



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Genesys Voice Platform Private Edition Guide

Provision Genesys Voice Platform

8/10/2022

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

-
- 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.

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

You can provision Nexus in GVP. The following sections include information on provisioning Nexus:

-
- Configure Genesys Voice Platform

The second table in the section "Media Control Platform" includes the list of "mcpConfig.nexus" parameters. More information is available in the section "Configuring MCPs to connect to Nexus (for Bot use cases)" after the table.

- Deploy Genesys Voice Platform

The section "Install Helm chart", for MCP, includes the values.yaml information on Nexus client configuration.

- `[[GVP/Current/GVPPEGuide/Metrics#GVPMCP|]]`

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 |