Genesys Voice Platform Private Edition Guide

Provision Genesys Voice Platform

6/10/2026

Contents

- 1 Tenant provisioning
  - 1.1 Overview
  - 1.2 Use Case 1 - Deploy New Tenant
  - 1.3 Use Case 2 - Update Existing Tenant
- 2 Provisioning Nexus connection in GVP MCP
  - 2.1 Integrating MCP to Nexus for Dialogflow voicebot
  - 2.2 Integrating MCP to Nexus for Agent Assist

- Administrator

Learn how to provision Genesys Voice Platform.

Related documentation:

-
-
-

RSS:

- For private edition

This page includes the following sections:

- Tenant provisioning
- Provisioning Nexus connection in GVP

Tenant provisioning

GVP Service Discovery (**SD**) container is used for provisioning tenant object in GVP Configuration Server and creating application objects for - Configuration Server, Media Control Platform app objects under Resource Manager logical resource group.

Overview

GVP Service Discovery (**SD**) container (running in **K8s**) does the following:

1. Runs a timer that gets invoked every **`1 min`** by default (this is configurable).
2. Checks **Consul** for the registered MCPs and then checks **GVP Configuration Server (CS)** for the MCPs present there and does the necessary addition/removal of MCPs from CS to sync with Consul data.
3. Checks the **`tenant-inventory`** configmap in the **`gvp`** namespace of the **K8s cluster** and, if configmap is updated from the last run, then based on the new data creates/updates tenant information.
4. Note that SD processes one tenant at a time.

The following **objects** are created in CS as part of GVP Tenant creation, and unless specified SD uses default values for those objects:

- The **Tenant** object itself with properties set in the **Annex** section
- The following **IVR Profiles**:
  - IVRAppDefault

    - conference
    - cpd
    - record
    - media
- The following **Transactions** object (used by **Recording Uploader**):
    - hybrid_integration

For Tenant creation, the following parameters **SHOULD** be specified:

- As part of Tenant provisioning, the tenant is registered to **GWS** and you get a **GWS-CCID**. This parameter is **mandatory** for GVP tenant creation.
- The **id** for the tenant is a **mandatory** parameter. This can be arbitrary string, but the preferable value is the **last 4-digits of the GWS-CCID**.
- The tenant **name** parameter is **preferred** to be populated.
- The **default-application** parameter should be set to **IVRAppDefault** always.
- The provisioned parameter should be set to 'true', unless the git pipeline workflow does that. Once this is set, then Service Discovery populates the tenant's contact center id (uuid) and the tenant's default IVR-Profile's dbid in Consul KV store.

Provision Consul to provide `gvp_config_dbid` and `ivr_app_dbid` parameters with Tenant ID and default IVR application ID values from the GVP Configuration Server database.

Where,

- **gvp_config_dbid** is the DB ID for a particular tenant.
- **ivr_app_dbid** is the DB ID for the default IVR application for that tenant.

Example request:

```
curl -H "Authorization: Bearer " https:///v1/kv/tenants//gvp
{
"gvp_config_dbid": "161",
"ivr_app_dbid"  : "217"
```

Use Case 1 - Deploy New Tenant

- As mentioned above, deploying a new tenant with SD and CS simply boils down to creating a configmap in your K8s cluster under gvp namespace.

Note: SD doesn't support creating multiple tenants in bulk, so you need to provide one JSON data file per tenant and repeat the process for multiple tenants.

- A bare minimum JSON should contain the minimal set of parameters mentioned above:

```
{
```

```
    "name": "CustomerX",
    "id": "2026",
    "gws-ccid": "285bd12f-5e4a-4c76-ad93-752ee1a82026",
    "default-application": "IVRAppDefault"
}
```

- Delete the existing **tenant-inventory** configmap if any: **`kubectl -n gvp delete configmap tenant-inventory --ignore-not-found`**
- Create the configmap with your JSON file: **`kubectl -n gvp create configmap tenant-inventory --from-file tenant-2026.json`**
- This configmap is mounted as a **volume** in SD - so the new JSON is updated in the **`/etc/config`** folder of the SD container.

```
[genesys@gvp-sd-bfcdd567f-8hrzm config]$ pwd
/etc/config
[genesys@gvp-sd-bfcdd567f-8hrzm config]$ ls -ls
total 0
0 lrwxrwxrwx 1 root root 27 May  5 12:59 tenant-2026.json > ..data/tenant 2026.json
```

- In the **next cycle**, SD will detect the new file and process it - thus creating/updating tenant and the associated objects.
- Once processed, SD **ignores** the file in subsequent cycles.

Use Case 2 - Update Existing Tenant

Note: Service Discovery cannot change/update configuration for Environment tenant. This limitation will be addressed in future release.

The mechanism for updating existing tenant is very similar to deploying new tenant. In this case, the existing JSON for the tenant needs to be updated with new parameters.

Note: You MUST always keep the minimal set of parameters present in the JSON even if in the case of update.

Example 1 - Recording destination provisioning for recording uploader

WEM provisioning is supported through the **hybrid_integration** transactions object. Only change is to specify the additional **WEM parameters** in the **Tenant JSON** file. An example JSON file with the WEM parameters may look like the following:

```
{
    "name": "CustomerX",
    "id": "2026",
    "gws-ccid": "285bd12f-5e4a-4c76-ad93-752ee1a82026",
    "default-application": "IVRAppDefault",
    "provisioned": "true",
    "transactions": {
```

```
        "name": "hybrid_integration",
        "recording-uploader.destFolder": {
            "destType": "Folder"
        },
        "recording-uploader.destFolder.mediaUpload": {
            "folder_path": "/rup/recordings"
        }
    }
}
```

The rest of the steps are the same as above. Delete existing configmap and create new with updated JSON.

Provisioning Nexus connection in GVP MCP

Integrating MCP to Nexus for Dialogflow voicebot

This section explains how to integrate MCP to Nexus for Dialogflow voicebot.

Pre-requisites

The following are the pre-requisites for integrating MCP to Nexus for Dialogflow voicebot:

1. The Nexus bot endpoint URL. For example:

   ```
   ws://nexus-production.nexus.svc.cluster.local/nexus/v3/bot/connection
   ```

2. You should register the Tenant to Nexus.
3. Verify that the Nexus API key is configured for the tenant by logging into the Tenant's Configuration Server > Transactions > DesginerEnv > Nexus.

Configuration

Update the following parameters in the section **mcpConfig** in the **values.yaml** file and deploy/redeploy MCP:

```
nexus_asr1.pool.size: 1

  nexus_asr1.provision.vrm.client.resource.uri: "[\"ws://nexus-
production.nexus.svc.cluster.local/nexus/v3/bot/connection\"]"

  nexus_asr1.provision.vrm.client.resource.type: "ASR"

  nexus_asr1.provision.vrm.client.resource.name: "ASRC"

  nexus_asr1.provision.vrm.client.resource.engines: "nexus"

  nexus_asr1.provision.vrm.client.resource.engine.nexus.audio.codec: "mulaw"

  nexus_asr1.provision.vrm.client.resource.engine.nexus.audio.samplerate: 8000
```

```
nexus_asr1.provision.vrm.client.resource.engine.nexus.enableMaxSpeechTimeout: true

nexus_asr1.provision.vrm.client.resource.certificate: ""

nexus_asr1.provision.vrm.client.resource.privatekey: ""

nexus_asr1.provision.vrm.client.resource.proxy: ""

nexus_asr1.provision.vrm.client.TransportProtocol: "WEBSOCKET"
```

The mandatory parameters are:

```
nexus_asr1.pool.size
nexus_asr1.provision.vrm.client.resource.uri.
```

You can leave the remaining parameters as default.

For more information on the above-mentioned parameters, refer to Configure Genesys Voice Platform.

Validation

Make test calls and check the following statements in MCP logs:

```
Int 50148 5A1A9632-10053FC2 140635978869056 asr_open 5A1A9632-10053FC2-asr-nexus-471699/
success
Int 50149 5A1A9632-10053FC2 140635978869056 asr_close 5A1A9632-10053FC2-asr-nexus-471699
Int 50159 5A1A9632-10053FC2-asr-nexus-471699 140635978869056 ws_stats 5c49970a-d41a-91f5-6ece-
af6d71bf2120 Tx: total 477/76320, sent 477/76320, failed 0/0, dropped 0/0 Rx: 0/0
```

Integrating MCP to Nexus for Agent Assist

This section explains how to integrate MCP to Nexus for Agent Assist.

Pre-requisites

The following are the pre-requisites for Integrating MCP to NEXUS for Agent Assist:

1. The Nexus Agent Assist endpoint URL. For example:

   ```
   ws://nexus-production.nexus.svc.cluster.local/athena/v1/agent-assist/voice/connection
   ```

2. You should register the Tenant to Nexus.
3. The GWS URL and client secrets. For example:

   ```
   https://gauth-int.nlb02-westus2.int.dev.genazure.com/auth/v3/oauth/token
   ```

Configuration

For configuration, complete these steps:

1. Create a new Kubernetes secret:

   ```
   apiVersion: v1
    kind: Secret
   ```

```
metadata:
  name: shared-gauth-gvp-client-secret
  namespace: gvp
type: Opaque
data:
 gauth-gvp-client-secret: $CLIENT_SECRET
```

Note: As regards Client secret, you must obtain the value from your GWS and replace “$CLIENT_SECRET " with the actual secret.

2. Update the following parameters in the section **secrets** in the **values.yaml** file:

```
gws:
 enabled: true
 clientName: "gvp_client"
 clientSecret:
   secretName: "shared-gauth-gvp-client-secret"
   secretKey: "gauth-gvp-client-secret"
```

3. Update the following parameters in the section **gws** in the **values.yaml** file and deploy/redeploy MCP:

```
gws:
  authEndpoint: $
```

Note: Replace “$" with the actual URL.

4. Add the following parameters to record the IVR profile of the Tenant:

```
"recordingclient.stream1.dest": "fixed,",
"recordingclient.stream1.dialogflow.engineId": "fixed,dfaa",
"recordingclient.stream1.encoding": "fixed,audio/mulaw",
"recordingclient.stream1.gauth": "fixed,true",
"recordingclient.stream1.xccid": "fixed,true",
"recordingclient.gvp.config.msml.record.enablesipfilerecording": "fixed,true"
```

Validation

Make test calls and check the following statements in the MCP logs:

```
Int 50156 F5099632-100028BD-0_268435867 140575662311744 streamer_open 0_268435867success
Int 50157 F5099632-100028BD-0_268435867 140575662311744 streamer_close 0_268435867
```

The MCP metrics section includes the following Nexus-related triggers:

MCP_WEBSOCKET_CLIENT_OPEN_ERROR	A websocket client error opening a connection to service, such as Nexus
MCP_WEBSOCKET_CLIENT_PROTOCOL_ERROR	A websocket client received a protocol error from the service, such as Nexus