



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.

# Voice Microservices Private Edition Guide

[Configure Voice Microservices](#)

---

## Contents

- 1 Override Helm chart values
  - 1.1 Deployment section
  - 1.2 Image section
  - 1.3 Config section
  - 1.4 Secrets section
  - 1.5 HPA section
  - 1.6 Resources section
  - 1.7 Log volume
- 2 Configure Kubernetes
- 3 Configure security
  - 3.1 Security context configuration
  - 3.2 Secrets for Voice services

---

Learn how to configure Voice Microservices.

### Related documentation:

- 
- 
- 

### RSS:

- [For private edition](#)

## Override Helm chart values

For general information about overriding Helm chart values, see *Overriding Helm Chart values in the Genesys Multicloud CX Private Edition Guide*.

When deploying Voice services, certain parameters must be enabled or modified based on customer requirements and environment. For each of the Voice services, an override **values.yaml** file must be created that overrides certain sections of the default configuration for the service. In this document, we use the following format for creating an override **values.yaml** file: **\_override\_values.yaml**.

The **\_override\_values.yaml** file contains the following sections:

- Deployment
- Image
- Config
- Secrets
- HPA
- Resources
- Log volume

### Deployment section

This section can be used to specify minimum and max instances that will be started for each service. By default, the minimum replica count is 1, and the maximum replica count is 10. You can modify it per your load requirements. For RQ service alone it is recommended to set replica count to 2 or more based on load for high availability.

```
deployment:
namespace: voice      # Namespace of voice service
replicaCount: 1      # Min replica count when service is deployed
```

---

```
maxReplicas: 10      # Max replica count to which the service will scale.
```

## Image section

This section has information about the registry from which the voice services will be deployed.

```
image:
  registry: pureengage-docker-staging.jfrog.io # registry from where image needs to be
  deployed
  pullPolicy: Always                          # whether to pull image always
  imagePullSecrets: "mycred"                  # Secrets needed for pulling image from
  registry
```

## Config section

The config section contains configuration parameters that need to be overridden for all voice services.

Additional information needs to be passed for SIP Cluster Service: dnsServer. Get the DNS Server value from the above section (Configure DNS server for voice-sip).

```
# Set the redis port to be used.
context:
  envs:
    redis:
      port: 6379          # Redis port
      dnsServer: "10.202.0.10" # DNS server address. Needed only for SIP Service.
```

## Secrets section

This section captures all the secrets needed by voice services for connecting to infraservices (Consul, Kafka, Redis). The default values for Redis and Kafka secrets are the same as what is created above.

```
# set the secrets
secrets:
  redisCache:
    general:
      enabled: true
  consulACL:
    volumes:
      - name: consul-shared-secret
        secret:
          secretName: consul-voice-token
```

## HPA section

The HPA section captures whether HPA is enabled for a service or not and what is the CPU and memory percentage used for scale up and scale down. Common HPA for the following voice services: Agent Service, Config Service, Call State Service, Registrar Service, SIP Front End service, Dial Plan Service.

```
hpa:
  targetCPUPercent: 60      # Average CPU percentage which determine scale up and down
  targetMemoryPercent: 60  # Average Memory percentage which determine scale up and down
  enabled: true             # Horizontal Pod scalar enabled
```

---

For SIP Proxy and RQ Services, HPA is set to false:

```
hpa:
  enabled: false          # Horizontal Pod scalar enabled
```

For SIP Cluster and Orchestration Services, HPA is set as follows:

```
hpa:
  targetCPUPercent: 50   # Average CPU percentage which determine scale up and down
  targetMemoryPercent: 50 # Average Memory percentage which determine scale up and down
  enabled: true          # Horizontal Pod scalar enabled
```

## Resources section

This section captures the resource request and limits for each voice service. The default resource given below is set for each service. You can modify this request and limit based on your load requirement.

```
resources:
  requests:
    cpu: "250m"
    memory: "256Mi"
  limits:
    cpu: "500m"
    memory: "512Mi"
```

For Orchestration and SIP Cluster Services, the CPU and memory requirement is high. Genesys recommends the following setting:

```
resources:
  requests:
    cpu: "500m"
    memory: "1Gi"
  limits:
    cpu: "1500m"
    memory: "4Gi"
```

## Log volume

This section captures parameters pertaining to log volumes needed by SIP Cluster Service. These parameters are needed for storing logging of SIP Server binary, which runs inside the SIP Cluster Service. Configure the values for **storageClass** and **volumeName** based on the recommendation given in the Persistent Volume section.

```
# pvc will be created for logs
volumes:
  pvcLog:
    create: true
    claim: sip-log-pvc
    storageClass:
    volumeName:

  pvcJsonLog:
    create: true
    claim: sip-json-log-pvc
    storageClass:
    volumeName:
```

---

```
log:
  mountPath:

jsonLog:
  mountPath:
```

## Configure Kubernetes

For information, see the following resources:

- [Override Helm chart values](#)
- [Configure security](#)
- [Secrets for Voice services](#)
- [Deploy Voice Microservices](#)

## Configure security

Before you deploy the Voice Microservices, be sure to read Security Settings in the *Setting up Genesys Multicloud CX Private Edition* guide.

### Security context configuration

The security context settings define the privilege and access control settings for pods and containers. For more information, see the Kubernetes documentation.

By default, the user and group IDs are set in the **values.yaml** file as 500:500:500, meaning the **genesys** user.

```
containerSecurityContext:
  readOnlyRootFilesystem: false
  runAsNonRoot: true
  runAsUser: 500
  runAsGroup: 500
```

```
podSecurityContext:
  fsGroup: 500
  runAsUser: 500
  runAsGroup: 500
  runAsNonRoot: true
```

### Secrets for Voice services

Create the following Kubernetes secrets for other infrastructure services:

1. Kafka
2. docker-registry

---

### 3. Redis

#### Kafka secrets

Kafka secrets must be created when Kafka is deployed. The secret is referenced in the Voice Microservices **values.yaml** file.

When Kafka is deployed without authentication, create the secret for Kafka as follows:

```
kubectl create secret generic -n voice kafka-secrets-token --from-literal=kafka-secrets={"bootstrap\":"}
for ex, kubectl create secret generic -n voice kafka-secrets-token --from-literal=kafka-secrets={"bootstrap\":"infra-kafka-cp-kafka.infra.svc.cluster.local:9092\"}
```

When Kafka is deployed with authentication, create the secret for Kafka using this method:

```
kubectl create secret generic -n voice kafka-secrets-token --from-literal=kafka-secrets={"bootstrap\":" , \"username\":" , \"password\":" }
for ex, kubectl create secret generic -n voice kafka-secrets-token --from-literal=kafka-secrets={"bootstrap\":"infra-kafka-cp-kafka.infra.svc.cluster.local:9092\", \"username\":"kafka-user\", \"password\":"kafka-password\"}
```

#### Redis secrets

Ensure Redis is installed before you deploy the Voice Services.

Use the following commands to create Redis secrets:

```
export REDIS_PASSWORD=$(kubectl get secret infra-redis-redis-cluster -n infra -o jsonpath="{.data.redis-password}" | base64 --decode)
kubectl create secret generic -n voice redis-agent-token --from-literal=redis-agent-state={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-callthread-token --from-literal=redis-call-state={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-config-token --from-literal=redis-config-state={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-tenant-token --from-literal=redis-tenant-stream={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-registrar-token --from-literal=redis-registrar-state={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-sip-token --from-literal=redis-sip-state={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-ors-stream-token --from-literal=redis-ors-stream={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-ors-token --from-literal=redis-ors-state={"password\":"$REDIS_PASSWORD"}
kubectl create secret generic -n voice redis-rq-token --from-literal=redis-rq-state={"password\":"$REDIS_PASSWORD"}
```

#### JFrog secrets

Use the following commands to create JFrog secrets:

```
kubectl create secret docker-registry --docker-server= --docker-username="$JFROG_USER" --docker-password="$JFROG_PASSWORD" -n voice
```