settings** menu in GAX.
You can add this feature to the GAX application by adding the `fs-admin-access-privileges` option in the GAX application. For more information, see Enable Dial Plan Administration.
After adding this feature to the GAX application, you can assign or revoke access to roles. Depending on the assigned privilege, the Dial Plan section is visible or hidden to the respective roles. See Assign privilege.

> **Important**
>
> This feature is available in Genesys SIP Feature Server Plugin release GAX 8.1.200.72 or later.

Dial Plan section

Enable Dial Plan Administration

Add the following option in GAX application to enable the Dial Plan Administration privilege:

[fs-gax-plugin]\fs-admin-access-privileges=FS_DIALPLAN_ADMIN

> **Important**
>
> If you do not add this feature to the GAX application, then all users who have the **Adminstrative Access to Genesys SIP Feature Server** privilege can view the **Dial Plan** section.

Assign privilege

You can assign this privilege to selected roles.
To assign privilege:

1. Open GAX Configuration.
2. Go to **Home** > **Roles** > **Roles** > **New Properties** and select **Assigned Privileges**.
3. To enable access, select the **Dial Plan Administration** check box. Clear the check box to revoke access.
4. Click **Save**.

Maintenance

Maintenance tasks include starting, stopping, backing up, and updating Feature Server.

> **Important**
> In multisite, multiple data center environments, complete synchronization of data does not occur unless and until all Feature Server instances are running.

Starting Feature Server

To start and verify SIP Feature Server:

> **Warning**
> Do not start Feature Server until you have set the configuration options *replicationStrategyClassName* and *replicationOptions*. See Cassandra options.

1. To run Feature Server in secure (https) mode:
   - Open the `start.ini` file and uncomment `etc/jetty-ssl.xml`
   - In the IVR Profile, set `initial-page-url = https://`*`Feature Server IP address or host name`*`:8443/fs`
2. Use Genesys Administrator, not the command line, to start SIP Feature Server. If you are running more than one Feature Server, start the Master first.
3. In Genesys Administrator, verify that the Feature Server is running.
4. Verify that the GAX interface is running by logging in as the Default administrator (in other words, the Default user in Configuration Server):
   `GAX IP address``:`*`port`*`/gax`
5. At this point, only the Default administrator can log into the Feature Server GAX interface. To enable other users to log in as administrators, assign the Administrator role to them.

Stopping Feature Server

To stop Feature Server, use Solution Control Interface (SCI), not the command line. If you are running more than one Feature Server, stop the Master only after you have stopped all non-Master servers.

Important

- Stop Feature Server before disconnecting the network. Failing to do so can cause Windows to terminate unexpectedly.
- Feature Server version 8.1.201.94 and above supports graceful stop, in which Feature Server waits for all ongoing voicemail calls to complete. This waiting period can be configured by using the `suspending-state-timeout` configuration options. See Configuration options for related information.

Upgrading Feature Server

Use this procedure to upgrade from one version of SIP Feature Server 8.1.2 to another.

> **Important**
>
> While upgrading Feature Server, **launcher.xml** will not be updated and the options in **launcher.xml** remain unchanged.

> **Important**
>
> Beginning with SIP Feature Server version 8.1.203.XX, the **Embedded Cassandra cluster mode** is being deprecated and the feature will be completely removed in future versions. As part of this deprecation, the Embedded Cassandra cluster mode will be removed from the default installation in future versions. If your current deployment model uses the Embedded Cassandra cluster mode for SIP Feature Server, Genesys recommends you to migrate the deployment to other Cassandra modes.

Upgrading while Feature Server is running (recommended)

To upgrade a running Feature Server environment, stop and upgrade one Feature Server Cassandra cluster instance at a time, **beginning with the master Feature Server**.

1. On the Feature Server master node (which is also the Cassandra seeds node, in case you are running the embedded Cassandra cluster), back up all files in the **etc** folder, which includes the `cassandra.yaml` file.
2. Stop Feature Server.
3. Install Feature Server from the installation package. During the upgrade, the installer uses the values provided during a fresh installation.
4. Start Feature Server.
5. Repeat steps 2-4 for each Feature Server instance.
6. Upgrade or install and configure the Feature Server GAX Plug-in.
7. If you are updating a Feature Server 8.1.200.88 environment that also uses the Feature Server dial plan, you must run a migration script.
8. Optionally, create and assign voicemail profiles.

Upgrading while Feature Server is stopped

To upgrade a running Feature Server environment, upgrade one Feature Server Cassandra cluster instance at a time. Upgrade the master Feature Server last.

1. On the Feature Server master node (which is also the Cassandra seeds node, in case you are running the embedded Cassandra cluster), back up all files in the **etc** folder, which includes the `cassandra.yaml` file. Do **not** upgrade the master node until after you have upgraded all other nodes.
2. On a non-master node, install Feature Server from the installation package. During the upgrade, the installer uses the values provided during a fresh installation.
3. Start Feature Server.
4. Repeat steps 2-3 for each Feature Server instance.
5. Upgrade or install and configure the Feature Server GAX Plug-in.
6. If you are updating a Feature Server 8.1.200.88 environment that also uses the Feature Server dial plan, you must run a migration script.
7. Optionally, create and assign voicemail profiles.

Restore HTTPS configuration

The following procedure shows how to restore HTTPS configuration of Feature Server after an upgrade. This procedure is applicable only while upgrading Feature Server to version 8.1.201.92 or above.

1. After upgrading to version 8.1.201.93 or above, the Feature Server installation folder contains the following files retrieved from the folder containing the previous versions: **start.ini.bak** and **etc.bak** in both Windows and Linux operating systems. Previously, when upgrading Feature Server overwrote these files rather than retrieving existing files.
2. Compare and copy the difference in the values of **start.ini**, **start.ini.bak** and **etc/jetty-ssl.xml**, **etc.bak/jetty-ssl.xml** and apply the differences in **start.ini** and **etc/jetty-ssl.xml**.
3. Copy **etc.bak/keystore** to **etc/keystore** to restore the configuration.

> ### Important
> After upgrading to Feature Server version 8.1.201.92, the backup of the **etc** folder and the **start.ini** file will be named as **start.ini.backup** (Windows), **start.ini.bak** (Linux), **etc.backup** (Windows), and **etc.bak** (Linux), respectively.

Post upgrade steps for version 8.1.203 and later

If you have upgraded SIP Feature Server to version 8.1.203 and later and used external Cassandra in

your deployment, you can deactivate the Embedded Cassandra module from the deployment. Note that deactivating the Embedded Cassandra module is recommended but it is an optional step.

To deactivate the Embedded Cassandra module:

1. Locate the **start.ini** file in the path: **<FS installation folder>/start.ini**.
2. Open the file with a text editor and remove the line **--module=fs-cass11**.
3. Save the file.
4. Restart SIP Feature Server if it is running.
5. After restart, remove the installation files from the **<FS installation folder>/lib/fs-cass11** folder.

Switching from Thrift to CQL protocol for Cassandra communication

If you have upgraded SIP Feature Server to version 8.1.203 and later and used external Cassandra in your deployment, you can switch over to use the new CQL communication protocol by referring the following procedure:

- Make sure the Cassandra cluster is configured to accept connections on the CQL port (9042).
- Make sure you have python3 interpreter available on all the SIP Feature Server host as per prerequisites.
- In all the SIP Feature Server application,
  - Configure the **connection-type=**`cql` option in the **Cassandra** section of the application object
  - Configure the **process-launcher** option in the **python** section pointing to the Python 3.x executable.
  - Restart the SIP Feature Server node.

Post switchover steps after transitioning from Thrift to CQL protocol

Complete the following steps after you switched over the communication protocol from Thrift to CQL. Note that this procedure is recommended but it is optional.

1. Deactivate the Thrift library by referring the procedure here.
2. Deactivate the embedded jython module by referring the following procedure. The embedded jython is used for execution of dialplan and maintenance scripts, which is now replaced with the native python interpreter.

> **Important**
>
> Before deactivating the embedded jython module, ensure the Python executor is configured in the application options under the **python** section.

1. Locate the **start.ini** file in the path: **<FS installation folder>/start.ini**.
2. Open the file with a text editor and remove the line **`--module=fs-jython`**.
3. Save the file.
4. Restart SIP Feature Server if it is running.
5. After restart, remove the installation files from the **<FS installation folder>/lib/fs-jython** folder.

Re-initialize Feature Server backend

- Exporting data not stored in Configuration Server
- Cassandra cleanup
- Reimporting Feature Server specific data

This procedure enables you to recover corrupted configuration data. This is done by exporting configuration data to CSV files, clearing the corrupted data from the Cassandra database (the SIP Feature Server's backend), and then reimport it back to Cassandra. The import process cleans the corrupt data.

Summary

In some cases you might want to re-populate data into Cassandra, because the data in Cassandra is either corrupted, out-of-sync, or you want to migrate/create a new instance of the database required for another environment. Note that the procedures given in this article does not guarantee exact replication of Cassandra data and are not intended to replace the standard Cassandra backup and recovery procedure. However, these steps can help you to refresh provisioning data maintained by Feature Server and allow you to review, adjust, and re-populate Feature Server specific data (such as voicemails and Dialplan settings).

Warning

- Running this procedure might result in data loss. Therefore, Genesys recommends to have a backup at Cassandra level before you proceed if you plan to re-populate the same backend. You can also execute only specific steps, for example, you can export specific data or re-synchronize only configuration related content.
- If you want to execute export and cleanup steps, make sure Feature Server is made read-only mode using the respective configuration option before proceeding. After you complete the cleanup steps, make sure to revert Feature Server back to its full operational mode even before you restart the application, which is needed to begin configuration synchronization steps.

High level plan

Here is the high-level plan if you want to make changes to Cassandra data - export, cleanup and re-sync, and re-import data back to Cassandra.

EXPORT: Run scripts that export **User Roles**, **User Voicemail Profile assignments**, **User Group Voicemail Profile Assignments**, **User Calling Profile Assignments**, and **Device Calling Profile Assignments**.

CLEANUP and RE-SYNC: Run the Cassandra cleanup script. It forces Feature Server to perform the initial import steps at the next start. The cleanup script removes ALL configuration data in the Cassandra database that was previously sourced from Configuration Server during initial import and by processing real-time updates received.

> ### Warning
> The Cassandra cleanup script also *removes* data that you want to keep such as **User roles** and **User associations with Calling and Voicemail** profiles, as well as **User** and **User Group settings** such as email notification settings and Dial Plan Forwarding settings. Hence, you must export the data first and then restore each of those data sets in order to preserve the data. You must restart Feature Server and wait for it to complete initialization.

RE-IMPORT: Run scripts that restore **User Roles**, **User Voicemail Profile assignments**, **User Group Voicemail Profile Assignments**, **User Calling Profile Assignments** and **Device Calling Profile Assignments** after running the Cassandra cleanup script.

Exporting data not stored in Configuration Server

Exporting Feature Server data not stored in Configuration Server requires script deployment and execution, followed by a Cassandra cleanup procedure.

The following Python scripts save data, such as User roles, User Voicemail Profiles assignment, and User Group Voicemail Profiles assignment, that is not related to or synchronized with Configuration Server.

Each script creates a csv file that you can analyze, edit as needed, and use as the input data for the scripts restoring the data not contained in Configuration Server.

See Appendix: Feature Server Maintenance Python Scripts for Embedded Cassandra for information on how to deploy and run the script.

User Feature Server Roles

The User Feature Server Roles script creates a csv file containing records for all users with Roles different from the default User role. The csv file later serves as the input for the Restoring User Roles procedure.

Every record of the user roles csv file contains user name, user ID, and the corresponding set of roles assigned to the user. User ID consists of the corresponding person DBID and Configuration Server GUID separated by ‘@’; for instance: 57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9

csv file content example:

User name	User ID	Assigned roles
un00001	57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9	User,AdministratorGroupMailboxAdministrator
un00003	57428@dcc7a7ac-626a-40c7-b805-e14b71d438d9	User,GroupMailboxAdministrator
un00002	57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9	User,GroupMailboxAdministrator

For more information on usage of the **saveUserRoles.py** script, see Python Scripts.

User Voicemail Profile Assignments

The User Voicemail Profile Assignments script creates a csv file containing records for all users with a Voicemail Profile other than the one assigned to them by **System Profile**. The csv file later serves as the input for the Restoring User Voicemail Profile Assignments procedure.

Every record of the user voicemail profile csv file contains a user name, user ID, and the corresponding ID of a Voicemail Profile assigned to the user. User ID consists of the corresponding person DBID and Configuration Server GUID separated by '@'; for instance: 57426@cb2fdedd-a57f-49e6-a54d-3f930eb1dfc5

csv file content example:

User name	User ID	Voicemail Profile ID
un00002	57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9	11451ec2-d68a-4425-98eb-fbf22a24fc7a
un00001	57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9	924b7d17-38d9-493d-8727-004625e0a112
un00001	57426@f521b229-f599-47d4-81fd-2fbf15b02809	11451ec2-d68a-4425-98eb-fbf22a24fc7a
un00003	57428@f521b229-f599-47d4-81fd-2fbf15b02809	924b7d17-38d9-493d-8727-004625e0a112

For more information on usage of the **saveUserVmProfiles.py** script, see Python Scripts.

User Group Voicemail Profile Assignments

The User Group Voicemail Profile Assignments script creates a csv file containing records for all user groups with a Voicemail Profile other than the one assigned to them by **System Profile**. The csv file later serves as the input for the Restoring User Group Voicemail Profile Assignments procedure.

Every record of the user group voicemail profile csv file contains a group name, group ID and corresponding ID of a Voicemail Profile assigned to the user group. User group ID consists of the corresponding Agent Group DBID and Configuration Server GUID separated by '@'; for instance: 19613@dcc7a7ac-626a-40c7-b805-e14b71d438d9

csv file content example:

Group name	Group ID	Voicemail Profile ID
ag002	19613@dcc7a7ac-626a-40c7-b805-e14b71d438d9	924b7d17-38d9-493d-8727-004625e0a112
ag002	19613@f521b229-f599-47d4-81fd-2fbf15b02809	11451ec2-d68a-4425-98eb-fbf22a24fc7a
ag001	19612@f521b229-f599-47d4-81fd-2fbf15b02809	924b7d17-38d9-493d-8727-004625e0a112

For more information on usage of the **saveUsergroupVmProfiles.py** script, see Python Scripts.

User Calling Profile Assignments

The User Calling Profile Assignments script creates a csv file containing records for all users with a Calling Profile assigned to them. The csv file later serves as the input for the Restoring User Calling Profile Assignments procedure.

Every record of the user calling profile csv file contains a user name, user ID, and the corresponding ID of a Calling Profile assigned to the user. User ID consists of the corresponding person DBID and Configuration Server GUID separated by '@'; for instance:
57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9

csv file content example:

User name	User ID	Calling Profile ID
un00002	57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9	074b6d5a-63859a-4e13-9ef2-63884f88a99c
un00001	57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9	e969a3be-6337-4041-8a9c-a270843c6529

For more information on usage of the **saveUserCallingProfiles.py** script, see Python Scripts.

Device Calling Profile Assignments

The Device Calling Profile Assignments script creates a csv file containing records for all devices with a Calling Profile assigned to them. The csv file later serves as the input for the Restoring Device Calling Profile Assignments procedure.

Every record of the device calling profile csv file contains a device ID and the corresponding ID of a Calling Profile assigned to the device. Device ID consists of corresponding DN number and the switch name the device belongs to, separated by '@'; for instance: 10001@SwitchSA01.

csv file content example:

Device ID	Calling Profile ID
20001@SwitchSA02	24e06da6-1dd3-479a-a0a2-0db2c9aa767c
10001@SwitchSA01	a4ea866b-7dcc-4da6-97ae-24cb14f2e150

For more information on usage of the **saveDeviceCallingProfiles.py** script, see Python Scripts.

Cassandra cleanup

Cassandra cleanup is a prerequisite for Configuration Server data reimport. Without the cleanup no reimport occurs.

> **Important**
> Before performing this task, you **must** perform the Cassandra backup procedure. If the Feature Server recovery process does not succeed, you can restore the Cassandra data.

The Cassandra cleanup procedure consists of the following steps:

1. On the master Feature Server node, run the python script preparing Feature Server for synchronization with Configuration Server. The script truncates the Configuration Database synchronization-related column families to enable Configuration Server data reimport later.
   For more information on usage of the **cleanupColumnFamilies.py** script, see Python Scripts.
2. Perform the keyspace flush operation using nodetool (see Backing up Cassandra data).

Reimporting Configuration Data

Configuration Server data reimport

1. Stop and restart the master Feature Server node. When the master node starts it imports switch objects (DNs and Agent Logins) and tenant objects (Persons, Agent Groups, and Places).
2. Stop and restart all the other Feature Server nodes, one server at a time. Start each Feature Server only after the previous node has finished starting. Each Feature Server node imports the switch objects (DNs and Agent Logins) assigned to it.

> **Important**
> If multiple Feature Server nodes are assigned to the same switch, first restart one Feature Server node per switch, then proceed with restarting the rest of the nodes.

Synchronization without full reload

You can perform synchronization without forcing full reload, for example, to catch up with events missed during prolonged Feature Server outage. You can do so without doing full cleanup of database by following these steps:

1. Point your browser on the master Feature Server to this URL: `http://<fsserverhost>:<port>/fs/api/admin/reimport/init`
2. Log in as an administrator to initiate the reimport process.
3. To monitor progress, point your browser on the master Feature Server to this URL: `http://<fsserverhost>:<port>/fs/api/admin/reimport/state`
4. Log in as an administrator and display the current reimport state, which can be one of these two:
   - In progress

- Ready to start (Reimport is considered complete when the status is in 'Ready to start'.)

Feature Server does incremental re-sync of configuration data automatically to recover from sync issues when you set the option **[cluster] reimport-on-conf-history-log-error** to `true` in the master Feature Server application.

Reimporting Feature Server specific data

Run the scripts that restore the non-Configuration Server specific data that you saved by running the backup scripts above.

Restoring data not stored in Configuration Server

To restore data that are not stored in Configuration Server, run the following scripts using the output files created by export process.

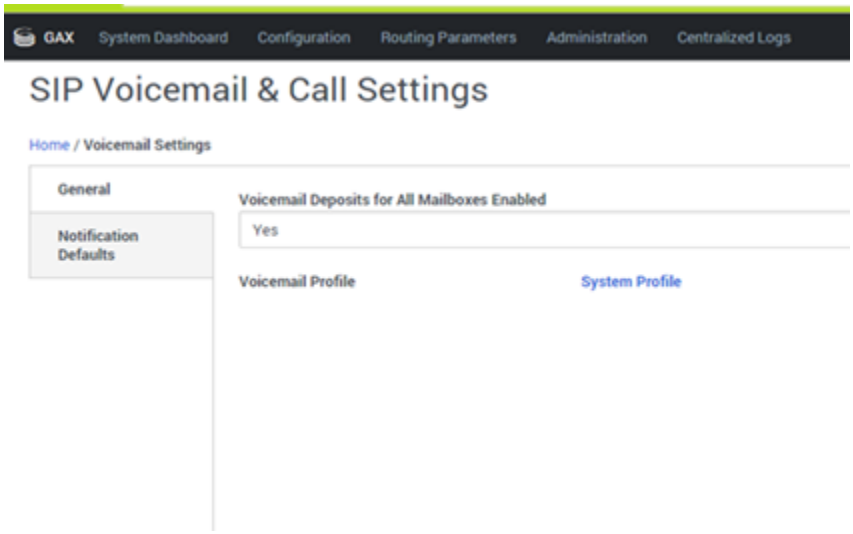
- **restoreUserRoles.py**
- **restoreUserVmProfiles.py**
- **restoreUsergroupVmProfiles.py**
- **restoreUserCallingProfiles.py**
- **restoreDeviceCallingProfiles.py**

For more information on usage of each of these scripts, see Python Scripts.

Enable / Disable Voicemail Deposits

Turn On/Off Voicemail Deposits for All Mailboxes is both a privilege and functionality. Installing Genesys SIP Feature Server Plugin for GAX 8.1.200.56 or higher adds this functionality to the GAX Server, and only users with the privilege can enable and disable it.

About the Privilege



If you have the privilege

Go to the Settings page for Feature Server, in Genesys SIP Feature Server Plugin for GAX. You'll see and be able to modify the selection **Voicemail Deposits for All Mailboxes Enabled** if you have the privilege.

Note: The embedded administrator account **default** always has this right—it cannot be removed.

If you do not have the privilege, and you have Feature Server version 8.1.201.67 or higher

- The **Voicemail Deposits for All Mailboxes Enabled** setting is visible but read-only.

If you have Feature Server version 8.1.201.66 or Lower

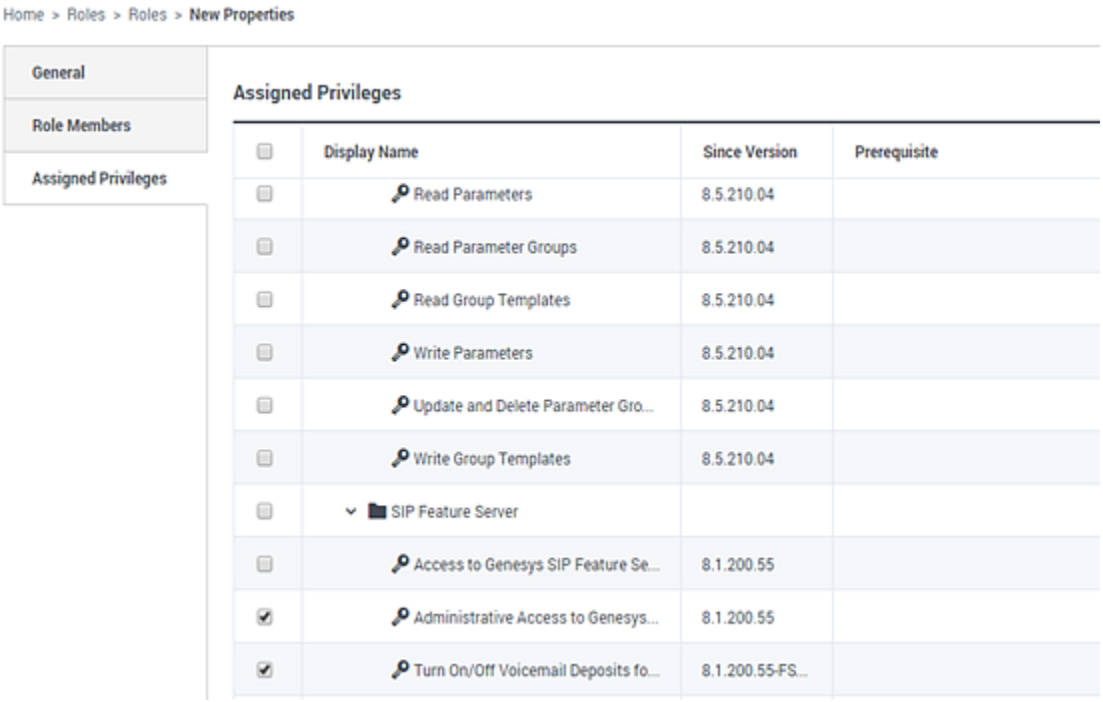
- The **Voicemail Deposit Enabled** setting is editable even for the user who does not have this privilege.

Assigning the Privilege

1. Assign this privilege to a Role that has administrative access to Genesys SIP Feature Server.
2. Assign that Role to Users or Access Groups.

Any user or member of an access group that has been assigned the role will have the **Turn On/Off Voicemail Deposits for All Mailboxes** privilege.

Enabling the Privilege



1. Open GAX Configuration.
2. Go to `Home > Roles > Roles > New Properties` and select **Assigned Privileges**.
3. To enable, select the checkbox for **Turn On/Off Voicemail Deposits for All Mailboxes**.

   To disable, clear the checkbox.

Check and Refresh Mailbox Counters

> **Important**
>
> Genesys recommends you to set the **cassandra-counter** option to `false` to allow the latest version of Feature Server manage counters automatically. Use the instructions in this article only if you are currently using the version 8.1.202.18 or earlier or cannot set up the recommended option.

Feature Server release 8.1.201.80 includes the following python scripts:

- **refreshMailboxCounters.py**—Renews or resets mailbox counters.
- **getAllMailboxCountersInfo.py**—Checks mailbox counters.

You can run python scripts on the master Feature Server instance to:

- correct an error that caused Feature Server to display incorrect totals for the number of messages in a mailbox.
- check the mailbox counters.

See Appendix: Feature Server Maintenance Python Scripts for Embedded Cassandra for information on how to deploy and run the python scripts.

Check Mailbox Counters

1. Verify that:
   - All Feature Servers are up and running.
   - Ensure that you set Feature Server in read-only mode using corresponding options to ensure no deposits or retrievals are taking place.
2. Run `getAllMailboxCountersInfo.py`.

Refresh Mailbox Counters

1. Verify that:
   - All Feature Servers are up and running.
   - Ensure that you set Feature Server in read-only mode using corresponding options to ensure no deposits or retrievals are taking place.
2. Run getAllMailboxCountersInfo.py to identify mailboxes having invalid counters.
   These mailboxes will be marked with `!!!` in the script output log. For example:

   ```
   Mailbox: 86025: Read value: '-1/0 (0/0)'; Calculated value: '0/0 (0/0)';  !!!
   ```

3. Create a .csv file which has the list of mailboxes that need to be refreshed.
4. Run the Python script `refreshMailboxcounters.py` using the input file created.

The content of the file csv file created in step 3 should be in this format:

```
4909045
4909185
4909005
4909889
```

Run as a scheduled task

If you are using Feature Server version 8.1.202.00 or above then you can schedule the update-mailbox-counters task as described in the Scheduled maintenance tasks page.

Remove Metadata of Expired Voicemail Messages

Feature Server release 8.1.201.88 includes the following python script:

- **removeExpiredMessages.py**—Removes expired messages.

You can run this python script on the master Feature Server instance to remove metadata of expired voicemail messages.

See Python Scripts for information on how to deploy and run the **removeExpiredMessages.py** script.

Remove metadata of expired voicemail messages

1. Ensure that all Feature Servers are up and running.
2. Run removeExpiredMessages.py.

Run as a scheduled task

If you are using Feature Server version 8.1.202.00 or above then you can schedule the delete-expired-messages task as described in the Scheduled maintenance tasks page.

Managing Feature Server Cassandra backend

Maintain Cassandra backend by following the recommendations that is suitable for the version of database that you use. You can find these instructions from Cassandra's official documentation, refer to an example instruction from here. If you are using embedded Cassandra in Feature Server, refer to Appendix: Performing maintenance operations on embedded Cassandra for its maintenance instructions.

Similarly, for backing up and restoring Cassandra data, follow the recommendations that is suitable for the version of database that you use from their official documentation. If you are using embedded Cassandra in Feature Server, refer to Appendix: Backing up and restoring embedded Cassandra data for its backup and restore instructions.

Python Scripts

> **Important**
> The maintenance scripts, described in this article, can only work with the External Cassandra deployment and require External Cassandra to have an open CQL port. Additionally, there are alternative versions of scripts, located in the **python/util** folder that are available for environments with Embedded Cassandra and/or for migration steps between Embedded and External Cassandra up to version 3.11. Python2 (or embedded jython module) is required to run these alternative scripts.

Supported script functions and parameters

Script location: **python/tools**

Before running the maintenance scripts

- Python3 interpreter is installed and it is up and running to run scripts.
- For running scripts, make sure that the path to folder **python/cassandra** is specified and the Python interpreter can find the Cassandra driver module.
- Before running the scripts, note the following:
  1. Set the environment variable **PYTHONPATH**, for example: **PYTHONPATH= <fs_location_dir>/python**
  2. Run the scripts from **<fs_location_dir>/python folder**
     An example command line looks like:

     ```
     python ./tools/<script_name> <parameters_list>
     ```

Including TLS specific arguments

If you are connecting a Cassandra server in a secured mode (TLS enabled), you must use the **--tls** option in the script command line for all scripts (except few scripts, which require a configuration JSON file for starting and configuring TLS there).

Additionally, add the path to the CA certificate file as an argument of the **--cert** option.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of the target Cassandra database.	CassNode01	Yes

Parameters	Description	Sample	Mandatory
Port (-p)	The CQL port of the target Cassandra database.	9042	Yes
TLS Mode (--tls)	Activates the TLS connection mode.	---tls	Yes
Certificate (--cert)	The full path to the location of SSL certificate file.	--cert /home/user/ssl/certs/ca.pem	Yes

Sample: `python ./script.py -H localhost -p 9042 --tls --cert /home/user/ssl/certs/ca.pem`

Modifying the default keyspace name (optional)

All scripts operate with the column families of the default keyspace name **sipfs**. If you want to define an alternative keyspace name, you can do so by specifying the new keyspace name as an argument of the **--keyspace** command line option.

Sample: `python ./script.py -H localhost -p 9042 --keyspace sipfs_new` where, `sipfs_new` is the new keyspace name.

Saving and restoring content of the column families

The maintenance scripts previously used for backing up a single column (**saveColumnFamily.py**, **restoreColumnFamily.py**) were removed in the CQL version. You can now do the same using the following CQL commands that is comparatively more simpler and efficient:

```
 cqlsh -u <username> -p <password> --keyspace <keyspace> -e "COPY <column_family_name> TO
'<path_to_backup_file>';"
 cqlsh -u <username> -p <password> --keyspace <keyspace> -e "COPY <column_family_name> FROM
'<path_to_backup_file>';"
```

Sample: For example, to backup and restore the **device** column family from the **sipfs** keyspace, use the following command:

```
saving of the column family content to CSV backup file:
    cqlsh -u user -p pass --keyspace sipfs -e "COPY device TO './backup_devices.csv';"
restoring of the column family content back from CSV file
    cqlsh -u user -p pass --keyspace sipfs -e "COPY device FROM './backup_devices.csv';"
```

applyMessageRetentionLimits.py

Functions:

- Changes the retention limits of existing voicemails.
- This script can be run in two modes:
  - **information mode** - which only reports on mailboxes and their associated voicemail profiles.
  - **execution mode**- which applies the new or changed retention limits.

- Information mode strives to resolve conflicts (where one mailbox is associated with multiple users or user groups with different profiles) before running in the execution mode.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of the target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of the target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Run Mode (--exec)	Activates the execution mode.	--exec	No
Output (-o)	The file name for the output log file with path.	**./cleanupColumnFamilies.log**	Yes

Sample:

```
python ./applyMessageRetentionLimits.py -H localhost -p 9042 -o
./applyMessageRetentionLimits.log
```

cleanupColumnFamilies.py

Functions: Terminates the configuration database synchronization-related column families to enable Configuration Server data reimport later.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of the target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	The file name for the output log file with path.	**./cleanupColumnFamilies.log**	Yes

Sample:

```
python ./cleanupColumnFamilies.py -H localhost -p 9042 --keyspace sipfs_new -o
./cleanupColumnFamilies.log
```

copyKeyspaceColumnFamilies.py

Functions: Copies the content of source keyspace column families to the destination keyspace column families.

Parameters: Use parameters in **copyKeyspaceInput.json** and pass it as an input file. This file exists in the same path where this python script is located.

Parameters	Description	Sample	Mandatory
sourceHost	Host of source Cassandra database.	FsNode01	Yes
sourcePort	The CQL port of source Cassandra database.	9042	Yes
destinationHost	The host name of the destination Cassandra database.	CassNode01	Yes
destinationPort	The CQL port number of the destination Cassandra database.	9042	Yes
sourceKeyspace	The source keyspace name.	sipfs	Yes
destinationKeyspace	The destination keyspace name.	sipfs_new	Yes
excludedCFs	A comma-separated list of column family names to be excluded from copying while running the '**copyKeyspaceColumnFamilies.py** script.	message_bytes, device	No
includedCFs	A comma-separated list of column family names to be copied while running the **copyKeyspaceColumnFamilies.py** script.	message_bytes, device	No
sourceHostUserName	The username of the user accessing source Cassandra.	FSadmin	Yes, if authentication is enabled in the source Cassandra cluster.
sourceHostPassword	The password of the user accessing source Cassandra.	FSadmin	Yes, if authentication is enabled in the source Cassandra cluster.
sourceHostTls	Set this option to `true` when SSL is enabled for the source Cassandra connection.	`true`	Yes, if SSL is enabled in the source Cassandra.
sourceHostCert	The path to the source CA certificate file.	**/home/certs/ca.pem**	Yes, if SSL is enabled in the source Cassandra.
destinationHostUserName	The username of the user accessing	FSadmin	Yes, if authentication is enabled in the

Parameters	Description	Sample	Mandatory
	destination Cassandra.		destination Cassandra cluster.
destinationHostPassword	The password of the user accessing destination Cassandra.	FSadmin	Yes, if authentication is enabled in the destination Cassandra cluster.
destinationHostTls	Set this option to `true` when SSL is enabled for the destination Cassandra connection.	`true`	Yes, if SSL is enabled in the destination Cassandra.
destinationHostCert	The path to destination CA certificate file.	**/home/certs/ca.pem**	Yes, if SSL is enabled in the destination Cassandra.

Sample:

```
python ./copyKeyspaceColumnFamilies.py -i ./copyKeyspaceInput.json -o
./copyKeyspaceContent_`date +%y%m%d-%H:%M`.log
```

copyKeyspaceSchema.py

Functions: Creates a keyspace and its column families in the destination Cassandra cluster copied from the source keyspace.

Parameters:

Use parameters in **copyKeyspaceInput.json** and pass it as input file. This file exists in the same path where this python script is located.

Parameters	Description	Sample	Mandatory
sourceHost	The host name of source Cassandra database.	FsNode01	Yes
sourcePort	The CQL port number of source Cassandra database.	9042	Yes
destinationHost	The host name of destination Cassandra database.	CassNode01	Yes
destinationPort	The CQL port number of destination Cassandra database.	9042	Yes
sourceKeyspace	The name of the source keyspace.	sipfs	Yes
destinationKeyspace	The name of the destination keyspace.	sipfs	Yes
excludedCFs	A comma-separated list of column family names to be excluded from	message_bytes, device	No

Parameters	Description	Sample	Mandatory
	copying while running the **copyKeyspaceColumnFamilies.py** script.		
includedCFs	A comma-separated list of column family names to be copied while running the **copyKeyspaceColumnFamilies.py** script.	message_bytes, device	No
sourceHostUserName	The username of the user accessing source Cassandra.	FSadmin	Yes, if authentication is enabled in the source Cassandra cluster.
sourceHostPassword	The password of the user accessing source Cassandra.	FSadmin	Yes, if authentication is enabled in the source Cassandra cluster.
sourceHostTls	Set this option to `true` when SSL is enabled in the source Cassandra connection.	`true`	Yes, if SSL is enabled in the source Cassandra.
sourceHostCert	The path to source CA certificate file.	**/home/certs/ca.pem**	Yes, if SSL is enabled in the source Cassandra.
destinationHostUserName	The username of the user accessing destination Cassandra.	FSadmin	Yes, if authentication is enabled in the destination Cassandra cluster.
destinationHostPassword	The password of the user accessing destination Cassandra.	FSadmin	Yes, if authentication is enabled in the destination Cassandra cluster.
destinationHostTls	Set this option to `true` when SSL is enabled in the destination Cassandra connection.	`true`	Yes, if SSL is enabled in the destination Cassandra.
destinationHostCert	The path to the destination CA certificate file.	**/home/certs/ca.pem**	Yes, if SSL is enabled in the destination Cassandra.
replicationStrategyClassName	The name of the replication strategy class.	`SimpleStrategy`, `NetworkTopologyStrategy`	No, `SimpleStrategy` is used by default.
replicationOptions	The value of the replication map or replication factor.	`{"usw1": 2}`, `1`	Yes, for `NetworkTopologyStrategy` No, for `SimpleStrategy` , (1 is set by default)

Replication strategy class and factor settings:

Class	Replication factor	Value Description
SimpleStrategy	replication_factor : N	Assign the same replication

Class	Replication factor	Value Description
		factor to the entire cluster.
NetworkTopologyStrategy	datacenter_name : N	Assign replication factors to each data center in a comma-separated list.

Sample:

```
python ./copyKeyspaceSchema.py -i ./copyKeyspaceInput.json -o ./copyKeyspaceSchema_`date
+%y%m%d-%H:%M`.log
```

exportMe.py

Functions:

Based on request, this script exports voicemail data in the **.wav** format from the Cassandra database in a client-understandable format.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (--dbhost)	The host name of target Cassandra database.	CassNode01	Yes
Port (--dbport)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (--fileLocation)	Input **.json** file name and its location.	**./in/export_me**	Yes
Output (--outputLocation)	Output result file name and its location.	**./out/export_me**	Yes

Sample:

```
python ./exportMe.py --dbhost localhost --dbport 9042 --fileLocation ./in/export_me --
outputLocation ./out/export_me
```

The following is a sample input JSON file:

```
{
  "caseid": "123456789",
  "consumers": [
    {
      "consumer": [
        { "name": "John Doe" },
        { "name": "John Q. Doe" },
        { "phone": "55551011" }
      ]
    },
    {
      "consumer": [
        { "name": "Dan Akroyd" },
```

```
        { "phone": "55551012" },
        { "phone": "555556162" },
        { "email": "danny@hollywood.com" },
        { "email": "funnyguy@comedy.org" },
        { "fbid": "Dan Akroyd" }
      ]
    }
  ],
  "gim-attached-data": { "kvlist": ["AcctNum", "SSN"] }
}
```

exportVoicemail.py

Functions:

To export Voicemail Mailbox Data, Greetings, Messages, and Message metadata.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output(-o)	Output folder name. It exports the following items: • Mailbox and mailmessage metadata - Export_Voicemail.json • Mail message wav files - messageid.wav file • Greeting wav file (if greeting is set)	**./exportVoicemail**	Yes
agent/agentgroup (--agent)	The export is done only for particular agent or agent group. The list should be provided as a **.csv** file.	**agents_agentgroups.csv**	No
TLS (--tls)	Enable if TLS is used.	NA	No

Sample:

To export all mailboxes, following is the sample script:

```
python exportVoicemail.py -H localhost -p 9042 -o voicemailExport
```

The following is a sample output JSON file:


```
{
    "1115": {
        "mailbox_settings": {
            "activeGreetingType": "Standard",
            "canRetrieve": "true",
            "depositState": "",
            "isAnswerOnlyOn": "false",
            "isEnrollmentDone": "false",
            "lastInvalidLoginTimestamp": "0",
            "lastLoginTimestamp": "0",
            "locale": "",
            "loginAttemptCount": "0",
            "maxMsgCount": "",
            "optoutPhone": "",
            "password": "",
            "reset": "false",
            "salt": "",
            "storageVersion": "5",
            "tenantDbid": "0",
            "timeZoneName": "",
            "unlockedTimestamp": "0"
        },
        "mailbox_greetings": {},
        "voicemail_messages": {
            "7617d1e3-e978-4457-8c9e-ae4d7ffe34d6": {
                "callerid": "799005",
                "callermailboxid": "799005",
                "duration": "2000",
                "fn": "1115-1713097934551.wav",
                "isnew": "true",
                "isprivate": "false",
                "priority": "MediumPriority",
                "retLimit": "",
                "storageVersion": "5",
                "timestamp": "1713097934740",
                "7617d1e3-e978-4457-8c9e-ae4d7ffe34d6": "7617d1e3-e978-4457-8c9e-
ae4d7ffe34d6.wav"
            }
        }
    },
    "111": {
        "mailbox_settings": {
            "activeGreetingType": "Standard",
            "canRetrieve": "true",
            "isAnswerOnlyOn": "false",
            "isEnrollmentDone": "false",
            "lastInvalidLoginTimestamp": "0",
            "lastLoginTimestamp": "0",
            "loginAttemptCount": "0",
            "reset": "false",
            "storageVersion": "5",
            "tenantDbid": "0",
            "unlockedTimestamp": "0"
        },
        "mailbox_greetings": {},
        "voicemail_messages": {
            "18e9f80b-41b2-4d71-93b8-a8291260b0cf": {
                "callerid": "799005",
                "callermailboxid": "799005",
                "duration": "2000",
                "fn": "111-1713097566464.wav",
```

```
                "isnew": "true",
                "isprivate": "false",
                "priority": "MediumPriority",
                "storageVersion": "5",
                "timestamp": "1713097567819",
                "18e9f80b-41b2-4d71-93b8-a8291260b0cf":
"18e9f80b-41b2-4d71-93b8-a8291260b0cf.wav"
            }
        }
    }
}
```

To export specific agent/agent group mailboxes, following is the sample script:

```
python exportVoicemail.py -H localhost -p 9042 -o voicemailExport --agents
agents_agentgroups.csv
```

The following is a sample input csv file:

```
Agent,Agent_Group
2345,
7894,
```

(Agent id 2345 has 111 mailbox, Agent id 7894 has 5687 mailbox)

The following is a sample output JSON file:

```
{
    "111": {
        "mailbox_settings": {
            "activeGreetingType": "Standard",
            "canRetrieve": "true",
            "isAnswerOnlyOn": "false",
            "isEnrollmentDone": "false",
            "lastInvalidLoginTimestamp": "0",
            "lastLoginTimestamp": "0",
            "loginAttemptCount": "0",
            "reset": "false",
            "storageVersion": "5",
            "tenantDbid": "0",
            "unlockedTimestamp": "0"
        },
        "mailbox_greetings": {},
        "voicemail_messages": {
            "18e9f80b-41b2-4d71-93b8-a8291260b0cf": {
                "callerid": "799005",
                "callermailboxid": "799005",
                "duration": "2000",
                "fn": "111-1713097566464.wav",
                "isnew": "true",
                "isprivate": "false",
                "priority": "MediumPriority",
                "storageVersion": "5",
                "timestamp": "1713097567819",
                "18e9f80b-41b2-4d71-93b8-a8291260b0cf":
"18e9f80b-41b2-4d71-93b8-a8291260b0cf.wav"
            }
        }
    },
    "5687": {
        "mailbox_settings": {
            "activeGreetingType": "Standard",
```

```
            "canRetrieve": "true",
            "depositState": "",
            "isAnswerOnlyOn": "false",
            "isEnrollmentDone": "false",
            "lastInvalidLoginTimestamp": "0",
            "lastLoginTimestamp": "0",
            "locale": "",
            "loginAttemptCount": "0",
            "maxMsgCount": "",
            "optoutPhone": "",
            "password": "",
            "reset": "false",
            "salt": "",
            "storageVersion": "5",
            "tenantDbid": "0",
            "timeZoneName": "",
            "unlockedTimestamp": "0"
        },
        "mailbox_greetings": {},
        "voicemail_messages": {
            "93e1fc1c-9ecf-44ec-a8f4-31508822ff3e": {
                "callerid": "799005",
                "callermailboxid": "799005",
                "duration": "2000",
                "fn": "5687-1713098709033.wav",
                "isnew": "true",
                "isprivate": "false",
                "priority": "MediumPriority",
                "retLimit": "",
                "storageVersion": "5",
                "timestamp": "1713098709192",
                "93e1fc1c-9ecf-44ec-a8f4-31508822ff3e": "93e1fc1c-9ecf-44ec-
a8f4-31508822ff3e.wav"
            }
        }
    }
}
```


exportVoicemailSettings.py

Functions:

Exports the Voicemail settings into a CSV file.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	Output folder name	**./exportVoicemailSettings**	Yes

Sample:

```
python ./exportVoicemailSettings.py -H localhost -p 9042 -o ./exportVoicemailSettings
```

forgetMe.py

Functions:

- Deletes the voicemail data of a customer when requested.
- Run this script on the master Feature Server instance to delete the voicemail data. The customer-related information received from the common Web UI will be transformed into a JSON input file.
- The **forgetMe.py** script will fetch the JSON files (that were added since the last execution time to fetch the ANI) added to the **gdpr-directory** option configured in the **[gdpr]** section of the master Feature Server.
- The script will then set the expiry duration to 21 days (by default) for voicemails that correspond to the ANI obtained. The voicemails will be deleted after the expiry duration.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (--dbhost)	The host name of target Cassandra database.	CassNode01	Yes
Port (--dbport)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Expiration Time (--expirationTime)	The voicemail data expiration time (seconds).	1814400	No
Input (--fileLocation)	Input JSON files location folder name.	**./forget_files**	Yes

Sample:

```
python ./forgetMe.py --dbhost localhost --dbport 9042 --fileLocation ./forgetMeFiles
```

The following is a sample input JSON file:

```
{
  "caseid": "123456789",
  "consumers": [
    {
      "consumer": [
        { "name": "John Doe" },
        { "name": "John Q. Doe" },
        { "phone": "55551011" }
      ]
    },
    {
      "consumer": [
        { "name": "Dan Akroyd" },
        { "phone": "55551012" },
        { "phone": "555556162" },
```

```
        { "email": "danny@hollywood.com" },
        { "email": "funnyguy@comedy.org" },
        { "fbid": "Dan Akroyd" }
      ]
    }
  ],
  "gim-attached-data": { "kvlist": ["AcctNum", "SSN"] }
}
```

getAllMailboxCountersInfo.py

Functions: Checks the mailbox counters.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Output (-o)	The name of the output file and its location.	**./getAllMailboxCountersInfo.log**	Yes

Sample:

```
python ./getAllMailboxCountersInfo.py -H localhost -p 9042 -o ./getAllMailboxCountersInfo.log
```

Before running the script verify the following:

- All Feature Servers are up and running.
- There is no voicemail activity (no one can deposit, read, or listen to voicemail).
- Then run the **getAllMailboxCountersInfo.py** script.

getColumnFamilyContent.py

Functions: Retrieves a list of rows (with particular columns content) from the target column family.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Column Family (-c)	The column name of the target column family.	device	Yes

Parameters	Description	Sample	Mandatory
Columns List (-k)	A list of target columns.	USER,NUMBER	No
JSON File (-j)	The file name of the resultant JSON file and its location.	**./results.json**	No
Output (-o)	The name of the output file and its location.	**./getColumnFamilyContent.log**	Yes

Sample:

```
python ./getColumnFamilyContent.py -H localhost -p 9042 -o ./getColumnFamilyContent.log -c
user -k ROLES,RESYNC_ID
```

getColumns.py

Functions: Retrieves the defined columns' values from the target column family.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Column Family (-f)	The column name of target column family.	user	Yes
Columns List (-c)	A list of target columns.	DB_VERSION,ROLES	Yes
Output (-o)	The name of the output file and its location.	**./getColumns.log**	Yes

Sample:

```
python ./getColumns.py -H localhost -p 9042 -o ./getColumns.log -f user -c DB_VERSION,ROLES
```

getMailboxCountAndLastLoginTime.py

Functions: Retrieves the list of mailboxes with assigned and last login date/time info.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra	9042	Yes

Parameters	Description	Sample	Mandatory
	database.		
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
JSON File (-j)	The file name of the resultant JSON file and its location.	**./results.json**	No
Output (-o)	The name of the output file and its location.	**./getMailboxCountAndLastLoginTime.log**	Yes

Sample:

```
python ./getMailboxCountAndLastLoginTime.py -H localhost -p 9042 -o
./getMailboxCountAndLastLoginTime.log
```

getUserKeys.py

Functions: Retrieves the list of users keys.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	The name of the output file and its location.	**./getUserKeys.log**	Yes

Sample:

```
python ./getUserKeys.py -H localhost -p 9042 -o ./getUserKeys.log
```

getUsers.py

Functions: Retrieves the list of users containing a column with a particular value.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes

Parameters	Description	Sample	Mandatory
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Column Key (-k)	Target column key.	DB_VERSION	Yes
Column Value (-v)	Target column value.	1	Yes
Output (-o)	The name of the output file and its location.	**./getUserKeys.log**	Yes

Sample:

```
python ./getUsers.py -H localhost -p 9042 -k DB_VERSION -v 1 -o ./getUsers.log
```

importVoicemail.py

Functions: To import Voicemail Mailbox Data, Greetings, Messages, and Message metadata.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
input directory (-i)	Directory where all the **.wav** files for voicemail messages are placed and the exported JSON file.	voicemailExport	Yes
Output directory (-o)	Directory where the execution log is stored.	importVoicemailReport	Yes
TLS(--tls)	Enable if TLS is used.	NA	No

Sample:

```
python importVoicemail.py -H localhost -p 9042 -i voicemailExport -o importVoicemailReport
```

importVoicemailSettings.py

Functions: Imports the Voicemail settings from the CSV file.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes

Parameters	Description	Sample	Mandatory
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (-i)	The name of the input folder where the CSV file is located.	**./inVoicemailSettings**	Yes
Output (-o)	The name of the output folder.	**./outVoicemailSettings**	Yes

Sample:

```
python ./importVoicemailSettings.py -H localhost -p 9042 -i ./importVoicemailSettings -o
./outputLogs
```

removeExpiredMessages.py

Functions: Removes expired messages.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Output (-o):	The name of the output file and its location.	**./removeExpiredMessages.log**	Yes

Sample:

```
python ./removeExpiredMessages.py -H localhost -p 9042 -o ./removeExpiredMessages.log
```

Before running the script:

- Ensure that all Feature Servers are up and running.
- Then run **removeExpiredMessages.py** script.

removeSwitchDescription.py

Functions: Removes the Switch description data from the database.

Parameters: Use parameters in **removeSwitchDescription.json** and pass it as an input file. This file exists in the same location where this script is located.

Parameters	Description	Sample	Mandatory
host	The host name of target Cassandra database.	FsNode01	Yes
port	The CQL port number of target Cassandra database.	9042	Yes
keyspace	The name of target keyspace.	sipfs	Yes
switchName	The name of target Switch.	SIP_Cluster	Yes
userName	The username of the user accessing the target Cassandra database.	FSadmin	Yes, if authentication is enabled in the Cassandra cluster.
password	The password of the user accessing the target Cassandra database.	FSadmin	Yes, if authentication is enabled in the Cassandra cluster.
hostTls	Set this option to `true` when SSL is enabled for the target Cassandra connection	`true`	Yes, if SSL is enabled for the Cassandra.
hostCert	The path to the Cassandra host CA certificate file.	**/home/certs/ca.pem**	Yes, if SSL is enabled for Cassandra.

Sample:

```
python  ./removeSwitchDescription.py -i ./removeSwitchDescription.json -o
./removeSwitchDescription.log
```

restoreDeviceCallingProfiles.py

Functions: Restores the Device Calling Profiles data that are not stored in Configuration Server (data saved by corresponding backup script).

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (-i)	The name of the input backup CSV file and its location.	**./savedDeviceCallingProfiles.csv**	**file**

Parameters	Description	Sample	Mandatory
Output (-o)	The name of the output file and its location.	**./restoreDeviceCallingProfiles.log**	Yes

Sample:

```
python ./restoreDeviceCallingProfiles.py -H localhost -p 9042  -i
./savedDeviceCallingProfiles.csv -o ./restoreDeviceCallingProfiles.log
```

The following is a sample input CSV file:

```
Device ID           Calling Profile ID                              * Note: The header row
should not present in the input CSV file content
------------------------------------------------------------
"20001@SwitchSA02", "24e06da6-1dd3-479a-a0a2-0db2c9aa767c"
"10001@SwitchSA01", "a4ea866b-7dcc-4da6-97ae-24cb14f2e150"
```

restoreUserCallingProfiles.py

Functions: Restores the User Calling Profiles data that are not stored in Configuration Server (data saved by corresponding backup script).

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (-i)	The name of the input backup CSV file and its location.	**./savedUserCallingProfiles.csv**	Yes
Output (-o)	The name of the output file and its location.	**./restoreUserCallingProfiles.log**	Yes

Sample:

```
python ./restoreUserCallingProfiles.py -H localhost -p 9042 -i ./savedUserCallingProfiles.csv
-o ./restoreUserCallingProfiles.log
```

The following is a sample input CSV file:

```
User name   User ID Calling                                  Profile
ID                                  * Note: The header row should not present in the
input CSV file content
-----------------------------------------------------------------------------------------------
"un00002",  "57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"0758d5a6-355a-4e13-9ef2-63884f88a99c"
"un00001",  "57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"e969a3be-6337-4041-8a9ca270843c6529"
```

restoreUsergroupVmProfiles.py

Functions: Restores the Usergroup Voicemail Profiles data that are not stored in Configuration Server (data saved by corresponding backup script).

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (-i)	The name of the input backup CSV file and its location.	**./savedUsergroupVmProfiles.csv**	Yes
Output (-o)	The name of the output file and its location.	**./restoreUsergroupVmProfiles.log**	Yes

Sample:

```
python ./restoreUsergroupVmProfiles.py -H localhost -p 9042 -i ./savedUsergroupVmProfiles.csv
-o ./restoreUsergroupVmProfiles.log
```

The following is a sample input CSV file:

```
Group name  Group ID                                        Voicemail Profile
ID                            * Note: The header row should not present in the input CSV file
content
-------------------------------------------------------------------------------------------------------
"ag002",    "19613@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"92a5fd17-2b4b-493d-8727-004625e0a112"
"ag002",    "19613@f521b229-f599-47d4-81fd-2fbf15b02809",
"11451ec2-d68a-4425-98ebfbf22a24fc7a"
"ag001",    "19612@f521b229-f599-47d4-81fd-2fbf15b02809",
"92a5fd17-2b4b-493d-8727-004625e0a112"
```

restoreUserRoles.py

Functions: Restores the User Roles data that are not stored in Configuration Server (data saved by corresponding backup script).

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes

Parameters	Description	Sample	Mandatory
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (-i)	The name of the input backup CSV file and its location.	**./savedUserRoles.csv**	Yes
Output (-o)	The name of the output file and its location.	**./restoreUserRoles.log**	Yes

Sample:

```
python ./restoreUserRoles.py -H localhost -p 9042 -i ./savedUserRoles.csv -o
./restoreUserRoles.log
```

The following is a sample input CSV file:

```
User name   User ID                                          Assigned
roles                                        * Note: The header row should not present in
the input CSV file content
-------------------------------------------------------------------------------------------------------------
"un00001",  "57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"User,Administrator,GroupMailboxAdministrator"
"un00003",  "57428@dcc7a7ac-626a-40c7-b805-e14b71d438d9",   "User,GroupMailboxAdministrator"
"un00002",  "57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9",   "User,GroupMailboxAdministrator"
```

restoreUserVmProfiles.py

Functions: Restores the User Voicemail Profiles data that are not stored in Configuration Server (data saved by corresponding backup script).

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (-i)	The name of the input backup CSV file and its location.	**./savedUserVmProfiles.csv**	Yes
Output (-o)	The name of the output file and its location.	**./restoreUserVmProfiles.log**	Yes

Sample:

```
python ./restoreUserVmProfiles.py -H localhost -p 9042 -i ./savedUserVmProfiles.csv -o
./restoreUserVmProfiles.log
```

The following is a sample input CSV file:

```
User name   User ID                                          Voicemail Profile
ID                          * Note: The header row should not present in the input CSV file
content
-------------------------------------------------------------------------------------------------------
"un00002",  "57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"11451ec2-d68a-4425-98ebfbf22a24fc7a"
"un00001",  "57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"92a5fd17-2b4b-493d-8727-004625e0a112"
"un00001",  "57426@f521b229-f599-47d4-81fd-2fbf15b02809",
"11451ec2-d68a-4425-98ebfbf22a24fc7a"
"un00003",  "57428@f521b229-f599-47d4-81fd-2fbf15b02809",
"92a5fd17-2b4b-493d-8727-004625e0a112"
```

saveDeviceCallingProfiles.py

Functions: The Device Calling Profile Assignments script creates a CSV file that contains records for all devices with a Calling Profile assigned to them. This CSV file later serves as the input for the Restoring Device Calling Profile Assignments procedure.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	The name of the output file and its location.	**./saveDeviceCallingProfiles.csv**	Yes

Sample:

```
python ./saveDeviceCallingProfiles.py -H localhost -p 9042 -o ./saveDeviceCallingProfiles.csv
```

The following is a sample output CSV file:

```
Device ID               Calling Profile ID
"20001@SwitchSA02",     "24e06da6-1dd3-479a-a0a2-0db2c9aa767c"
"10001@SwitchSA01",     "a4ea866b-7dcc-4da6-97ae-24cb14f2e150"
Every record of the device calling profile in the CSV file contains a device ID and the
corresponding ID of a Calling Profile assigned to the device.
Device ID consists of corresponding DN number and the switch name the device belongs to,
separated by ‘@’; for instance: 10001@SwitchSA01.
```

saveUserCallingProfiles.py

Functions:

The User Calling Profile Assignments script creates a CSV file that contains records of all Users with a Calling Profile assigned to them. The CSV file later serves as an input for the Restoring User Calling Profile Assignments procedure.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	The name of the output backup CSV file and its location.	**./saveUserCallingProfiles.csv**	Yes

Sample:

```
python ./saveUserCallingProfiles.py -H localhost -p 9042 -o ./saveUserCallingProfiles.csv
```

The following is a sample output CSV file:

```
User name   User ID                                          Calling Profile ID
"un00002",  "57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"0758d5a6-355a-4e13-9ef2-63884f88a99c"
"un00001",  "57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"e969a3be-6337-4041-8a9ca270843c6529"
```

Every record of the user calling profile in the CSV file contains a user name, user ID, and the corresponding ID of a Calling Profile assigned to the user. User ID consists of the corresponding person DBID and Configuration Server GUID separated by '@'; for instance: 57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9

saveUsergroupVmProfiles.py

Functions: The User Group Voicemail Profile Assignments script creates a CSV file that contains records of all User Groups with a Voicemail Profile other than the one assigned to them by System Profile. The CSV file later serves as an input for the Restoring User Group Voicemail Profile Assignments procedure.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	The name of the output backup CSV file and its location.	**./saveUsergroupVmProfiles.csv**	Yes

Sample:

```
python ./saveUsergroupVmProfiles.py -H localhost -p 9042 -o ./saveUsergroupVmProfiles.csv
```

The following is a sample output CSV file:

```
Group name  Group ID Voicemail                              Profile ID
"ag002",    "19613@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"92a5fd17-2b4b-493d-8727-004625e0a112"
"ag002",    "19613@f521b229-f599-47d4-81fd-2fbf15b02809",
"11451ec2-d68a-4425-98ebfbf22a24fc7a"
"ag001",    "19612@f521b229-f599-47d4-81fd-2fbf15b02809",
"92a5fd17-2b4b-493d-8727-004625e0a112"
```

Every record of the user group voicemail profile in the CSV file contains a group name, group ID and corresponding ID of a Voicemail Profile assigned to the user group. User group ID consists of the corresponding Agent Group DBID and Configuration Server GUID separated by '@'; for instance: 19613@dcc7a7ac-626a-40c7-b805-e14b71d438d9

saveUserRoles.py

Functions: The User Feature Server Roles script creates a CSV file that contains records of all users with Roles different from the default User role. The CSV file later serves as an input for the Restoring User Roles procedure.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	The name of the output backup CSV file and its location.	**./saveUserRoles.csv**	Yes

Sample:

```
python ./saveUserRoles.py -H localhost -p 9042 -o ./saveUserRoles.csv</tt>
```

The following is a sample output CSV file:

```
User name   User ID                                        Assigned roles
"un00001",  "57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"User,Administrator,GroupMailboxAdministrator"
"un00003",  "57428@dcc7a7ac-626a-40c7-b805-e14b71d438d9",   "User,GroupMailboxAdministrator"
"un00002",  "57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9",   "User,GroupMailboxAdministrator"
```

Every record of the user roles in the CSV file contains user name, user ID, and the corresponding set of roles assigned to the user. User ID consists of the corresponding person DBID and Configuration Server GUID separated by '@'; for instance: 57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9

saveUserVmProfiles.py

Functions: The User Voicemail Profile Assignments script creates a CSV file containing records of all users with a Voicemail Profile other than the one assigned to them by System Profile. The CSV file later serves as an input for the Restoring User Voicemail Profile Assignments procedure.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Output (-o)	The name of the output backup CSV file and its location.	**./saveUserVmProfiles.csv**	Yes

Sample:

```
python ./saveUserVmProfiles.py -H localhost -p 9042 -o ./saveUserVmProfiles.csv
```

The following is a sample output CSV file:

```
User name User ID Voicemail Profile ID
"un00002", "57427@dcc7a7ac-626a-40c7-b805-e14b71d438d9", "11451ec2-d68a-4425-98ebfbf22a24fc7a"
"un00001", "57426@dcc7a7ac-626a-40c7-b805-e14b71d438d9",
"92a5fd17-2b4b-493d-8727-004625e0a112"
"un00001", "57426@f521b229-f599-47d4-81fd-2fbf15b02809", "11451ec2-d68a-4425-98ebfbf22a24fc7a"
"un00003", "57428@f521b229-f599-47d4-81fd-2fbf15b02809",
"92a5fd17-2b4b-493d-8727-004625e0a112"
```

Every record of the User Voicemail Profile in the CSV file contains a user name, user ID, and the corresponding ID of a Voicemail Profile assigned to the user. User ID consists of the corresponding person DBID and Configuration Server GUID separated by ‘@’; for instance: 57426@cb2fdedda57f-49e6-a54d-3f930eb1dfc5

set_mailbox_user_tz.py

Functions: Updates the time zones of mailboxes and the time zones of users who are associated with the mailbox.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes

Parameters	Description	Sample	Mandatory
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Run Mode (--exec)	Activates the execution mode.	--exec	No
Input (-i)	Input TimeZones CSV file name.	**./mailbox_user_tz.csv**	Yes
Output (-o)	The name of the output file and its location.	**./set_mailbox_user_tz.log**	Yes

Sample:

```
python ./set_mailbox_user_tz.py -H localhost -p 9042 --exec -i ./mailbox_user_tz.csv -o
./set_mailbox_user_tz.log</tt>
```

The following is a sample input CSV file:

```
Mailbox Number      Time Zone ID            * Note: The header row should not present in the
input CSV file content
---------------------------------------
7100,               America/New_York
7200,               America/Los_Angeles
```

setAdminRoles.py

Functions: Sets administrative roles for users from the list.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Input (-i)	Input Users CSV file name.	**./setAdminRoles.csv**	Yes
Output (-o)	The name of the output file and its location.	**./setAdminRoles.log**	Yes

Sample:

```
python ./setAdminRoles.py -H localhost -p 9042 -i ./setAdminRoles.csv -o ./setAdminRoles.log
```

The following is a sample input CSV file:

```
User Key        * Note: The header row should not present in the input CSV file content
--------------
TServ.User1
TServ.User2
```

TServ.User3

updateMailboxMessagesCFFormat.py

Functions: Updates the Mailbox Message CF record format.

Parameters:

Parameters	Description	Sample	Mandatory
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No
Format (--cassandracounter)	Updating of the CF record to old format: --cassandracounter true new format: --cassandracounter false	false	Yes
Output (-o)	The name of the output file and its location.	**./updateCFFormat.log**	Yes

Sample:

```
python./updateMailboxMessagesCFFormat.py -H localhost -p 9042  --cassandracounter false -o
./updateMailboxMessagesCFFormat.log
```

The following is a sample format:

```
New Record Format:
<tt>KoMailbox,  a38c7205-fa51-495c-9dd0-d70b2c2c4155,   {"is_new": "y", "is_high_priority":
"n"}</tt>

Old Record Format:
<tt>KoMailbox,  a38c7205-fa51-495c-9dd0-d70b2c2c4155,   mailbox-5108-message-1</tt>
```

voicemail-core

corruptMessagesInMailboxmessages.py

This scripts replaces the following scripts:

- checkMailboxMessagesIntegrity.py
- cleanupMailboxMessagesCF.py
- getAllProblematicMailboxes.py

Functions: To discover and fix corrupted messages in mailboxMessages CF. MessageId present in

mailboxMessages CF but not present in mailmessage and message_bytes CF, this occurs when deleted messages reappear.

Parameters:

Parameters	Description	Sample	Mandatory	Default Value
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes	NA
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes	NA
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No	sipfs
Output (-o)	The output JSON file with corrupted messages list.	corruptMessagesInMailboxmessages	Yes	NA
Fix (-f)	If `-f` is not given, the script only finds the corrupted messages If `-f` is enabled, the script finds the corrupted messages and fix them.	NA	No	Fix disabled.
Consistency (-c)	Read and Write Consistency Levels.	LOCAL_QUORUM	No	ONE

Sample:

```
python corruptMessagesInMailboxmessages.py -H localhost -p 9042 -o
corruptMessagesInMailboxmessages
```

The following is a sample output JSON file:

```
{
  "1001": [
    "c007c70b-7431-45ec-974a-617722bf7b01",
    "928e95aa-9d4a-4b8c-96af-5097c61460ae"
  ]
}
```

corruptMessagesInMailboxmessagesAndMailMessage.py

This script replaces the following scripts:

- problematicMailMessages.py
- cleanupMailmessageCF.py

Functions:

To discover and fix corrupted messages in mailboxMessages CF and mailmessage CF. MessageId present in mailboxMessages and mailmessage CF but not present in message_bytes CF and retention limit not set for message bytes, this occurs when deleted messages reappear.

Parameters:

Parameters	Description	Sample	Mandatory	Default Value
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes	NA
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes	NA
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No	sipfs
Output (-o)	The output JSON file with corrupted messages list.	corruptMessagesInMailboxmessages	Yes	NA
Fix (-f)	If -f is not given, the script only finds the corrupted messages If -f is enabled, the script finds the corrupted messages and fix them.	NA	No	Fix disabled
Consistency (--consistency)	Read and Write Consistency levels.	LOCAL_QUORUM	No	ONE

Sample:

```
python corruptMessagesInMailboxmessagesAndMailMessage.py -H localhost -p 9042 -o
corruptMessagesInMailboxmessagesAndMailMessage
```

The following is a sample output JSON file:

```
{
  "1001": [
    "c007c70b-7431-45ec-974a-617722bf7b01",
    "928e95aa-9d4a-4b8c-96af-5097c61460ae"
  ]
}
```

countMailboxes.py

Functions: Gets the total number of mailboxes that a tenant uses, based on mailboxmessages CF.

Parameters:

Parameters	Description	Sample	Mandatory	Default Value
Host Name (-H)	The host name of	CassNode01	Yes	NA

Parameters	Description	Sample	Mandatory	Default Value
	target Cassandra database.			
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes	NA
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No	sipfs
Output (-o)	The output log contains the total mailbox count.	countMailboxes	Yes	NA

Sample:

```
python countMailboxes.py -H localhost -p 9042 -o countMailboxes
```

removeMailboxAndMailboxMessages.py

Functions: Deletes all mailboxes and its messages based on the given input. Remove the mailbox option (TServer→gvm_mailbox) from Agent or Agent Group before running this script.

Parameters:

Parameters	Description	Sample	Mandatory	Default Value
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes	NA
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes	NA
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No	sipfs
Input (-i)	Input Users CSV file with mailbox id to be removed.	mailbox_list.csv	Yes	NA
Output (-o)	The output log file contains details of deleted mailboxes and its messages.	removeMailboxAndMailboxMessages	Yes	NA
Consistency (--consistency)	Read and Write Consistency levels.	LOCAL_QUORUM	No	ONE

Sample:

```
python removeMailboxAndMailboxMessages.py -H localhost -p 9042 -o
removeMailboxAndMailboxMessages -i mailbox_list.csv
```

The following is a sample input CSV file:

```
799033
1001
```

removeMailboxMessages.py

Functions: Deletes all messages from the provided input mailbox list.

Parameters:

Parameters	Description	Sample	Mandatory	Default Value
Host Name (-H)	The host name of target Cassandra database.	CassNode01	Yes	NA
Port (-p)	The CQL port number of target Cassandra database.	9042	Yes	NA
Keyspace (--keyspace)	The name of target keyspace.	sipfs	No	sipfs
Input (-i)	Input Users CSV file with their mailbox ID.	mailbox_list.csv	Yes	NA
Output (-o)	The output log file contains details of deleted messages.	removeMailboxMessages	Yes	NA
Consistency (--consistency)	Read and Write Consistency levels.	LOCAL_QUORUM	No	ONE

Sample:

```
python removeMailboxMessages.py -H localhost -p 9042 -o removeMailboxMessages -i
mailbox_list.csv
```

The following is a sample input CSV file:

```
799033
1001
```

Configuration Database Synchronization

When there are updates in the Configuration Database, then Feature Server synchronizes the following objects.

- SIP Switch objects:
  - CFGDN
  - CFGAgentLogin
- Tenant objects:
  - CFGAgentGroup
  - CFGPerson
  - CFGPlace

SIP Feature Server synchronization with Configuration Database consists of the following three functional facilities:

- Initial Import
- Real-time Synchronization
- History Log Synchronization

Initial Import

Initial import synchronization occurs during the first time the Master Feature Server is started. During this synchronization, switch objects and tenant objects are imported from the Configuration Database. To avoid repeating the import, Feature Server saves/retains initial import flags in Cassandra Database. After the check, Feature Server imports when required.

> **Important**
>
> - All Feature Server nodes perform the initial import of switch objects regardless of master/confsync Feature Servers.
> - Master node performs the initial import of tenant objects.

Real-time Synchronization

During real-time synchronization, Feature Server dynamically synchronizes Cassandra database with

updates from the Configuration Database.

> **Important**
>
> - All Feature Server nodes perform the real-time sync of switch objects.
> - Only Master and Confsync Feature Server nodes perform real-time sync of Tenant related objects.

History Log Synchronization

History log retrieval synchronization occurs when Feature Server is disconnected and then reconnected to the Configuration Database. During this process, Cassandra is updated with all the changes in the Configuration Database during the time when Feature Server was disconnected. If the history log is lost owing to a Feature Server outage, reimport mechanism is triggered automatically. For more details on reimporting, see Reimporting Configuration Data.

> **Important**
>
> - Non-master/non-confsync Feature Server nodes perform Historical Synchronization of switch objects.
> - Master and confsync Feature Server nodes perform Historical Synchronization of tenant objects.

Retrieve provisioned and unprovisioned devices

Feature Server enables you to retrieve a list of all devices that are created in Feature Server by running the **getColumnFamilyContent.py** script. This script is located in the *<FS installation path>***python\util** folder.

See Python Scripts for information on how to deploy and run the **getColumnFamilyContent.py** script.

Sample command line

A sample command line to run the script:

```
java -jar <jython-version>.jar getColumnFamilyContent.py -H localhost -p 9160 -c dm_device -k
vendor,model,dn_list -o devices.log -x devices.csv
```

Note that the script accepts `-k` and `-x` only from Feature Server release version 8.1.201.94. The contents of the .csv file must be in the following format:

ID	Vendor	Model	dn_list
abcdef0000001	Genesys	420HD	1001,1002
abcdef0000002	Genesys	420HD	
abcdef0000003	Genesys	420HD	1001,1002

Key

- ID represents the mac address of the phones.
- Devices containing dn_list value are provisioned phones.
- Devices that do not contain dn_list value are unprovisioned phones.

Set up mailbox and user time zones

Feature Server now enables you to use the **set_mailbox_user_tz.py** script to update, in bulk, the time zones of mailboxes and the time zones of the users who are associated with that mailbox. This script is located in the *<FS installation path>***\python\util** folder. Use the `-e` option to enable the script to make changes.

> **Important**
>
> If the time zone of a group mailbox is updated by using this script, then the timezone of the users associated with the group mailbox will not be updated.

See Python Scripts for information on how to deploy and run the **set_mailbox_user_tz.py** script.

Sample command line

Use the following command line format to run the **set_mailbox_user_tz.py** script:

```
java -jar <jython-version>.jar set_mailbox_user_tz.py -H localhost -p 9160 -i
mailbox_user_tz.csv -o set_mailbox_user_tz.log
```

How to prepare the input file for the script?

Create an input file named **mailbox_user_tz.csv**.
The file must contain pairs delimited by a comma *<mailbox number>,<Time Zone Id>*.
For example, the following is a sample of the input-file content:
7100,America/New_York
7200,America/Los_Angeles

Scheduled maintenance tasks

> **Important**
> SIP Feature Server 8.1.203 requires external **Python 3.x interpreter** to execute scheduled maintenance tasks in environments where CQL connection mode is enabled for Cassandra connections. You can configure the python location using the **process-launcher** option in the **python** section of Feature Server configuration. After you start using the CQL connection and external python interpreter, you can remove the legacy jython interpreter module from Feature Server installation. For more details on removing unused modules, see the SIP Feature Server Deployment Guide.

Master Feature Server can schedule and run three maintenance tasks:

- update-mailbox-counters
- delete-expired-messages
- forget-me
- reimport

All the scheduled tasks must be configured in the configuration environment. All options intended for maintenance tasks are detailed in the **[ScheduledTasks]** section of the Configuration Options page.

Running maintenance tasks

The following are the two methods to run scheduled tasks:

Method 1

To run the schedule task manually by calling the Feature Server resource, complete the following steps:

1. Type the following address in your browser:
   `http://<fs-server-host:port>/fs/api/admin/tasks/start/<task-name>`

   where,
   <*fs-server-host:port*> is your Feature Server host.

   <*task-name*> is the name of your task.
2. When the site requests your credentials, log in by using administrator credentials to initiate this scheduled task.

Method 2

To enable Master Feature Server to automatically start scheduled tasks, set the **[Scheduled-task] <*task-name*>.active** option to `true` in the master Feature Server application, then scheduled task runs automatically at the time scheduled in the **[Scheduled Tasks]<*task-name*>.schedule** option in the master Feature Server application.
For example, to enable update-mailbox-counters tasks: **[ScheduleTask] update-mailbox-counters.active** = `true`

Cancelling maintenance tasks

You can cancel a scheduled task by calling HTTP resource on master Feature Server:

1. Type the following address in your browser:
   `http://<fs-server-host>:port/fs/api/admin/tasks/cancel/<task-name>`

   where,
   <*fs-server-host:port*> is your Feature Server host.

   <*task-name*> is the name of your task.
2. When the site requests your credentials, log in by using administrator credentials to cancel schedule tasks.

> **Warning**
> Genesys does not recommend cancelling tasks.

Monitoring maintenance tasks

You can monitor scheduled tasks using the Feature Server HTTP resource that displays the status of scheduled tasks:

1. Type the following address in your browser on the master Feature Server:
   `http://<fs-server-host>:<port>/fs/api/admin/tasks/state`

   where,
   <*fs-server-host:port*> is your Feature Server host.

   <*task-name*> is the name of your task.
2. When the site requests your credentials, log in by using administrator credentials to display the current state of scheduled tasks, which can be one of the following:

   - In progress
   - Ready to start

GDPR Compliance

Feature Server release 8.1.202.10 includes the following python script:

- **forgetMe.py**—Deletes the voicemail data of a customer when requested.

You can run this python script on the master Feature Server instance to delete the voicemail data. The customer-related information received from the common Web UI will be transformed into a JSON input file. The forgetMe.py script will fetch the JSON files (that were added since the last execution time to fetch the ANI) added to the **gdpr-directory** option configured in the **[gdpr]** section of the master Feature Server. The script will then set the expiration time to 21 days for voicemails that correspond to the ANI obtained. The voicemails will be deleted after the expiration time.

From release 8.1.202.16, Forget Me will fetch files based on the last script execution time.

Feature Server release 8.1.2 includes the following python script:

- **exportMe.py**—Exports voicemail data in .wav format when requested by the customer, from the Cassandra database in a client-understandable format.

Data, such as name, phone number, email address, bank details, and IP address are considered as Personally Identifiable Information (PII). Anything that is likely to identify an individual, or a combination of other held data to identify an individual is considered as PII. According to EU GDPR, when a client requests to access his/her personal data that is available with the contact center, the PII associated with the client should be exported from the database in client-understandable format.

Forget Me

The **forgetMe.py** script will parse the input JSON file and obtain the ANI. It will check whether the ANI has any voicemail information associated in the Feature Server Cassandra database. If any data is present, then the respective records will be queried in the column families: *mailmessage*, *message_bytes*, and *mailboxmessages* using its key and the expiration time will be set to 21 days. If no messages are present for an ANI, the operation will be skipped and the appropriate status messages will be written to the log file. The log file is in the format: **sipfs-forgetMe-<DDMMYYYY>.log**. The output file includes the input file information, and the execution results, success and failures with appropriate error information.

Notes

- Feature Server does not delete the data upon manual script execution. Instead, it will set the expiration time which will make the data inaccessible.
- The Feature Server script will query the Cassandra database based on the ANI which is received as one of the inputs.
- The expiration time (TTL) of the Voicemail messages and the metadata of the corresponding ANI will be set to 21 days.

- Voicemail messages corresponding to an ANI residing in different mailboxes will also be set with the new expiration time.
- The expiration time will be set for the corresponding records of the column families: *mailmessage*, *mailboxmessages*, and *message_bytes* from **sipfs** keyspace.
- The script will be invoked once a day scheduled by automatic triggering within Feature Server.

Sample input JSON file

The following is a sample input JSON file:

```
{
   "caseid":"123456789",
   "consumers":[
      {"consumer":
         [
            {"name":"John Doe"},
            {"name": "John Q. Doe"},
            {"phone":"555551212"}
         ]
      },
      {"consumer":
         [
            {"name":"Dan Akroyd"},
            {"phone":"555556161"},
            {"email":"danny@hollywood.com"},
            {"fbid":"Dan Akroyd"}
         ]
      }],
   "gim-attached-data":{"kvlist":["AcctNum", "SSN"]}
 }
```

If the retention limit is already set for voicemail messages, then the retention limit of the voicemail will be set as the expiration time, provided the retention limit is less than 21 days.

If the customer makes a second request to delete the voicemails associated with their ANI, then voicemail set with expiration time during the first request will be skipped during execution. The voicemails deposited after the first request alone will be set with the new expiration time.

Sample Output File

```
======================================================================
| forgetMe.py script run |
======================================================================

<input.json>//actual content

execution start time = 2018-04-19 13:23:22
[debug]ANI '555556161' has voicemail data
[debug]Expiration time set for 555556161 in mailmessage
[debug]Expiration time set for 555556161 in mailboxmessages
[debug]Expiration time set for 555556161 in message_bytes
[debug]ANI '555556162' has no voicemail data
execution end time = 2018-04-19 13:23:22
```

Scheduling the forget-me Task

The **forgetMe.py** script execution should be scheduled once a day from SIP Feature Server. You can schedule the forget-me task as described in Scheduled maintenance tasks.

> **Important**
> The "update-mailbox-counters" task should run once a day to overcome the mailbox counter issue caused by running the "forget-me" task.

Configuration Options

The following options enable automatic script execution of Forget Me:

- **forget-me.active** = `true` (activate/deactivate)
- **forget-me.cmd** = `forgetMe.py --dbhost <host> --dbport <port>`
- **forget-me.schedule** = `0 0 4 ? * *` (Schedule)

Export Me

SIP Feature stores voicemail recordings of the client in the Cassandra database. The requirement is to develop a script that queries the Cassandra database based on the ANI fetched from the input file and export the voicemail recordings. This script should be triggered and executed automatically from SIP Feature Server once a day. The contact center administrator receives the client details from a common web user interface as specified in the GDPR compliance page and generates the input JSON file that is common for all components/solutions.

Input source - **export-<DDMMYYYY>.json**

Output naming - **sipfs-exportMe-<DDMMYYYY>-execution.log** (UTC is the time standard to be used for naming)

- SIP Feature Server stores the client's voicemail recordings in Cassandra DB.
- The customer care administrator will receive the customer details through a common web user interface and generate the input JSON file required for Export Me.
- SIP Feature Server will contain the exportMe script that fetches the ANI from the input file that is added to the gdpr-directory.
- The script will parse all the input files that were added since the last execution time to fetch the ANI.
- The script will query the Cassandra DB for voicemail recordings associated with each ANI and export them as .wav files.
- Voicemail recordings for an ANI residing in different mailboxes will also be exported.
- The script will be triggered and executed automatically once a day using scheduled tasks.

Input and Output Formats

The following arguments are mandatory. If any of the following arguments are missing, the script will fail after writing the appropriate error messages to the output file. Only the Cassandra host and port will be configured in the Scheduled tasks command. Other parameters will be added to command line arguments from the SIP Feature Server application when the task is triggered.

Script run-command - **python exportMe.py --dbhost <host> --dbport <port> --fileLocation <gdpr-directory>**

Where:

- **<host>**: Cassandra DB host name of SIP Feature Server
- **<port>**: Cassandra DB port of SIP Feature Server
- **<gdpr-directory>**: Absolute path to the directory where input files are stored

Sample run-command - **python exportMe.py --dbhost 10.31.12.99 --dbport 9042 --fileLocation "C:\Users\joanselm\Documents\privacy gdpr**

Sample Input File

export-<DDMMYYYY>-<any optional content>.json

The phone number here refers to the ANI that SIP Feature Server will use to query the SIP Feature Server DB.

The following is a sample input JSON file:

```
{
   "caseid":"123456789",
   "consumers":[
      {"consumer":
         [
            {"name":"John Doe"},
            {"name": "John Q. Doe"},
            {"phone":"55551011"}
         ]
      },
      {"consumer":
         [
            {"name":"Dan Akroyd"},
            {"phone":"55551012"},
            {"phone":"555556162"},
            {"email":"danny@hollywood.com"},
            {"email":"funnyguy@comedy.org"},
            {"fbid":"Dan Akroyd"}
         ]
      }],
   "gim-attached-data":{"kvlist":["AcctNum", "SSN"]}
 }
```

There are two primary output files:

- **<component>-<script name>-<DDMMYYYY>-access.json**
  This will include input file name at the top, JSON format, and references to recordings available for each ANI. The recordings will be exported as .wav files. For each input file fetched, access.json will

be generated individually.

- **<component>-<script name>-<DDMMYYYY>-execution.log**
  This will include input file at the top. The log will contain the execution results, and success and failure messages with appropriate error information.

The naming convention for the exported recording file will contain the following format:

ANI-<Date from timestamp>.wav

Sample Output Files

access.json

```
{
   "inputfile":"export.json",
   "caseid":"123456789",
   "consumers":[
      {"55551011":
         [
            {"file":"55551011-11072018.wav"},
            {"file":"55551011-23072018.wav"},
          ]
      },
      {"55551012":
         [
            {"file":"55551012-11072018.wav"},
            {"file":"55551012-23072018.wav"},
            {"msg":"Message marked for deletion"},
         ]
      }],
 }
```

execution.log

```
//input file

=====================================================================

| exportMe.py script run |
=====================================================================
[INFO] Execution start time = 2018-10-01 18:22:14
[INFO] DBHOST:172.24.131.63 DBPORT:9042
[INFO] KEYSPACE:sipfs
[INFO] ANI list obtained from inputfile:['55551011', '55551012']
[DEBUG] Message with key:65b2073e-4191-49e6-a013-a324255010d2 and callerid:55551011 is
present in message_bytes of keyspace sipfs
[DEBUG] ANI '55551011' has voicemail data
[DEBUG] Voicemail exported as 55551011-03072018-072321.wav
[DEBUG] ANI 55551012 has no new voicemail data
[INFO] Execution end time = 2018-10-01 18:22:14
```

Scheduling the export-me Task

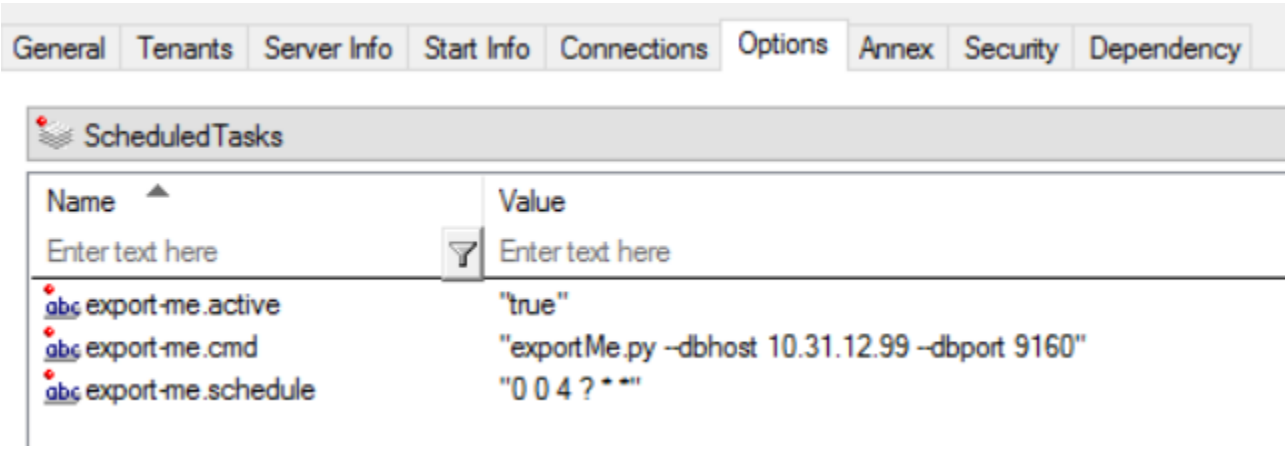
The feature also requires automatic script execution from SIP Feature Server once a day. This will be achieved using scheduled maintenance tasks. Scheduled maintenance tasks can be executed only from the master Feature Server. Export Me will be added as a task under **ScheduledTasks** that is available in SIP Feature Server application options.

task-name - export-me

Configuration Options

The following options enable automatic script execution of Export Me:

- **export-me.active** = `true` (Activate/Deactivate)
- **export-me.cmd** = `exportMe.py --dbhost <host> --dbport <port>` (Command line: The other parameters are added from SIP Feature Server)
- **export-me.schedule** = `0 0 4 ? * *` (Schedule)



The tasks can be started/stopped using the web APIs, which are already available for scheduled tasks.

Appendix

This section of the guide describes additional information pertaining to Cassandra maintenance.

Appendix: Backing up and restoring embedded Cassandra data

You can back up your Cassandra storage folders and use the saved storage folders to restore your Cassandra data if needed.

> **Important**
> Feature Server is capable of restoring its data by replicating it from other nodes in the Cassandra cluster. The restoration procedure described below is an exceptional point-in-time recovery measure. As with any backup procedure, **do not** make restoration a part of your regular maintenance process.

Backing up Cassandra data

Backing up your Cassandra storage folder requires you to:

- flush the Feature Server Cassandra keyspace on all Feature Server nodes, then
- save the Cassandra **storage** folder of each node

> **Tip**
> On the Linux platform, you can flush all the nodes in a datacenter by running the flush command using a parallel ssh utility such as pssh.

For each Feature Server instance:

1. Change to the Feature Server deployment location.
2. Verify the vms-port parameter in **launcher.xml**. The default value is 8080.
3. Change to the **lib** directory:
   - For Linux:
     ```
     cd work/jetty-0.0.0.0-8080-fs.war-_fs-any-/webapp/WEB-INF/lib
     ```
   - For Windows:
     ```
     cd work\jetty-0.0.0.0-8080-fs.war-_fs-any-\webapp\WEB-INF\lib
     ```
4. Verify the JMX port parameter in launcher.xml. The default value is 9192.
5. While Feature Server is running, run the nodetool flush command:

- For Linux:
```
java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-
cli-1.1.jar:cassandra-all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h
localhost -p 9192 flush sipfs

java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-
cli-1.1.jar:cassandra-all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h
localhost -p 9192 flush system
```
- For Windows:
```
java -cp libthrift-0.7.0.jar;cassandra-all-1.1.12.jar;cassandra-
thrift-1.1.12.jar;commons-cli-1.1.jar org.apache.cassandra.tools.NodeCmd -h
localhost -p 9192 flush sipfs

java -cp libthrift-0.7.0.jar;cassandra-all-1.1.12.jar;cassandra-
thrift-1.1.12.jar;commons-cli-1.1.jar org.apache.cassandra.tools.NodeCmd -h
localhost -p 9192 flush system
```

6. Copy the Cassandra **storage** folder to a safe location.

Restoring Cassandra data

> **Important**
> Use this recovery procedure only when absolutely necessary. Do not make it a part of your regular maintenance process.

To restore your Cassandra data from your backed-up storage folders:

1. Stop all Feature Server instances.
2. Copy the **storage** folder backups to their original location on each Feature Server.
3. Restart each Feature Server instance, beginning with the master instance.

Appendix: Performing maintenance operations on embedded Cassandra

The nodetool is a Cassandra utility for managing a Cassandra cluster. Use it for Feature Server Cassandra Maintenance.

Regular maintenance repairs inconsistencies across all data ranges, and ensures that replicated data is consistent across all nodes. You should perform maintenance on a node after a Feature Server upgrade and after new Feature Server nodes were added to the cluster. Apply additional maintenance steps after you remove a node from the cluster. See Performing maintenance after removing a node from the cluster for more details.

> **Important**
>
> The following Feature Server Cassandra Maintenance procedures are applicable only in the embedded Cassandra deployment. For information about the External Cassandra maintenance tasks, refer to the Cassandra documentation.

Performing regular maintenance

Run the nodetool command `repair` on all Feature Server hosts in a Cassandra cluster. See Running Nodetool on a Feature Server Host and Running nodetool repair on all Feature Server hosts in a Cassandra cluster for details about running nodetool and nodetool repair, respectively.

> **Important**
>
> All Feature Server hosts must be up and running to run repair. Running repair increases memory use and may not succeed if the Java heap space max limit is reached. For information on the Java heap size, see Hardware and software prerequisites.

As a best practice, schedule the regular maintenance weekly during the maintenance window (low usage hours), for example, by using Cron on Linux or as a Task Scheduler on Windows.

If you are planning for an ad hoc repair, then start the repair from the Feature Server instances that are configured as alternatevoicexml.

Running nodetool on a Feature Server host

The Nodetool utility has a command line interface. The steps below describe how to run nodetool commands on Cassandra when it is deployed on a host, as part of a Feature Server deployment.

1. Change the current directory.

In the Linux shell terminal:

```
cd Feature Server Installation Directory/work/
jetty-vms_host-http_port-fs.war-_fs-any-/webapp/WEB-INF/lib
```

In the Windows command prompt window:

```
cd Feature Server Installation Directory\work\
jetty-vms_host-http_port-fs.war-_fs-any-\webapp\WEB-INF\lib
```

where...

Feature Server Installation Directory is where Feature Server is installed in that host.

vms_host is the IP address of the vms host parameter that Feature Server started with, defined in launcher.xml or in the command line. The default is 0.0.0.0.

http_port is the port number that Feature Server started with, defined in launcher.xml. The default is 8080.

2. Run the Nodetool utility.

On Linux

Note that .jar is separated by the colon ":" punctuation mark.

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-Cassandra version.jar:commons-
cli-1.1.jar:cassandra-all-Cassandra version.jar org.apache.cassandra.tools.NodeCmd -h
Cassandra host -p jmx_port nodetool command
```

On Windows

Note that .jar is separated by the semicolon ";" punctuation mark.

```
java -cp libthrift-0.7.0.jar;cassandra-thrift-Cassandra version.jar;commons-
cli-1.1.jar;cassandra-all-Cassandra version.jar org.apache.cassandra.tools.NodeCmd -h
Cassandra host -p jmx_port nodetool command
```

where...

Cassandra version could be either 1.1.6 or 1.1.12 depending on Feature Server version installed. Please check the actual cassandra-all-*.jar file name in current directory.

Cassandra host is the hostname or IP address where Feature Server is running.

jmx_port is the JMX port number that Feature Server started with, defined in launcher.xml. Default=9192.

nodetool command can be `ring [keyspace name]` or `removetoken <token>` or `repair` or `flush`.

For example, the following command line runs nodetool and sends the command `ring` to the Feature Server running on the local host:

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-cli-1.1.jar:cassandra-
all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 ring sipfs
```

Running Nodetool with Cassandra JMX Authentication

If Cassandra JMX authentication is enabled, then the nodetool command requires JMX username and password as command line arguments:

On Linux

Note that .jar is separated by the colon ":" punctuation mark.

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-Cassandra version.jar:commons-
cli-1.1.jar:cassandra-all-Cassandra version.jar org.apache.cassandra.tools.NodeCmd -h
Cassandra host -p jmx_port -u username -pw password nodetool command
```

On Windows

Note that .jar is separated by the semicolon ";" punctuation mark.

```
java -cp libthrift-0.7.0.jar;cassandra-thrift-Cassandra version.jar;commons-
cli-1.1.jar;cassandra-all-Cassandra version.jar org.apache.cassandra.tools.NodeCmd -h
Cassandra host -p jmx_port -u username -pw password nodetool command
```

Running Nodetool with Secured Cassandra JMX

If Cassandra JMX port is secured using SSL/TLS, then use the following command to run the nodetool commands such as ring or repair:

> **Important**
> This command is supported in Feature Server release 8.1.202.04 and later.

On Linux

Note that .jar is separated by the colon ":" punctuation mark.

```
java -Dssl.enable=true -Dcom.sun.management.jmxremote.ssl.need.client.auth=true
-Dcom.sun.management.jmxremote.ssl=true -Dcom.sun.management.jmxremote.registry.ssl=true
-Djavax.net.ssl.keyStore=<keyStore_file>
-Djavax.net.ssl.keyStorePassword=<keyStore_password_file>
-Djavax.net.ssl.trustStore=<cert_store_file>
-Djavax.net.ssl.trustStorePassword=<cert_store_password_file> -cp
libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-cli-1.1.jar:cassandra-
all-1.1.12.jar:fs-nodetool-utility-fs-9-SNAPSHOT.jar
com.genesyslab.nodetool.utility.NodeCmdCustom -h Cassandra host -p jmx_port nodetool command
```

On Windows

Note that .jar is separated by the semicolon ";" punctuation mark.

```
java -Dssl.enable=true -Dcom.sun.management.jmxremote.ssl.need.client.auth=true
-Dcom.sun.management.jmxremote.ssl=true -Dcom.sun.management.jmxremote.registry.ssl=true
-Djavax.net.ssl.keyStore=<keyStore_file>
-Djavax.net.ssl.keyStorePassword=<keyStore_password_file>
-Djavax.net.ssl.trustStore=<cert_store_file>
-Djavax.net.ssl.trustStorePassword=<cert_store_password_file> -cp
libthrift-0.7.0.jar;cassandra-thrift-1.1.12.jar;commons-cli-1.1.jar;cassandra-
all-1.1.12.jar;fs-nodetool-utility-fs-9-SNAPSHOT.jar
com.genesyslab.nodetool.utility.NodeCmdCustom -h Cassandra host -p jmx_port nodetool command
```

Running nodetool repair on all Feature Server hosts in a Cassandra cluster

On Linux

```
cd genesys/fs/work/jetty-0.0.0.0-8080-fs.war-_fs-any-/webapp/WEB-INF/lib

java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-cli-1.1.jar:cassandra-
all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 repair -pr
```

On Windows

```
cd genesys\fs\work\jetty-0.0.0.0-8080-fs.war-_fs-any-\webapp\WEB-INF\lib

java -cp libthrift-0.7.0.jar;cassandra-thrift-1.1.12.jar;commons-cli-1.1.jar;cassandra-
all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 repair -pr
```

Switching Snitches

To change an **endpoint_snitch** from **SimpleSnitch** to **PropertyFileSnitch** or **GossipingPropertyFileSintch**, follow these steps:

1. Stop all SIP Feature Server instances.
2. Change **replicationOptions** and **replicationStrategyClassName** in the Cassandra section of the SIP Feature Server application, as required.
3. Create **cassandra-topology.properties** or **cassandra-rackdc.properties** file based on **PropertyFIleSnitch** or **GossipingPropertyFileSnitch** properties, respectively.
4. Copy the above-mentioned property files into the **<FS installed location>/resources** directory.
5. Update the **endpoint_snitch** value in **cassandra.yaml** file in each Cassandra node to reflect the required snitch value (**PropertyFIleSnitch** or **GossipingPropertyFileSnitch**).
6. Restart the master instance first, and then restart all other SIP Feature Server instances.
7. Run the `nodetool` repair commands. For details, refer here.

Switching Seeds

To change the master instance, you must change the seed value inside the master and non-master nodes.

> **Important**
> The seed value is the FQDN/IP address of the master instance.

Steps to switch seeds:

1. Stop all SIP Feature Server instances.
2. In the current master SIP Feature Server application, under the **Cluster** section, change the master value to `false`.
3. In the new master SIP Feature Server application, under the **Cluster** section, change the master value to `true`.
4. Edit the **cassandra.yaml** file for each SIP Feature Server instance and change the seed value. The seed value is the FQDN/IP address of the master instance.
5. Perform these changes sequentially one instance at a time.
6. Restart the master instance first, and then restart all other SIP Feature Server instances.
7. Run the `nodetool` repair commands. For details, refer here.

Performing maintenance after removing a node from the cluster

When you remove a node from the ring, you must also remove the corresponding tokens. Use these software procedures:

1. Run nodetool `ring` to obtain tokens. (See Running Nodetool on a Feature Server Host)
2. Run nodetool `removetoken` to remove nodes from the ring.
   Note: Only nodes that are down can be removed.
3. Run nodetool `ring` to validate removal.
4. Run nodetool `repair` on all Feature Server hosts in the cluster.
5. Run nodetool `ring` to validate repair.

For example, run nodetool `ring` to obtain tokens of nodes to be removed:

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-cli-1.1.jar:cassandra-
all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 ring sipfs
```

...to see the following output returned tokens of nodes which are down:

```
<fs-host-IP1> DC1 RAC1 Up Normal 1.17 MB100.00% 167086018864645871692761019448293152722
<fs-host-IP2> DC1 RAC2 Up Normal 1.29 MB 100.00% 260037876766820018229186112944720563l6
<fs-host-IP3> DC2 RAC1 Down Normal 1.15 MB 100.00% 41007983964572150951275225962045789866
<fs-host-IP4> DC2 RAC2 Down Normal 1.16 MB 100.00% 53685600614278234503162023330018045221
```

The following nodetool commands remove <fs-host-IP3> and <fs-host-IP4> nodes from the ring:

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-cli-1.1.jar:cassandra-
all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 removetoken
41007983964572150951275225962045789866

java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.12.jar:commons-cli-1.1.jar:cassandra-
all-1.1.12.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 removetoken
53685600614278234503162023330018045221
```

After token removal, running nodetool ring should provide the following output:

```
<fs-host-IP1> DC1 RAC1 Up Normal 1.17 MB 100.00% 167086018864645871692761019448293152722
<fs-host-IP2> DC1 RAC2 Up Normal 1.29 MB 100.00% 260037876766820018229186112944720563l6
```

Appendix: Feature Server Maintenance Python Scripts for Embedded Cassandra

This section explains how to deploy and run the Python script.

How to deploy the script?

To deploy the script, follow these steps:

1. On the master Feature Server instance, copy the <*jython-version*>.jar file **from** <FS installation path>\work\jetty-x.x.x.x-pppp-fs.war-_fs-any-\webapp\WEB-INF\lib **to** <FS installation path>\python\.
   The value of <*jython-version*>.jar varies depending on the Feature Server version:
   - 8.1.202.09 and earlier - jython-2.7b1.jar
   - 8.1.202.10 and later - jython-standalone-2.7.1b2.jar
   - 8.1.202.17 and later - jython-standalone-2.7.1b3.jar
   - 8.1.202.45 and later - jython-standalone-2.7.2.jar
2. Open console and navigate to <FS installation path>\python\util, which contains the scripts.
3. Copy both the above-mentioned python scripts to the path <FS installation path>\python\.
4. Enter the command to set JYTHONPATH:
   Windows

   set JYTHONPATH=<FS installation path>\python

   Linux

   export JYTHONPATH=<FS installation path>/python

How to run the script?

Use the following command line format to run the script:

```
java -jar <jython-version>.jar <scriptname>.py <script input parameters>
```

The value of <*jython-version*>.jar varies depending on the Feature Server version.

Feature Server version	Jython file
8.1.202.09 and earlier	jython-2.7b1.jar
8.1.202.10 and later	jython-standalone-2.7.1b2.jar
8.1.202.17 and later	jython-standalone-2.7.1b3.jar

Feature Server version	Jython file
8.1.202.45 and later	jython-standalone-2.7.2.jar

> **Important**
>
> Remove the **Jython JAR** file from the **<FS installation path>\python** directory after the execution of the script.

Appendix: Remove a node from Feature Server deployed with Embedded Cassandra

Prepare your environment to run node tool on Feature Server host as described in the Running Nodetool on a Feature Server Host page.

First, listing current nodes:

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.6.jar:commons-cli-1.1.jar:cassandra-
all-1.1.6.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 ring
```

Address	DC	Rack	Status	State	Load	Owns	Token
Xx	xx	xx	xx	xx	xx		837941282122956078124
10.52.86.38	usw1	RAC1	Down	Normal	?	59.09%	141891808762306545351
10.51.29.29	usw1	RAC2	Up	Normal	547.84 KB	3.45%	200553042096189663544
10.51.28.170	usw1	RAC1	Up	Normal	498.49 KB	37.46%	837941282122956078124

When the node is up, you can run nodetool decommission in the node you want to remove. For example, node 10.51.29.29 is up so run the following command in this node to remove it.

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.6.jar:commons-cli-1.1.jar:cassandra-
all-1.1.6.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192 decommission
```

If the node is down, it results in ‘dead’ Cassandra node situation if you add a node to the cluster and then delete the Virtual Machine.
To recover from this problem, you must remove the ‘dead’ node. There is no connection to Cassandra on the node being deleted, so you cannot use ‘decommission’ command. Use ‘removetoken’ instead. For example, node 10.52.86.38 is Down. Run the following command by using its token in the any other node to remove 10.52.86.38.

```
java -cp libthrift-0.7.0.jar:cassandra-thrift-1.1.6.jar:commons-cli-1.1.jar:cassandra-
all-1.1.6.jar org.apache.cassandra.tools.NodeCmd -h localhost -p 9192
```

Note that this command takes a while to complete.