SIP Feature Server Administration Guide

Re-initialize Feature Server backend

5/6/2025

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,AdministrativeUser,GroupMailboxAdministrator
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	924b57d1-738d-493d-8727-004625e0a112
un00001	57426@f521b229-f599-47d4-81fd-2fbf15b02809	11451ec2-d68a-4425-98eb-fbf22a24fc7a
un00003	57428@f521b229-f599-47d4-81fd-2fbf15b02809	924b57d1-738d-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	924b57d1-738d-493d-8727-004625e0a112
ag002	19613@f521b229-f599-47d4-81fd-2fbf15b02809	11451ec2-d68a-4425-98eb-fbf22a24fc7a
ag001	19612@f521b229-f599-47d4-81fd-2fbf15b02809	924b57d1-738d-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	07487d5a-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.