Genesys Interaction Recording Solution Guide

GIR Voice Processor deployment using Podman

4/2/2025

Contents

GIR Voice Processor deployment using Podman

This deployment is applicable for Voice Processor 9.0.000.39 or higher for installing GIR VP using Podman.

Currently GIR Voice Processor is deployed using Docker swarm in Premise and through Elastic Container Service in AWS. Due to known issues with Docker Swarm and Docker in RHEL 8, GIR Voice Processor is moving towards Podman for deploying Voice Processor.

Limitations

- Currently there are no alternatives for Docker swarm features with Podman. We may have to use a load balancer for load balancing with Podman.
- Network mode *overlay* is not supported in Podman, we will use *host* network mode.

Prerequisites

- Podman 4.9 or higher on x86_64 Linux host.
- Podman-compose 1.0.6 or higher (On systems with python3.6 only podman-compose 1.0.6 is supported).
  - `podman --version`
- PostgreSQL 12.11 or higher.
- Genesys Info Mart 8.5 or higher installed on Microsoft SQL Server or PostgreSQL. For information on the system requirements for GIM, see Genesys Info Mart Requirements.
- Interaction Recording Web Services (RWS) 8.5.201.90 or higher.
- SpeechMiner 8.5 or higher if you are using SpeechMiner. We recommend that you install Speechminer before deploying Voice Processor.

We recommend that you have the following details before proceeding with deployment:

- Host name, port, database name for Genesys Info Mart, and user (read-only) credentials.
- Host name and port for the Interaction Recording Web Services (RWS), and Operation Admin (ops) credentials.
- Configuration Manager credentials for an account with access to the IVR Profile.
- Host name, port, and credentials needed to post to SpeechMiner Interaction Receiver. These details are required only for new installations.

Preparing Podman environment

Extracting installation files

You can download the GIR VP image from the Genesys customer portal. The GIR VP image is a .tar file that contains the installation and configuration files required to set up and run the Voice Processor. Extract and copy the files from the image using the following steps:

1. Load the GIR VP image.
   ```
   zcat <.tar file> | podman load
   ```
2. View the list of container images and make a note of the newly loaded image.
   ```
   podman image ls
   ```
3. Add a custom tag to the image for your reference.
   ```
   podman tag <image ID> <tag>
   ```
4. Copy the files from the image.
   ```
   id=$(podman run --rm -dt <image> cat) && podman cp $id:/rps/compose . && podman stop $id
   ```
   In the above command, *--rm* option will delete the image file after creating the Podman container.

Now you have the sample configuration files in **./compose/defaults**, an Ansible playbook in **./compose** to help you set up and run the Voice Processor, and an SQL file for database setup. You will need these files for installing and configuring the Voice Processor.

Podman logs

Podman logs location can be find using:

```
podman inspect --format='{{.HostConfig.LogConfig.Path}}' <container-id>
```

Podman container logs can be accessed simply by `podman logs <container name>`.

Configuring Voice Processor

This section contains the following sub-sections:

- PostgreSQL database configuration
- Service level configuration Documentation:CR:Solution:VP:8.5.2
- Genesys Voice Platform profile configuration Documentation:CR:Solution:VP:8.5.2
- Tenant level configuration Documentation:CR:Solution:VP:8.5.2
- GIM DB ETL configuration Documentation:CR:Solution:VP:8.5.2

PostgreSQL database configuration

The Voice Processor requires a service-specific database that tracks work in progress items. This

database runs on a PostgreSQL server. Set up the database using the following steps:

1. Create a database in your PostgreSQL server for the Voice Processor.
2. Create a PostgreSQL user and grant all privileges to the database that you created in the previous step.
3. Assign a password to the user that does not contain a backward slash (\) or quotation marks, as they might cause issues later.
4. Make a note of the database name, user name, and password — they are needed when configuring the Voice Processor in later steps.
5. Confirm that the **standard_conforming_strings** parameter of the PostgreSQL server is set to `on` (default).

> **Important**
>
> - GIR Voice Processor must have a separate PostgreSQL DB from Config Server since the Voice Processor PostgreSQL DB requires the `standard_conforming_strings` setting to be on and the Config Server PostgreSQL DB requires the `standard_conforming_strings` setting to be off.
> - Voice Processor supports connections to PostgreSQL DB when `password_encryption` is set to md5 or scram-sha-256 in the **postgresql.conf** file.
> - When using a PostgreSQL DB for Genesys Info Mart, if `password_encryption` is set to scram-sha-256 in the **postgresql.conf** file, the Genesys Info Mart version must be 8.5.016.04 or higher.

6. Run the provided script, `create_node_rps_tables_v2.sql`, against this new database to provision it.

> **Important**
>
> To avoid possible conflicts with their settings requirements, Genesys recommends not hosting the Voice Processor and Configuration Server databases on the same PostgreSQL instance.

Service level configuration

You can follow the instructions provided with the configuration files available in the default directory. You can copy the provided configuration files and make changes to your copies. We recommend that you use a version control repository to store your configurations. Add the PostgreSQL database, user name, and password to the **nodeRpsDb** setting in your copy of **settings-override.json**.

Voice Processor database settings

To enable TLS connection to the Voice Processor database, set the **ssl** parameter to `true` and configure the **trustedCA** parameter under **nodeRpsDb** in **settings-override.json**.

```
nodeRpsDb:
  database: <database name>
  host: <db server hostname>
  port: <db port>
  user: <db user>
  ssl: < true / false >
  trustedCA: false / true / "<path to root certificate>"
```

The **ssl** parameter is optional and its default value is `false`. When you set it to `true`, the Voice Processor establishes a secure connection to the GIM database using TLS 1.2. Additionally, when the **ssl** parameter is set to `true`, the **trustedCA** parameter can be interpreted as follows:

- Do not authenticate the server certificate when the **trustedCA** value is `false`.
- Authenticate the server certificate against the system's root authorities when the **trustedCA** value is `true`.
- Authenticate the server certificate against the specified root authorities. Set **vpdb_ca_cert** in your copy of **settings-override.json** with the `<path to root certificate>` value.
- While enabling TLS on the Voice Processor service, please validate that Podman has access to certificate files.

Voice Processor HTTPS settings

The **rwsBaseUri** setting in **settings-override.json** supports HTTPS. For example:

`https://<RWS hostname>:<RWS port>`

To use HTTPS on the Voice Processor service API, set **https** to `true` in your copy of **settings-override.json** . You must provide the server private key, public key, and path to the files.

```
https: true
tls:
  privkey: <path to the private key file>
  pubkey: <path to the public key file>
```

MCP post basic authentication

Add the following lines to the **settings-override.json** file to enable basic authentication for the endpoint used by the MCP to post recording metadata:

```
authUsername: "<basic auth username>"
authPassword: "<basic auth password>"
```

If you add these options, you must also configure the Voice Platform profile option, **recording client.callrec_authorization**, in the **[gvp.service-parameters]** section to match these credentials. As basic authentication involves sending the credentials in plain text format, we strongly recommend that you use TLS for maximum security. Note that the other Voice Processor endpoints are not authenticated. Therefore, you must install the Voice Processor behind a firewall or API gateway to restrict access. You can obtain a summary of endpoints exposed by the Voice Processor service by accessing: `http://<GIR VP hostname>:<port>/apidoc`

Setting the GR Voice Processor image

To find the values for `<image ID>:<tag>`, use the `podman image ls` command.

Genesys Voice Platform profile configuration

Use HTTPS protocol in the Voice Processor URL when HTTPS is enabled in the Voice Processor service API.

```
recordingclient.callrec_dest = fixed,https://<VP hostname>:<VP port>/api/contact-centers/<CCID>/recordings/
```

Use HTTPS protocol in the SpeechMiner Interaction Receiver URL when HTTPS is enabled on the SpeechMiner Interaction Receiver. When using HTTPS for the SpeechMiner URL, by default, the Voice Processor does not validate SpeechMiner server certificate. You can set **sm_ca_cert** in your copy of **settings-override.json** with the `<path to root certificate>` value to authenticate the server certificate against the specified root authorities.

```
recordingclient.rp.speechminer_uri: fixed,https://<Speechminer backend hostname>/interactionreceiver/
```

Tenant level configuration

As the Voice Processor is designed to support Genesys cloud multi-tenancy model, settings that may vary from tenant to tenant are stored in an RWS group settings called **rps-provisioning**:

> **Important**
>
> You need an Ops Admin user account to access these settings. For more information on how to update settings in RWS, see Settings API.

You must specify the Ops Admin user name and password in your copy of **settings-override.json**. The tenant level configuration values are set to the RWS group settings **rps-provisioning** using HTTP POST. For example:

```
curl -u <Ops admin user>:<password> -X POST -H "Content-Type: application/json" <rwsBaseUri>/api/v2/ops/contact-centers/<ccid>/settings/rps-provisioning -d @rps-settings.json
```

Where **rps-settings.json** contains settings like: `eventDataFilters`, `gimDb`, `rwsPostRecBaseUri` and others.

To confirm the Voice Processor per tenant settings, use HTTP GET. For example:

```
curl -u <Ops admin user>:<password> -X GET "<rwsBaseUri>/api/v2/ops/contact-centers/<ccid>/settings/rps-provisioning?location=*&ignoreParentLocations=false"
```

GIM database

You must provide information needed to access the tenant's GIM database. To enable TLS connection to the GIM database, set the **ssl** parameter to `true` and configure the **trustedCA** parameter under GIM database settings in tenant level configuration.

```
{
    "name": "gimDb",
    "value": {
        "primary": {
            "host": "<GIM server hostname>",
            "port": <GIM server port (default 5432 for Postgres, 1433 for MS SQL)>,
            "user": "< DB user name >",
            "database": "<database name",
            "password": "<DB user password>",
            "dbType": "<postgres or mssql, default postgres>",
            "ssl": < true / false >,
            "trustedCA": false / true / "<path to root certificate>",
        },
        "backup": {
            < same settings as for primary >
        }
    }
}
```

The **ssl** parameter is optional and its default value is `false`. When you set it to `true`, the Voice Processor establishes a secure connection to the GIM database using TLS 1.2. Additionally, when the **ssl** parameter is set to `true`, the **trustedCA** parameter can be interpreted as follows:

- Do not authenticate the server certificate when the **trustedCA** value is `false`.
- Authenticate the server certificate against the system's root authorities when the **trustedCA** value is `true`
- Authenticate the server certificate against the specified root authorities by performing the following steps:
  1. Set **gim_ca_cert** in your copy of **settings-override.json** with the `<path to root certificate>` value.
  2. Set **trustedCA** to `/rps/rpsdata/gimCA` in GIM database settings to be posted to tenant level configuration.

The **backup** parameter is optional. You can omit it if there is only one GIM database available.

RWS posting

You must specify the RWS instance to which recordings are posted. As this is a region-based setting, multi-regional deployments can ensure that recording data stays within the jurisdictional boundaries. The Voice Processor instance selects the location identified through the nodePath of the RWS server from which the setting is retrieved or the nearest matching parent. The **backup** parameter is optional. The URL supports HTTPS. When using HTTPS for the RWS URL, by default, the Voice Processor does not validate RWS server certificate. You can set **rws_ca_cert** in your copy of **settings-override.json** with the `<path to root certificate>` value to authenticate the server certificate against the specified root authorities.

For example, the following setting applies to all Voice Processor instances:

```
{
    "name": "rwsPostRecBaseUri",
    "location": "/",
    "value": {
          "primary": "http://<hostname>:<port>{/<optional routing prefix>}",
          "backup": "http://<hostname>:<port>{/<optional routing prefix>}"
    }
}
```

The following setting would override the above global setting for Voice Processor instances that retrieved the setting from an RWS node with nodePath `/US` or `/US/*` :

```
{
    "name": "rwsPostRecBaseUri",
    "location": "/US",
    "value": {
          "primary": "http://<hostname>:<port>{/<optional routing prefix>}",
          "backup": "http://<hostname>:<port>{/<optional routing prefix>}"
    }
}
```

Event filtering

You can use filters to remove unwanted data from the recording metadata. The event filtering settings are similar to RPS except the mechanism of how the default filters are disabled.

```
{
    "name": "eventDataFilters",
    "value": {
        "attachedDataFilter": "regexp for new attached data filter",
        "attachedDataFilterException": "regexp for new attached data filter exception",
        "acwCustomDataFilter": "regexp for new ACW data filter",
        "acwCustomDataFilterException": "regexp for new ACW data filter exception"

        -- or, to disable the default filters or filter exceptions --

        "disableAttachedDataFilter": true,
        "disableAttachedDataFilterException": true,
        "disableAcwCustomDataFilter": true,
        "disableAcwCustomDataFilterException": true
    }
}
```

The default filters are:

- **attachedDataFilter**: `^ORSI:|^WWE|^PegAG`
- **attachedDataFilterException**: `^(GRECORD_(PARTITIONS|PROGRAM)|GSRS_STATE|GSIP_REC_FN)$`
- **acwCustomDataFilter**: `^ORSI:|^WWE|^PegAG`
- **acwCustomDataFilterException**: `^(GRECORD_(PARTITIONS|PROGRAM)|GSRS_STATE|GSIP_REC_FN)$`

Complete after-call work (ACW) threshold

The ACW threshold indicates how long the Voice Processor waits, in minutes, following the end of an interaction to update custom data. Custom data entered by agents after this interval is not added to recording metadata. The default value is zero.

```
{
    "name": "acwThresholdMinutes",
    "value": <ACW wait interval in minutes>
}
```

GIM DB ETL configuration

You must configure the GIM ETL application properly to ensure recording metadata is posted from the Voice Processor to SpeechMiner in a timely manner.

The **etl-start-time**, **etl-end-time**, and **etl-timezone** options in the **[schedule]** section are used to configure a daily maintenance period during which population of GIM data is paused for maintenance purpose. New recordings posted to the Voice Processor during this period are not processed and they are held temporarily in a database until the maintenance period finishes and the relevant GIM data becomes available. You must configure the **maintain-start-time** option such that the GIM ETL maintenance job begins and completes during the maintenance period.

The **etl-frequency** option in the **[schedule]** section is used to specify the cycle time of the GIM ETL jobs that populate the recording metadata used by the Voice Processor. We recommend that you use the default value of one minute. Note that any time longer than 3 minutes may cause subsequent delays in recording posts. If a longer **etl-frequency** setting is used, then the value of the Voice Processor service setting, **rpsInitialInteractionTimeout**, should be increased accordingly.

The **user-event-data-timeout** option in the **[gim-etl]** section is used to ensure that custom attached data entered during after-call work is captured. You can increase the default value of one hour if your agents will spend more than a few minutes in after-call work.

Important

Consult Genesys before setting non-default values for the following options.

The **max-call-duration**, **merge-failed-is-link-timeout**, and **extract-data-stuck-threshold** options in the **[gim-etl]** section must be configured properly to ensure completeness of the call metadata recorded in GIM. For more information on these options, see Operations-Related Options for Genesys Info Mart.

Deploying and Starting Voice Processor

Deploy Voice Processor to the newly configured Podman image, referencing your copies of the default configuration files. This step also starts Voice Processor.

Before deploying, the **settings-override.json** file (or the yaml files you have designated to provide these settings) have several mandatory parameters that must be configured, as described below.

In the **settings-override.json** file, the following parameters are required:

- **rwsBaseUri** – Specifies the address of the RWS cluster that will provide Voice Processor with contact center settings, including tenant-specific configurations such as Genesys Info Mart (GIM) database information and ACW Wait Time. Example: <code>http://some-rws-host.com:8090</code>
- **region** – Controls the `region` section in the metadata POSTed to RWS. This must match the **crRegion** setting of the RWS cluster to which recordings are posted. Example: `usa`

- **nodeRpsDb** – This section specifies the Voice Processor Persistence Database, which is required for storing recording metadata while Voice Processor is processing them.
  - **database** – Database on the host that will hold recording metadata. Example: `noderpsdb`
  - **host** – Host of the Persistence Database. Example: `noderpsdb.com`
  - **port** – Port that the Persistence Database is listening on. Example: `5432`
  - **user** – The user name to connect to the Persistence Database.
  - **password** – The password to connect to the Persistence Database.
- **rwsUsername** – The user name for authenticating with RWS.
- **rwsPassword** – The password for authenticating with RWS.

For an example of how the yaml files should be structured, you can refer to the default yaml files that were included with Voice Processor. These files are located at **<INSTALL_DIR>/defaults/**, where **<INSTALL_DIR>** is the location where you extracted the installation files to during the Preparing your Podman environment step.

After configuring the default configuration files, deploy and start Voice Processor:

```
sudo ansible-playbook \
       -e docker_config=<defaults directory of Your Docker Configuration Yaml (e.g. docker-
config.yml)> \
       -e logger_config=<defaults directory of Your Logger Configuration Yaml (e.g. logger-
config.yml)> \
       -e settings_override=<defaults directory of Your Voice Processor Configuration Yaml
(e.g. settings-override.yml)> \
       -e secrets=<defaults directory of Your Voice Processor Secrets Yaml (e.g.
secrets.yml)> \
       -e service_user=<user name under which gir vp service is running>
       -e service_group=<group name under which gir vp service is running>
        gir-vp-playbook.yml
```

> ### Important
>
> If the above options are not specified, then the .yml files in **./compose/defaults** will be used.

After starting the Voice Processor, update the Voice Processor endpoint (**/api/active-version**) with the version of your Voice Processor instance. You do not require any credentials to do this.

The setting to post the active version:

```
{ "version":"<GIR VP Version>" }
```

Example

```
curl -X POST -H "Content-Type: application/json" -d '\{ "version": "9.0.000.25" }'
girvp.company.com/api/active-version
```

GIR VP container will run as a Systemctl service: `girvp.service`.

When a container stopped, Systemctl will start a new Podman container:

```
sudo systemctl status girvp.service
```

Validating

1. Place a call to an agent or a test agent that is configured for recording.
2. Verify that the call arrives at the SpeechMiner UI. It should take 5 to 15 minutes depending on your configured ACW wait setting.
3. Assuming that live traffic is not recorded, you can use the health check endpoint `<domain:port>/api/status?verbose=1` . The items *recordingsInProcess* or the MCP Post operational status can be helpful in determining whether or not the recording is arriving at the Voice Processor. This also helps you to isolate GVP configuration problems from problems with the Voice Processor service. If a load balancer is used, the node serving the health check may not be the one that handled the recording. Therefore, several health checks may be required to cover the whole cluster.

Upgrading

Docker object configurations and secrets cannot be upgraded. We recommend that you remove the container, update the required configurations, and redeploy.

```
podman container rm <gir_vp>
```

After starting the Voice Processor, update the Voice Processor endpoint (**/api/active-version**) with the version of your Voice Processor instance. You do not require any credentials to do this.

The setting to post the active version:

```
{ "version":"<GIR VP Version>" }
```

Example `curl -X POST -H "Content-Type: application/json" -d '\{ "version": "9.0.000.25" }' <hostname/Ip address>:8889/api/active-version`

Miscellaneous Podman tips

- To view a list of running containers:

  ```
  podman ps
  ```

- To view a list of all containers (started/stopped):

  ```
  podman ps -a
  ```

- To view the container logs:

  ```
  podman logs <container name>
  ```

- To remove everything to start again:

  ```
  podman stop <gir_vp> podman container rm <gir_vp>
  ```

Load balancing with Podman

GIR VP supports load balancing with Nginx and httpd. To set up load balancing in a Premise environment, see Setting up the Load Balancer in a Single-Tenant Environment.