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.

WebRTC Private Edition Guide

8/28/2025

Table of Contents

Contents

Find links to all the topics in this guide.

Related documentation:

-
-

RSS:

- For private edition

WebRTC is a service available with the Genesys Multicloud CX private edition offering.

Overview

Learn more about WebRTC, its architecture, and how to support high availability and disaster recovery.

- About WebRTC
- Architecture
- High availability and disaster recovery

Configure and deploy

Find out how to configure and deploy WebRTC.

- Before you begin
- Configure WebRTC
- Configure Webphone
- Provision WebRTC
- Deploy
- Deploy Webphone
- Upgrade, rollback, or uninstall WebRTC

Observability

Learn how to monitor WebRTC with metrics, alerts, and logging.

- Observability
- Metrics and alerts
- Logging

About WebRTC

Contents

Learn about WebRTC and how it works in Genesys Multicloud CX private edition.

Related documentation:

-
-
-

RSS:

- For private edition

Web Real-Time Communication (WebRTC) is a real-time communication over the internet that enables an agent to connect with the Genesys contact center environment to perform their business operations.

WebRTC is a shared (multitenant) service that acts as the signaling and media gateway. The signaling gateway is used to interwork WebRTC with Session Initiation Protocol (SIP), and the media gateway is used to terminate the Interactive Connectivity Establishment (ICE) and Secure Real-time Transport Protocol (SRTP).

WebRTC bridges the calls that are initiated/received by the browser. The SIP Server handles these calls as a SIP call to provide core Genesys features such as routing and IVR. These features are handled by Genesys for browser endpoints with the help of MCP in the call flow. Third-party component CoTURN is used to implement TURN and STUN servers.

Supported Kubernetes platforms

WebRTC is supported on the following cloud platforms:

- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)

See the WebRTC Release Notes for information about when support was introduced.

Architecture

Learn about WebRTC architecture.

Related documentation:

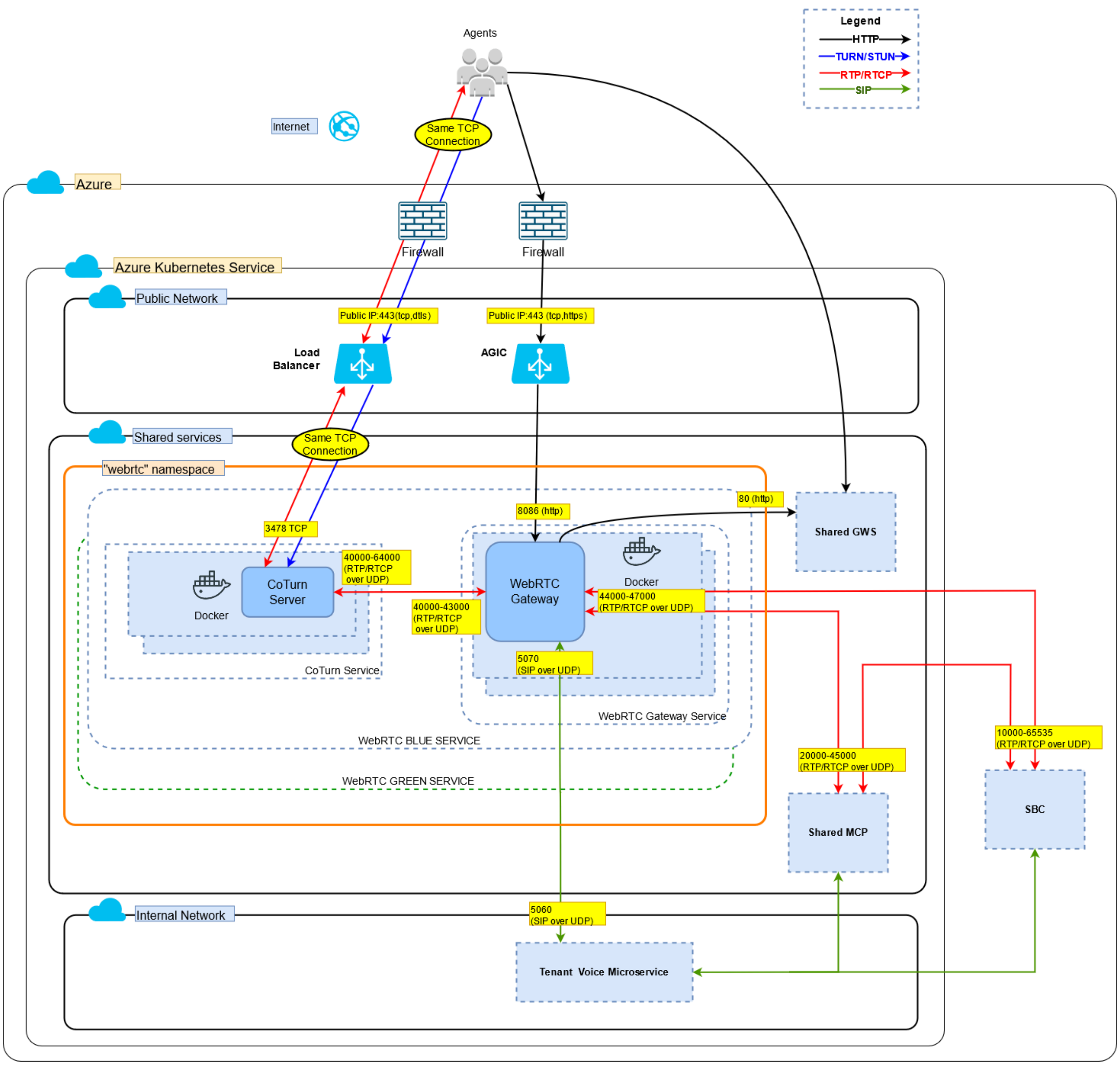
-
-
-

RSS:

- For private edition

For more information about the overall architecture of Genesys Private Edition Cloud, see: Architecture.

Genesys Web Services (GWS) provides Tenant specific information to WebRTC. Workspace Web Edition (WWE) Agent Workspace retrieves all the required information such as tenant ID and WebRTC locations from GWS and sends them to WebRTC.


Legend
HTTP
TURN/STUN
RTP/RTCP
SIP
Agents
Internet
Same TCP Connection
Azure
Firewall
Firewall
Azure Kubernetes Service
Public Network
Public IP:443(tcp,dtls)
Public IP:443 (tcp,https)
Load Balancer
AGIC
Shared services
Same TCP Connection
"webrtc" namespace
8086 (http)
80 (http)
Shared GWS
3478 TCP
40000-64000 (RTP/RTCP over UDP)
CoTurn Server
Docker
Docker
40000-43000 (RTP/RTCP over UDP)
WebRTC Gateway
44000-47000 (RTP/RTCP over UDP)
5070 (SIP over UDP)
CoTurn Service
WebRTC Gateway Service
WebRTC BLUE SERVICE
WebRTC GREEN SERVICE
10000-65535 (RTP/RTCP over UDP)
20000-45000 (RTP/RTCP over UDP)
SBC
Shared MCP
Internal Network
5060 (SIP over UDP)
Tenant Voice Microservice

High availability and disaster recovery

Find out how this service provides disaster recovery in the event the service goes down.

Related documentation:

-
-
-

RSS:

- For private edition

Service	High Availability	Disaster Recovery	Where can you host this service?
WebRTC Media Service	N = N (N+1)	Active-spare	Primary or secondary unit

See High Availability information for all services: High availability and disaster recovery

Resources that are used for Primary region should be replicated for the backup region. For example, if a primary region has 10 WebRTC pods, the same number of pods must be deployed in the backup region also.

If a call from Genesys Voice Platform (GVP) /Session Border Controller (SBC) is from one region and WebRTC is in another region, the bandwidth between the regions is used for the backup. For this scenario, Pulse Code Modulation mu-law (PCMU) codec is used which requires 100kbps per call.

WebRTC will follow the Workspace Web Edition (WWE) pattern for its failover. For example, If WWE primary is location 1 and the backup is location 2, WebRTC will also backup from location 1 to location 2.

Before you begin

Contents

Find out what to do before deploying WebRTC.

Related documentation:

-
-
-

RSS:

- For private edition

Limitations and assumptions

All prerequisites described under Third-party prerequisites, Genesys dependencies, and Secrets have been met.

Download the Helm charts

Download the Helm charts from the webrtc folder in the JFrog repository. See Helm charts and containers for WebRTC for the Helm chart version you must download for your release.

For information about how to download the Helm charts in Jfrog Edge, see the suite-level documentation: Downloading your Genesys Multicloud CX containers

WebRTC contains the following containers:

Artifact	Type	Functionality	JFrog Containers and Helm charts
webrtc	webrtc gateway container	Handles agents' sessions, signalling, and media traffic. It also performs media transcoding.	https:////webrtc/webrtc/
coturn	coturn container	Utilizes TURN functionality	https:////webrtc/coturn/
webrtc-service	Helm chart		https://// webrtc-service-.tgz

Third-party prerequisites

For information about setting up your Genesys Multicloud CX private edition platform, see Software requirements.

The following are the third-party prerequisites for WebRTC:

Third-party services

Name	Version	Purpose	Notes
Keda	2.0	Custom metrics for scaling. Use of Keda or HPA is configurable through Helm charts.	KEDA can be enabled or disabled for WebRTC. But, WebRTC cannot be configured to use HPA instead of KEDA.
Load balancer		VPC ingress. For NGINX Ingress Controller, a single regional Google external network LB with a static IP and wildcard DNS entry will pass HTTPS traffic to NGINX Ingress Controller which will terminate SSL traffic and will be setup as part of the platform setup.	
A container image registry and Helm chart repository		Used for downloading Genesys containers and Helm charts into the customer's repository to support a CI/CD pipeline. You can use any Docker OCI compliant registry.	
Command Line Interface		The command line interface tools to log in and work with the Kubernetes clusters.	

Storage requirements

WebRTC does not require persistent storage for any purposes except Gateway and CoTurn logs. The following table describes the storage requirements:

Persistent Volume	Size	Type	IOPS	Functionality	Container	Critical	Backup needed
webrtc-gateway-log-volume	50Gi	RW	medium	storing gateway log files	webrtc	Y	Y

webrtc-coturn-log-volume	50Gi	RW	medium	storing coturn log files	coturn	N	Y

Persistent Volume and Persistent Volume Claim will be created if they are configured. The size for them optional and should be adjusted according to log rate described below:

Gateway:

idle: 0.5 MB/hour per agent

active call: around 0.2MB per call per agent.

Example: For 24 full hours of work, where each agent call rate is constant and is around 7 to 10 calls per hour, we will require around ~500GB for 1000 agents, with around ~20GB being consumed per hour.

CoTurn:

For 1000 connected agents, the load rate is approximately 3.6 GB/hour which scales linearly and increases or decreases with the number of agents and stays constant whether calls are performed or not.

Network requirements

Ingress

WebRTC requires the following Ingress requirements:

- Persistent session stickiness based on cookie is mandatory. Stickiness cookie should contain the following attributes:
  - SameSite=None
  - Secure
  - Path=/
- No specific headers requirements
- Whitelisting (optional)
- TLS is mandatory

Secrets

WebRTC supports three types of secrets: CSI driver, Kubernetes secrets, and environment variables.

> **Important**
>
> GWS Secret for WebRTC should contain the following grants:

```
grant_type=authorization_code
grant_type=urn:ietf:params:oauth:grant-type:token-exchange
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer
grant_type=client_credentials
```

For GWS secrets, CSI or Kubernetes secret should contain `gwsClient` and `gwsSecret` key-values.

GWS secret for WebRTC must be created in the WebRTC namespace using the following specification as an example:

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: webrtc-gws-secret
  namespace: webrtc
data:
  client_id: XXXXX
  client_secret: YYYYY
```

ConfigMaps

Not Applicable

WAF Rules

The following Web Application Firewall (WAF) rules should be disabled for WebRTC:

WAF Rule	Number of rules
REQUEST-920-PROTOCOL-ENFORCEMENT	920300
	920440
REQUEST-913-SCANNER-DETECTION	913100
	913101
REQUEST-921-PROTOCOL-ATTACK	921150
REQUEST-942-APPLICATION-ATTACK-SQLI	942430

Pod Security Policy

Not applicable

Auto-scaling

WebRTC and CoTurn auto-scaling is performed by KEDA operator. The auto-scaling feature requires Prometheus metrics. To know more about KEDA, visit https://keda.sh/docs/2.0/concepts/.

Use the following option in YAML values file to enable the deployment of auto-scaling objects:

```
deployment:
  keda:              true
```

You can configure the Polling interval and maximum number of replicas separately for Gateway pods and CoTurn pods using the following options:

```
gateway:
  scaling:
  pollingInterval:  30
  maxReplicaCount:  100

coturn:
  scaling:
    pollingInterval:  30
    maxReplicaCount:  100
```

- Gateway Pod Scaling
  - Sign-ins

```
gateway:
  scaling:
   pollingInterval:   30
   maxReplicaCount:   100
   prometheusAddress: http://monitoring-prometheus-prometheus.monitoring:9090
   thresholdSignins: 25
```

- CPU based scaling

WebRTC auto-scaling is also performed based on the CPU and memory usage. The following YAML shows how CPU and memory limits should be configured for Gateway pods in YAML values file:

```
gateway:
  scaling:
   prometheusAddress: http://monitoring-prometheus-prometheus.monitoring:9090
   pollingInterval:    30
   maxReplicaCount:    100
   thresholdSignins:   25
   thresholdCpu:       60
   thresholdMemory:    60
```

- CoTurn Pod scaling

Auto-scaling of CoTurn is performed based on CPU and memory usage only. The following YAML shows how CPU and memory limits should be configured for CoTurn pods in YAML values file:

```
coturn:
  scaling:
   pollingInterval:    30
   maxReplicaCount:    100
   thresholdCpu:       60
   thresholdMemory:    60
```

SMTP settings

Not applicable

Browser requirements

Browsers

Name	Version	Notes
Firefox	Current release or one version previous	Genesys also supports the current ESR release. Genesys supports the transitional ESR release only during the time period in which the new ESR release is tested and certified. For more information, see Firefox ESR release cycle. Firefox updates itself automatically. Versions of Firefox are only an issue if your IT department restricts automatic updates.
Chrome	Current release or one version previous	Chrome updates itself automatically. Versions of Chrome are only an issue if your IT department restricts automatic updates.
Microsoft Edge Chromium	Current release	

Genesys dependencies

WebRTC has dependencies on several other Genesys services and it is recommended that the provisioning and configuration of WebRTC be done after these services have been set up.

Service	Functionality

GWS	Used for environment and tenants configuration reading
GAuth	Used for WebRTC service and Agents authentication
GVP	Used for voice calls - conferences, recording, and so on
Voice microservice	Used to handle voice calls
Tenant microservice	Used to store tenant configuration

For detailed information about the correct order of services deployment, see Order of services deployment.

GDPR support

Not applicable

Configure WebRTC

Contents

Learn how to configure WebRTC.

Related documentation:

-
-
-

RSS:

- For private edition

Override Helm chart values

Download the WebRTC Helm charts from JFrog using your credentials. Override the configuration parameters in the **values.yaml** file to provide deployment-specific values for certain parameters. You can override values in the Helm charts to configure Private Edition. For more information about overriding Helm chart values, see the "suite-level" documentation about how to override Helm chart values: Overriding Helm chart values

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
deployment.namespace	Name of Kubernetes namespace for WebRTC deployment	mandatory	webrtc	string	You can modify the default namespace used to deploy applications in the `deployment.namespace` option.	`deployment: namespace: production`
deployment.priorityClassName	Name of the priority class for pods that specify the importance of a pod relative to other pods	optional		string		
deployment.nodeSelector	Node selector for Gateway and CoTurn pods	optional		Specification		`deployment: nodeSelector: genesysengage.com/`

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
						nodepool: general
deployment.tolerations	Include this parameter in the Gateway and CoTurn, if the content of toleration exists.	optional		Specification		deployment: tolerations: - operator: Exists effect: NoSchedule key: "k8s.genesysengage.com/ nodepool"
deployment.ingress.domain	Ingress domain	mandatory		string		deployment: ingress: domain: apps.vce-c0.eps.genesys.com
deployment.ingress.annotations	WebRTC Annotation for ingress controller	mandatory		Specification	As the default value of the HAProxy route timeout is set to 30 s, there is a possibility it interferes with the WebRTC long-polling timeout (30 s) and disconnect the session.	deployment ingress: annotations: kubernetes.io/ ingress.class: nginx01-internal nginx.ingress.kubernetes. affinity: cookie nginx.ingress.kubernetes. affinity-mode: persistent nginx.ingress.kubernetes. ssl-redirect: "false" nginx.ingress.kubernetes. session-cookie-path: "/; Secure" nginx.ingress.kubernetes. session-

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
						cookie-samesite: None {{!}}- {{!}}{{!}}deployment.ingre {{!}}{{!}}If this option is defined, tls option is declared in the Ingress specification {{!}}{{!}}optional {{!}}{{!}} {{!}}{{!}}Specification {{!}}{{!}} {{!}}{{!}}deployment: ingress: tls: secretName: webrtc.api01-eastus2.dev.g tls-secret
deployment.affinity	Pod affinity descriptions	optional		Specification		deployment: affinity: podAntiAffinity: preferredDuringScheduling - weight: 100 podAffinityTerm: labelSelector: matchExpressions: - key: servicename operator: In values: - webrtc-gateway - webrtc-coturn

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
						`topologyKey: failure-domain.beta.kubernetes.io/zone`
deployment.dnsPolicy	Kubernetes DNS Policy that is applied in the Pods	optional				`deployment: nodeSelector: genesysengage.com/nodepool: general`
deployment.dnsConfig	All DNS settings must be provided using the dnsConfig field in the Pod specification	optional				`deployment: dnsConfig: options: - name: ndots value: "3"`
deployment.keda	Enable KEDA usage for the Gateway and CoTurn horizontal auto-scaling	optional	false	true/false		
deployment.coturnDeployment	Type of CoTurn deployment - `internal`: the internal LBs are created and the IP addresses of that LBs must be used in the firewall or other ways to be exposed externally. `external`: the external LBs are created with given external	mandatory		internal/ external	For Premise Edition - This parameter is configured as `external`	

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	static IPs (IPs for the green and blue LBs must be set with `lbIpBlue` and `lbIpGreen` during the infra-color deployment.					
deployment.coturnService.annotations	Annotation that is added to the Kubernetes LoadBalancer Service object	optional				`deployment:` `coturnService:` `annotations:` `service.beta.kubernetes.i` `azure-load-balancer-resource-group: service-webrtc-westus2-dev`
monitoring.enabled	Enable monitoring content - dashboards, alerts, metrics	optional	false	true/false		
monitoring.dashboards	Enable ConfigMaps deployment that contains dashboards	optional	false	true/false		
monitoring.prometheusMetrics	Enables Prometheus metrics to deploy PodMonitors	optional	false	true/false		
monitoring.prometheusAlerts	Enable Prometheus rules for alerts	optional	false	true/false		
image.imagePullSecrets	Secrets to pull image, list	mandatory				`image:` `imagePullSecrets:`

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
						`- myRegistrySecret`
image.pullPolicy	Kubernetes pull policy of all containers	optional	Always	Always/ IfNotPresent		
image.initContainerImage	Image for initialization container - used to create log folders. If image is not specified, the init container is not applied and the logs are written into `logPath`	optional		string		
image.webrtc	Repository/ directory to get the Gateway image	mandatory		string		`pureengage-docker-staging.jfrog.io/webrtc`
image.coturn	Repository/ directory to get the CoTurn image	mandatory		string		`pureengage-docker-staging.jfrog.io/webrtc`
image.webrtcVersion	Versions of the WebRTC Gateway container	mandatory		string		`9.0.000.88`
image.coturnVersion	Versions of the CoTurn container	mandatory		string		`9.0.000.88`
gateway.replicas	Number of Gateway pods on the deployment stage	optional	1	integer		
gateway.workersCount	Number of Gateway worker threads that handle calls. 1 worker	optional	3	integer		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	handles 25 registrations/ calls. CPU and Memory request depends on the number of workers.					
gateway.voiceSipProxy	Voice microservice - SIP proxy address	mandatory		string, address		`voice-sipproxy.voice.svc.cluste`
gateway.turnExternalUriBlue	FQDNs of CoTurn blue LB	mandatory		string, address		
gateway.turnExternalUriGreen	FQDNs of CoTurn green LB	mandatory		string, address		
gateway.authRedirectUri	GWS/WEE redirect URI for WWE authentication	mandatory		string, address		
gateway.authService	GAuth service address	mandatory		string, address		
gateway.envService	GWS9.x Environment service address	mandatory		string, address		
gateway.cfgService	GWS9.x configuration service address	optional		string, address		
gateway.enableTranscoding	Enable or disable transcoding on the Gateway side. Transcoding is enabled by default. If the transcoding is disabled, the Gateway can handle more agent sessions but OPUS codec is not	optional	true	true/false		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	supported.					
gateway.enable1pccCalls	Specifies if the 1pcc operations are enabled	optional	false	true/false		
gateway.arguments	Any additional options that are applied to the Gateway containers	optional		Array of strings		`gateway: arguments: [ '-codecs pcmu,pcma,opus=120', '-sip-disallowed-codecs opus,telephone-event' ]`
gateway.podAnnotations	Any additional annotations that are applied to the Gateway pods	optional				`gateway: podAnnotations: prometheus.io/scrape: "true" prometheus.io/port: "10052" prometheus.io/path: "/metrics"`
gateway.resources	Describes the resources requested for the Gateway pods. **Important** Do not specify this option, if you do not need resources requests/limits.	optional		Section		`gateway: resources: requests: cpu: 800 memory: 150 limits: memory: "8Gi"`
gateway.resources.requests.cpu	Requested amount of CPU milliunits.	optional	800	integer		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	Important This value is per worker and is multiplied by the `gateway.worke` option in helm					
gateway.resources.requests.memory	Requested amount of Memory (in MB). Important This value is per worker and is multiplied by the `gateway.worke` option in helm	optional	150	integer		
gateway.resources.limits.memory	Absolute value for Gateway memory usage limit	optional	"8Gi"	Kubernetes value for the resource limit		

gateway.scaling	Describes the auto-scaling parameters. If the `deployment.keda` option is set to `false`, you can skip this option.	optional		Section		`gateway:`
						`  scaling:`
						`pollingInterval: 30`
						`maxReplicaCount: 100`
						`prometheusAddress: http://monitoring-prometheus-prometheus.monitoring:909(`
						`thresholdSignins: 70`

gateway.scaling.prometheusAddress	Describes the auto-scaling parameters. If the `deployment.keda` option is set to `false`, you can skip	optional	http://monitoring-prometheus-prometheus.monitoring:909	string, address		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	this option.					
gateway.scaling.pollingInterval	KEDA polling interval (in seconds) - the interval to check each trigger on. See KEDA documentation for more information.	optional	30	integer		
gateway.scaling.maxReplicaCount	Maximum number of replicas that are raised by KEDA/HPA. See KEDA documentation for more information.	optional	100	integer		
gateway.scaling.thresholdSignins	In persons - number of registered agents that causes the Gateway auto-scaling if exceeded	optional	71	integer		

gateway.budget.minAvailable	Option to configure the `PodDisruptionBudget` option. Do not specify this option, ff you do not need the `PodDisruptionBudget` option for the Gateway deployment.	optional		Kubernetes PodDisruptionBudget (PBD) value		`gateway:`
						`  budget:`
						`minAvailable: 50%`

gateway.secrets.type	Describes where the secrets are taken - in Kubernetes secrets, CSI driver, or from the Environment	mandatory		csi k8s env		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	variables					
gateway.secrets.csi.gws	If the `secrets.type` option is set to `csi`, the name of the CSI object contains the GWS secret			string		
gateway.secrets.k8s.gws	If the `secrets.type` option is set to k8s, the name of the Kubernetes Secret object that contains the GWS secret			string		
gateway.secrets.env.gwsClient	If the `secrets.type` option is set to env, the value is GWS clientid created for WebRTC			string		
gateway.secrets.env.gwsSecret	If the `secrets.type` option is set to env, the value is GWS secret for the client given clientid			string		

gateway.securityContext	Security context for the Gateway container	optional		Specification		`gateway:`
						`securityContext:`
						`runAsUser: 500`
						`runAsGroup: 500`

gateway.serviceAccountName	Name of the ServiceAccount that is used to run the Gateway pod	optional		string		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
gateway.logPath	Path to the log-directory. used for both - PVC or HostPath types of logs. Also, check the `esServer` option. If `/mnt/log/webrtc` is specified, the `/mnt/log/webrtc//webrtcgw` logfiles are created and used in the mentioned path. If the `image.initContainerImage` option is not specified, the folder with the pod name will not be created and the `/mnt/log/webrtc/webrtcgw` logfiles will be created. Important If this option is set to `stdout`, the entire WebRTC GW logs are produced to the `stdout` in JSON format.	mandatory	"/mnt/log/webrtc"	string		`"/export/vol1/PAT/infra/webrtc"`

gateway.logPvc	Option for Persistent Volume Claim used for the Gateway logs. If logPvc is not defined, the	optional		Section		`gateway:`
						`  logPvc:`
						
						`pvcName: webrtc-gateway-log-pvc`

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example

	HostPath is used for the logs mount.					`volumeName: webrtc-gateway-log-volume`
						`storageClassName: genesys-webrtc`
						`capacity: 5Gi`
						`volumeSpec:`
						`accessModes: - ReadWriteMany`
						`persistentVolumeReclaimPol Retain`
						`nfs:`
						`path: /export/vol1/PAT/infra/webrtc`
						`server: 192.168.30.51`

gateway.logPvc.pvcName	Name of the Persistent Volume Claim. If this option is present, the PVC is created. Else, the `hostpath` is used for the Gateway logs.	optional		string		
gateway.logPvc.volumeName	PersistentVolume name for the PVC. Single Volume is used for both green and blue deployments of the gateway	optional		string		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
gateway.logPvc.volumeSpec	If the Perisitent Volume specification is configured in the `gateway.logPvc.volumeSpec` option, the PersistentVolume object with name from the `gateway.logPvc.volumeName` option is created using this specification.	optional		Specification		gateway: logPvc: volumeSpec: accessModes: - ReadWriteMany persistentVolumeReclaimPo Retain nfs: path: /export/vol1/PAT/infra/webrtc server: 192.168.30.51
gateway.logPvc.volumeAnnotations	Any additional annotations that are used for the PersistentVolume if the `gateway.logPvc.volumeSpec` is specified here.	optional		Specification		gateway: logPvc: volumeAnnotations: pv.kubernetes.io/bound-by-controller: 'yes'
gateway.esServer	Specifies the destination for the ElasticSearch logging - ElasticSearch server address or stdout. Gateway produces messages in the ElasticSearch format.	optional	stdout	network address or "stdout"		
gateway.restartPolicy	Restart policy for gateway pods.	Optional	Always	depends on cluster		
coturn.port	Coturn port that is used by the	optional	443	integer		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	CoTurn Load Balancer					
coturn.lbIpBlue	External IP for CoTurn blue Load Balancer service. The IP must be same as the one used for the `gateway.turnExternalUriBlue` A-record	mandatory		IP address		
coturn.lbIpGreen	External IP for CoTurn green Load Balancer service. The IP must be same as the one used for the `gateway.turnExternalUriGreen` A-record	mandatory		IP address		
coturn.replicas	Number of CoTurn pods	optional	1	integer		

coturn.podAnnotations	Any additional annotations that are applied for CoTurn pods	optional		Specification		`coturn:`
						`podAnnotations:`
						`pods/realtime: "true"`
						`pods/owner: "1051"`

coturn.resources	Describes resources requested for the CoTurn pods. Do not specify this option if you do not need resources requests/limits.	optional		Section		`coturn:`
						`resources:`
						`requests:`
						`cpu: "0.5"`
						`memory: "768Mi"`
						`limits:`
						`memory: "8Gi"`

coturn.resources.requests.cpu	Requested	optional	0.5	Kubernetes		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	amount of CPU. Coturn requires 0.08CPU per call.			CPU request format		
coturn.resources.requests.memory	Requested amount of Memory	optional	150	Kubernetes memory request format		
coturn.resources.limits.memory	Absolute value for the Coturn memory usage limit	optional	"8Gi"	Kubernetes value for resoure limit		

coturn.scaling	Describes the autoscaling parameters. If the `deployment.keda` option is set to `false`, you can skip this section	optional		Section		`coturn:`
						`  scaling:`
						
						`pollingInterval: 30`
						
						`maxReplicaCount: 100`
						
						`thresholdCpu: 60`
						
						`thresholdMemory: 60`

coturn.scaling.pollingInterval	Specifies the KEDA polling interval in seconds - the interval to check each trigger on. Refer to KEDA documentation for more information.	optional	30	integer		
coturn.scaling.maxReplicaCount	Maxium number of replicas that are raised by KEDA/HPA. Refer to KEDA documentation for more information.	optional	100	integer		
coturn.scaling.thresholdSignins		optional	71	integer		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	percentage					
coturn.scaling.thresholdCpu	In percentage. The target value is the average of the CPU resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods.	optional	60	integer		
coturn.scaling.thresholdMemory	In percentage. The target value is the average of the memory resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods.	optional	60	integer		

coturn.budget.minAvailable	Option to configure `PodDisruptionBudget`. Do not specify this option, if you do not need `PodDisruptionBudget` for the CoTurn deployment.	optional		Kubernetes PDB value		`coturn:`
						`  budget:`
						`    minAvailable: 50%`

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example

coturn.securityContext	Security context for the CoTurn container.	optional		Specification		`coturn:`
						`securityContext:`
						`runAsUser: 500`
						`runAsGroup: 500`

coturn.serviceAccountName	Name of the ServiceAccount to use to run the CoTurn pod.	optional		string		
coturn.logPath	Path to the log-directory. This can be the directory path or "stdout". This path is used for both PVC or HostPath types of logs. Example: If `/mnt/log/webrtc` is specified, `"/mnt/log/webrtc//turn.xxx.log"` logfile is created and used in the mentioned path. If `image.initContainerImage` is not specified, the folder with pod name will not be created and `mnt/log/webrtc/turn.xxx.log` logfile will be created.	mandatory	"/mnt/log/webrtc"	string		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example

coturn.logPvc	Section for Persistent Volume Claim used for CoTurn logs. If this option not defined, the `HostPath` is used for logs mount.	optional	"/mnt/log/webrtc"	Section		`coturn:`
						`  logPvc:`
						`pvcName: webrtc-coturn-log-pvc`
						`storageClassName: default`
						`capacity: 10Gi`
						`volumeName: webrtc-coturn-log-volume`
						`volumeSpec:`
						`      nfs:`
						`server: 192.168.1.5`
						`path: /storage/webrtc`
						`volumeMode: Filesystem`
						`persistentVolumeReclaimPo`
						`Retain`

coturn.logPvc.pvcName	Name of PersistentVolumeClaim. If this option is present, PVC will be created. Else, the `HostPath` is used for CoTurn logs.	optional		string		
coturn.logPvc.storageClassName	StorageClass name for the CoTurn PVC	optional		string		
coturn.logPvc.capacity	Volume capacity	optional		Kubernetes capacity storage values		
coturn.logPvc.volumeName	Persistent Volume name for the	optional		string		

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	PVC. Single Volume is used for both green and blue deployments of the CoTurn logs					

coturn.logPvc.volumeSpec	If the Persistent Volume specification is configured in `coturn.logPvc.volumeSpec`, the Persistent Volume object with name from the `coturn.logPvc.volumeName` will be created using this specification.	optional		Specification		`gateway:`
						`  logPvc:`
						`volumeSpec:`
						`accessModes:`
						`        -`
						`ReadWriteMany`
						`persistentVolumeReclaimPo`
						`Retain`
						`      nfs:`
						`path:`
						`/export/`
						`vol1/PAT/`
						`infra/`
						`webrtc`
						`server:`
						`192.168.30.51`

coturn.logPvc.volumeAnnotations	Any additional annotations that are used for the Persistent Volume, if the `coturn.logPvc.volumeSpec` option is specified	optional		Specification		`gateway:`
						`  logPvc:`
						`volumeAnnotations:`
						`pv.kubernetes.io/`
						`bound-by-`
						`controller:`
						`'yes'`

coturn.restartPolicy	Restart policy for coturn pods.	optional	Always	depends on cluster		
labels.common	Describes the additional labels for common resources	optional				
labels.gateway	Describes the additional	optional				

Option name	Description	Is mandatory	Default value	Valid value	Notes	Example
	labels for the Gateway resources - pods, deployments, and services					
labels.coturn	Describes the additional labels for the CoTurn resources - pods, deployments, and services	optional				
labels.alerts	Describes the additional labels for the alert objects	optional				

Configure Kubernetes

Document the layouts for the following so customers can create them if their Helm chart doesn't include a way to do this:

- *ConfigMaps*
- *Secrets*

Configure security

The security context settings define the privilege and access control settings for pods and containers.

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

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

Arbitrary UIDs in AKS

If you want to use arbitrary UIDs in your Azure Kubernetes Services deployment, override the **securityContext** settings in the **values.yaml** file, so that you do not define any specific IDs.

```
podSecurityContext:
  runAsNonRoot: true
  runAsUser: null
  runAsGroup: 0
  fsGroup: null

securityContext:
  runAsNonRoot: true
  runAsUser: null
  runAsGroup: 0
```

Configure the service

Before proceeding with the deployment process, perform the following pre-steps:

1. **Review values-template.yaml in helm charts**: It provides all the available options with comments and explanations.
2. **Configure all the options in your own values file**: Configure/overwrite values for options that you need. Use the values-template.yaml file from the package that displays the list of available options with their description.

> **Important**
>
> Do not configure **deployment.type** and **deployment.color** options in values.yaml-file(s). These values should be used only during deployment process as command-line parameters to specify the deployment process.

Sample values.yaml file:

```
deployment:
  namespace:        webrtc
  ingress:
    domain: apps.vce-c0.eps.genesys.com
    annotations:
      kubernetes.io/ingress.class:                         nginx01-internal
      nginx.ingress.kubernetes.io/affinity:                cookie
      nginx.ingress.kubernetes.io/affinity-mode:           persistent
      nginx.ingress.kubernetes.io/ssl-redirect:            "false"
      nginx.ingress.kubernetes.io/session-cookie-path:     "/; Secure"
      nginx.ingress.kubernetes.io/session-cookie-samesite: None
  dnsPolicy: ClusterFirst
  dnsConfig:
    options:
```

```
      - name: ndots
        value: "3"
  keda: false
  coturnDeployment: external


monitoring:
  enabled:          false
  dashboards:       false
  prometheusMetrics: false
  prometheusAlerts:  false


image:
  imagePullSecrets:
  - webrtcjfrogsecret
  initContainerImage: pureengage-docker-staging.jfrog.io/alpine:3.7-curl
  webrtc: pureengage-docker-staging.jfrog.io/webrtc
  coturn: pureengage-docker-staging.jfrog.io/webrtc
  webrtcVersion: 9.0.000.88
  coturnVersion: 9.0.000.88


gateway:
  logPath:                 "/export/vol1/PAT/infra/webrtc"
  logPvc:
    pvcName:               webrtc-gateway-log-pvc
    volumeName:            webrtc-gateway-log-volume
    storageClassName:      genesys-webrtc
    capacity:              5Gi
    volumespec:
      accessModes:
        - ReadWriteMany
      persistentVolumeReclaimPolicy: Retain
      nfs:
        path: /export/vol1/PAT/infra/webrtc
        server: 192.168.30.51
  esServer:                stdout
  replicas:                 1
  workersCount:            1
  voiceSipProxy:          voice-sipproxy.voice.svc.cluster.local:5080;transport=tcp
  turnExternalUriBlue:   192.168.30.208
  turnExternalUriGreen:  192.168.30.209
  authRedirectUri:       http://gauth.apps.vce-c0.eps.genesys.com:80
  authService:           http://gauth-auth.gauth.svc.cluster.local:80
  envService:            https://gws.apps.vce-c0.eps.genesys.com
  resources:
    requests:
      # NB! 800m per worker, MUST be integer, not string - will be multiplied by
workersCount in helm
      cpu:  800
      # NB! 150Mi per worker, MUST be integer, not string - will be multiplied by
workersCount in helm
      memory: 150
    limits:
      memory: "8Gi"
  secrets:
    type: env
    env:
     gwsClient: external_api_client
     gwsSecret: secret
  securityContext:
    runAsUser: 500
```

```
    runAsGroup: 500


coturn:
  logPath:                  "/export/vol1/PAT/infra/coturn/"
  logPvc:
    pvcName:                webrtc-coturn-log-pvc
    volumeName:             webrtc-coturn-log-volume
    storageClassName:        genesys-webrtc
    capacity:                5Gi
    volumeSpec:
      accessModes:
        - ReadWriteMany
      persistentVolumeReclaimPolicy: Retain
      nfs:
        path: /export/vol1/PAT/infra/webrtc
        server: 192.168.30.51
  replicas:    1
  port:        443
  lbIpBlue:  192.168.30.208
  lbIpGreen: 192.168.30.209
  securityContext:
    runAsUser: 500
    runAsGroup: 500
```

3. **PersistentVolume (PV) and PersistentVolumeClaim (PVC)**: If you plan to use PV for logs, create the PV and then specify it for PVC of Gateway and CoTurn.

   PV can also be created during the common-infrastructure deployment. You should review the `values-template.yaml` file and then configure the PV specification for Gateway and CoTurn.

   Single PV/PVC pair will be used for both Green and Blue deployments of Gateway, and another single PV/PVC pair will be used for both Green and Blue deployments of CoTurn.

Configure Webphone

Contents

Learn how to configure Webphone.

Related documentation:

-
-
-

RSS:

- For private edition

Override Helm chart values

Download the Webphone Helm charts from JFrog using your credentials. Override the configuration parameters in the **values.yaml** file to provide deployment-specific values for certain parameters. You can override values in the Helm charts to configure Private Edition. For more information about overriding Helm chart values, see the "suite-level" documentation about how to override Helm chart values: Overriding Helm chart values.
The following table includes the Helm chart values required for configuring Webphone service.

Option Name	Description	Is Mandatory?	Default Value	Valid Value
image.repository	Webphone image repository/ directory.	mandatory		string
image.tag	Version of Webphone container.	mandatory		string
image.pullPolicy	Kubernetes pull policy of all containers.	mandatory		string
image.imagePullSecrets	Secrets to pull image.	optional		list
replicaCount	Number of desired Webphone pods.	optional		integer
webphone.cfgService	GWS9.x environment service address for configuring Webphone service.	mandatory		string
webphone.authService	GAuth service address.	mandatory		string
webphone.webrtcService	WebRTC service address.	mandatory		string

Option Name	Description	Is Mandatory?	Default Value	Valid Value
webphone.webphoneService	Webphone service address.	mandatory		string
webphone.authRedirectUri	GWS/WEE redirect URI for WWE authentication.	mandatory		
secrets.gwsClientId	GWS client ID injected as environment variable.	mandatory	"webrtc_secret_client"	
k8s.gws	Name of Kubernetes Secret object that contains secret for GWS client ID.	optional		
k8s.gwsSecretPath	File path in Kubernetes Secret object that contains the GWS secret.	optional		
secrets.env.gws	Secret for GWS client ID, injected as environment variable.	optional		string
service.enabled	To enable deployment of Kubernetes Service for Webphone pods.	mandatory	true	true/false
service.type	Type of Webphone Service object.	optional	ClusterIP	string
service.port	Port of Webphone Service object.	optional	80	integer
ingress.enabled	To enable deployment of Kubernetes Ingress for Webphone Ingress. This parameter can be enabled only if Webphone service is enabled.	mandatory	true	true/false

ingress.annotations	Any additional annotations applied for Webphone Ingress.	optional	`ingress:`
			`  annotations:`
			
			`nginx.ingress.kubernetes.io/affinity: cookie`
			
			`nginx.ingress.kubernetes.io/affinity-mode: persistent`	

Option Name	Description	Is Mandatory?	Default Value	Valid Value

			`nginx.ingress.kubernetes.io/session-cookie-name: webphonesession`
			
			`nginx.ingress.kubernetes.io/session-cookie-path: /; Secure`
			
			`nginx.ingress.kubernetes.io/session-cookie-secure: "true"`
			
			`nginx.ingress.kubernetes.io/session-cookie-samesite: None`	

ingress.host	FQDNs of Webphone. This field is mandatory, if ingress is enabled.	optional		string
ingress.ingressClassName	Classname of ingress.	optional		string

resources	Resources requested for Webphone pods.	optional	`resources:`
			`  limits:`
			`    cpu: 500m`
			`    memory: 1024Mi`
			`  requests:`
			`    cpu: 50m`
			`    memory: 128M`
				list

podAnnotations	Any additional annotations applied for Webphone pods.	optional		array
podSecurityContext	Security context for Webphone pod.	optional		array
securityContext	Security context for Webphone container.	optional		array
nodeSelector	Node selector for Webphone pods.	optional		array
tolerations	If toleration exists, this parameter is inserted into toleration of Webphone pods.	optional		array
affinity	Pod affinity descriptions.			

Option Name	Description	Is Mandatory?	Default Value	Valid Value
serviceAccountName	Name of the Service Account used to run the Webphone pods.	optional	default	
priorityClassName	Name of the priority class for pods that indicates the importance of a pod relative to other pods.	optional		string
dnsPolicy	Kubernetes DNS Policy applied in the pods.	optional		
dnsConfig	All DNS settings are provided using this field in the Pod Spec.	optional		
autoscaling.enabled	To enable autoscaling of Webphone.	mandatory	false	true/false
autoscaling.minReplicas	Minimal number of Webphone pods.	optional	1	integer
autoscaling.maxReplicas	Maximum number of Webphone pods.	optional	2	integer
autoscaling.targetCPUUtilizationPercentage	CPU usage threshold to trigger autoscaling.	optional	80	integer
autoscaling.targetMemoryUtilizationPercentage	Memory usage threshold to trigger autoscaling.	optional		integer
monitoring.enabled	To enable monitoring of Webphone service.	mandatory	false	true/false
monitoring.scrapeInterval		optional		

Configure the service

Before proceeding with the deployment process, perform the following pre-steps:

1. **Review values-template.yaml in helm charts**: It provides all the available options with comments and explanations.
2. **Configure all the options in your own values file**: Configure/overwrite values for options that you need. Use the values **template.yaml** file from the package that displays the list of available options

with their description.

Sample values.yaml file:

```
image:
  repository: pureengageuse1-docker-multicloud.jfrog.io
  tag:        100.0.007.0000
  pullPolicy: IfNotPresent

webphone:
  cfgService:      "https://gws."
  authService:     "https://gauth."
  webrtcService:   "https://webrtc."
  webphoneService: "https://webphone."

secrets:
  gwsClientId: "webrtc_service_client"
  env:
    gws: "secret"

ingress:
  annotations:
    kubernetes.io/ingress.class: nginx
    kubernetes.io/ingress.allow-http: "true"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/affinity: cookie
    nginx.ingress.kubernetes.io/affinity-mode: persistent
    nginx.ingress.kubernetes.io/session-cookie-name: webphonesession
    nginx.ingress.kubernetes.io/session-cookie-path: /; Secure
    nginx.ingress.kubernetes.io/session-cookie-secure: "true"
    nginx.ingress.kubernetes.io/session-cookie-samesite: None
    nginx.ingress.kubernetes.io/session-cookie-conditional-samesite-none: "true"
    nginx.ingress.kubernetes.io/session-cookie-max-age: "86400"
    nginx.ingress.kubernetes.io/session-cookie-change-on-failure: "true"
  hosts:
    - host: webphone.apps.qrtph6qa.westus2.aroapp.io
      paths:
        - path: "/"

monitoring:
  enabled: false

serviceAccount:
  create: false
  name: default

podSecurityContext:
  runAsGroup: 0
  runAsNonRoot: true

rbac:
  create: false
```

Provision WebRTC

Contents

- Administrator

WebRTC does not require tenant provisioning.

Related documentation:

-
-
-

RSS:

- For private edition

List any provisioning needed to deploy, run, or manage the service. For example:

- *Designer: Create an Access Group specific to Designer Developer, Admin.*
- *Agent Setup: Create Agent Setup options to provide access to Administrator, Supervisor. or Ops.*
- *Genesys Info Mart: Update the CTL_CONFIG table in the GIM DB to control ETL and DB maintenance behavior.*

Tenant provisioning

Describe how to provision the tenant service for .

Deploy

Contents

Learn how to deploy WebRTC Media Service (WebRTC) into a private edition environment.

Related documentation:

-
-
-

RSS:

- For private edition

Assumptions

- The instructions on this page assume you are deploying the service in a service-specific namespace, named in accordance with the requirements on Creating namespaces. If you are using a single namespace for all private edition services, replace the namespace element in the commands on this page with the name of your single namespace or project.
- Similarly, the configuration and environment setup instructions assume you need to create namespace-specific (in other words, service-specific) secrets. If you are using a single namespace for all private edition services, you might not need to create separate secrets for each service, depending on your credentials management requirements. However, if you do create service-specific secrets in a single namespace, be sure to avoid naming conflicts.

> **Important**
> Make sure to review Before you begin for the full list of prerequisites required to deploy WebRTC.

WebRTC uses blue-green model of deployment. It has the following main deployment principles:

- Both components - WebRTC Gateway and CoTurn Server - are deployed for each color and switched together
- Blue WebRTC Gateway is always configured to work with Blue CoTurn and green WebRTC Gateway is always configured to work with green CoTurn
- WebRTC have two FQDNs to reach active and inactive deployments:
    - **webrtc.domain.com** - active deployment. For example: webrtc.genesyshtcc.com
    - **webrtc-test.domain.com** - inactive deployment for tests. For example: webrtc-test.genesyshtcc.com

Deploy

You can deploy WebRTC using:

- Internal CoTurn Load Balancer or
- External CoTurn Load Balancer

Deploying WebRTC using internal CoTurn Load Balancer

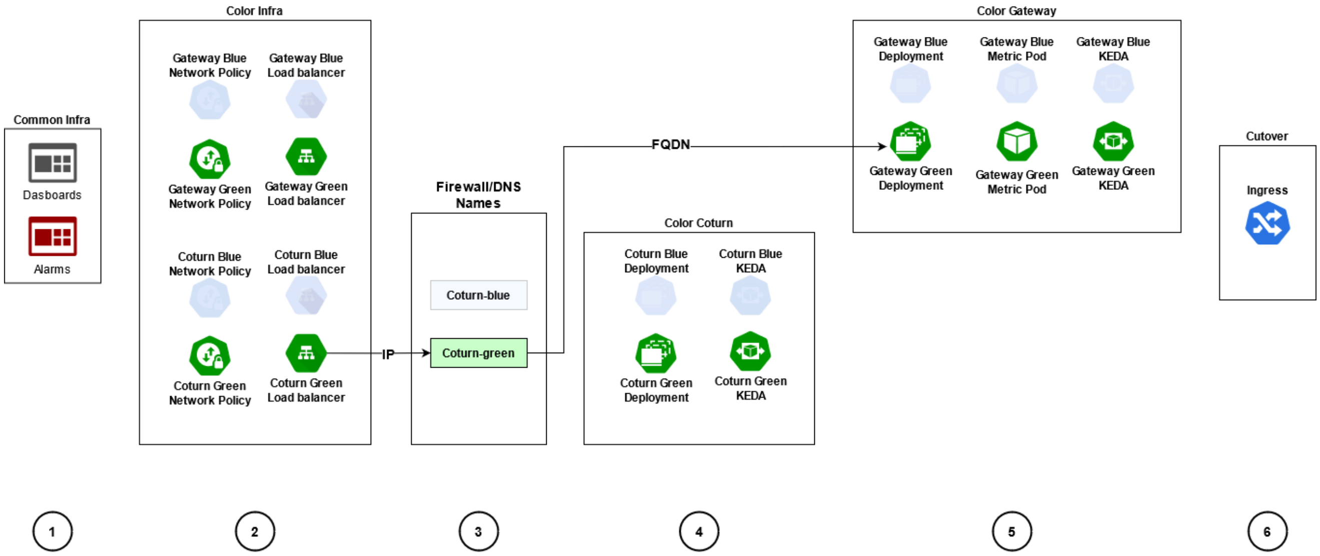
Initial deployment and Upgrade use the same sequence:

1. Deploy/upgrade inactive color of deployment
2. Make the cutover

You need to deploy the Color Infra package with CoTurn Load Balancer to get the IP address assigned automatically for the CoTurn Load Balancer by the infrastructure. Then, the infrastructure team should assign the IP to the CoTurn Load Balancer, create the FQDN for the IP and ensure that the IP is set in the firewall and is available from outside the cluster.

> **Important**
> The IP address assigned to the CoTurn Load Balancer must be external and available outside the cluster. Else, the media will not get through the WebRTC.

The following image shows the steps involved in deploying WebRTC using the internal CoTurn Load Balancer:



Follow the below steps to deploy WebRTC using internal CoTurn Load Balancer:

1. **Create common infrastructure elements such as dashboards and alarms**: This step deploys dashboards, alarms, and other common infrastructure elements.

   > **Important**
   > You should perform this step even if you do not require the dashboard and alarms.

   Run the following command to create the common infrastructure elements:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --set-string deployment.color="" webrtc-infra {HelmRepoPath}/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-string deployment.color="" webrtc-infra wrtchelmrepodevwestus2/webrtc-service --version=0.1.93 -n webrtc
   ```

2. **Create infrastructure elements for the deployment color**: This step deploys the infrastructure objects such as Turn Load Balancer, Gateway Service Object, Gateway Network Policies, and Turn Network Policies for the given color of deployment.
   You should also specify the INACTIVE color of deployment in this step.

   > **Important**
   > You should configure the `deployment.coturnDeployment` option with the value `internal` in your values.yaml file.

   Run the following command to deploy the infrastructure objects:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --set-string deployment.color={INACTIVE_COLOR} webrtc-infra-{INACTIVE_COLOR} {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-string deployment.color=blue webrtc-infra-blue wrtchelmrepodevwestus2/webrtc-service --version=0.1.93 -n webrtc
   ```

3. **Get the IPs from the CoTurn Load Balancers, create DNS records and firewall rules**: This step gets the IP address from the Colurn Load Balancer created in Step 2. The name of LoadBalancer will be similar to: webrtc-coturn-service-{COLOR}.
   Create appropriate FQDN for this IP address in your DNS. This FQDN will be used by the WebRTC agents from outside the cluster to establish the RTP stream. Though you can use the IP address as it is, it is not the best practice to do so.

4. **Create CoTurn elements for the deployment color**: This step is to Upgrade/Deploy CoTurn for INACTIVE color.
   Run the following command to upgrade/deploy the INACTIVE color of deployment:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=coturn --
set-string deployment.color={INACTIVE_COLOR} webrtc-coturn-{INACTIVE_COLOR}
{HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=coturn --set-
string deployment.color=blue webrtc-coturn-blue wrtchelmrepodevwestus2/webrtc-
service --version=0.1.93 -n webrtc
```

5. **Create Gateway elements for deployment color using the information from Step 3**: This step is to Upgrade/Deploy Gateway for INACTIVE color. You shoud also specify the extenal FQDN of the CoTurn LoadBalancer in this step using the `gateway.turnExternalUriBlue` or `gateway.turnExternalUriGreen` options.
   Run the following command:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=gateway
--set-string deployment.color={INACTIVE_COLOR} --set-string
gateway.turnExternalUri{INACTIVE_COLOR}={COTURN FQDN INACTIVE_COLOR} webrtc-gateway-
{INACTIVE_COLOR} {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
```

   Example for Blue deployment:

```
helm upgrade --install -f ./k8s/values.yaml--set-string deployment.type=gateway --set-
string deployment.color=blue --set-string gateway.turnExternalUriBlue=turn-
blue.ext.mydoamin.com  webrtc-gateway-blue wrtchelmrepodevwestus2/webrtc-service --
version=0.1.93 -n webrtc
```

   Or, you can specify the IP of the Blue CoTurn Load Balancer

```
helm upgrade --install -f ./k8s/values.yaml--set-string deployment.type=gateway --set-
string deployment.color=blue --set-string gateway.turnExternalUriBlue=12.106.34.55
webrtc-gateway-blue wrtchelmrepodevwestus2/webrtc-service --version=0.1.93 -n webrtc
```

6. **Create/update Ingress controller rules for Active/Inactive routing for Gateway deployments**: This step is to Install/upgrade ingress without changing the active color. The same step is used for the Cutover.

> ## Important
> If you are deploying/upgrading `green`, specify the current ACTIVE color of deployment in the `deployment.color` option. Then specify `blue` and vice versa. If you deploying/upgrading `green` and specify `green` for the **cutover** step, the current active deployment will be switched to the just deployed/upgraded `green`.

   You must perform this step even if you are not planning to make the cutover right now. This step is to upgrade the ingress and environment.

   Run the following command to create/upgrade Ingress controller rules:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=cutover
--set-string deployment.color={ACTIVE_COLOR} webrtc-ingress {HelmRepoPath }/webrtc-
service --version={WebRTC Charts Version}
```

   Example:

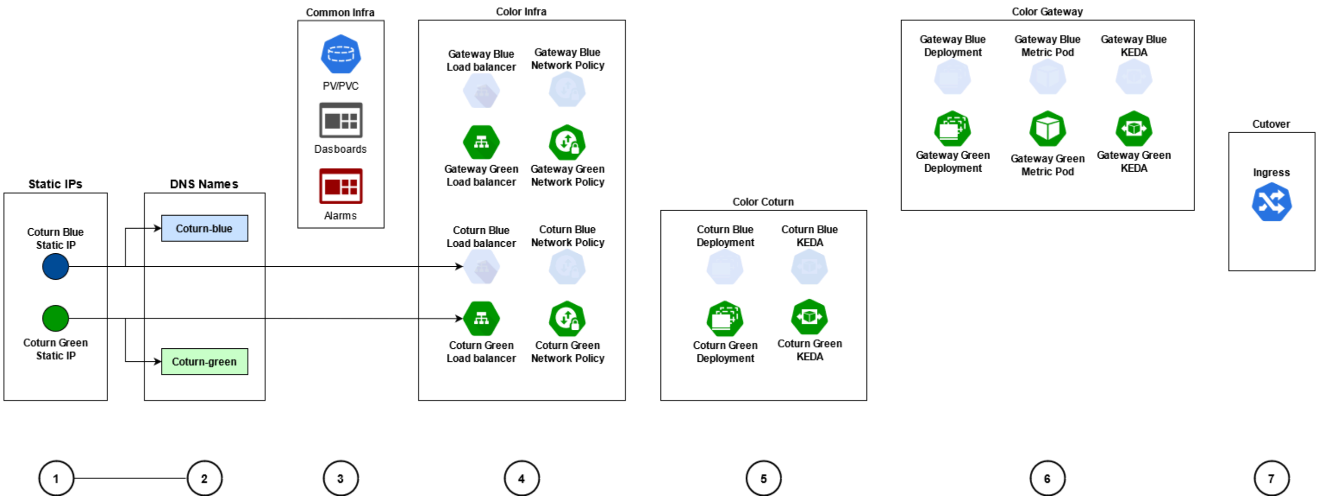
```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=cutover --
set-string deployment.color=green webrtc-ingress wrtchelmrepodevwestus2/webrtc-
service --version=0.1.93 -n webrtc
```

Deployment with external CoTurn Load Balancer

Initial deployment and Upgrade use the same sequence:

1. Deploy/upgrade inactive color of deployment
2. Make the cutover

The following image shows the steps involved in deploying WebRTC using the external CoTurn Load Balancer:



Follow the below steps to deploy WebRTC with external CoTurn Load Balancer

1. **Create static IPs for CoTurn**: This step is to specify the pre-created public IP for CoTurn Green in the `coturn.lbIpGreen` option and public IP for CoTurn Blue in the `coturn.lbIpBlue` option.
2. **Create DNS records for the created IPs**: This step is to specify the public FQDNs for CoTurn. Specify the pre-created public FQDN for CoTurn Green in the `gateway.turnExternalUriGreen` option and public FQDN for CoTurn Blue in the `gateway.turnExternalUriBlue` option.
3. **Create common infrastructure elements**: This step will deploy Persistent Volumes, Persistent Volume Claims, dashboards, alarms, and other common infrastructure elements.

> **Important**
>
> You need to run this step even if you are not using the dashboard and alarms.

Run the following command to create the infrastructure elements:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --
set-string deployment.color="" webrtc-infra {HelmRepoPath}/webrtc-service --
version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-
```

```
string deployment.color="" webrtc-infra wrtchelmrepodevwestus2/webrtc-service --
version=0.1.93 -n webrtc
```

4. **Create infrastructure elements for deployment color**: This step is to deploy the infrastructure objects such as Turn Load Balancer, Gateway Service Object, Gateway Network Policies, and Turn Network Policies for the given color of deployment.
   You must specify INACTIVE color of deployment for this step.

   > **Important**
   >
   > Configure the `deployment.coturnDeployment` option with the value `external` in your `values.yaml` file.

   Run the following command to create the infrastructure elements:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=infra --
   set-string deployment.color={INACTIVE_COLOR} webrtc-infra-{INACTIVE_COLOR}
   {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=infra --set-
   string deployment.color=blue webrtc-infra-blue wrtchelmrepodevwestus2/webrtc-service
   --version=0.1.93 -n webrtc
   ```

5. **Create CoTurn elements for deployment color**: This step is to upgrade/deploy CoTurn for inactive color.
   Run the following command to specify the INACTIVE color of deployment:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=coturn --
   set-string deployment.color={INACTIVE_COLOR} webrtc-coturn-{INACTIVE_COLOR}
   {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

   ```
   helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=coturn --set-
   string deployment.color=blue webrtc-coturn-blue wrtchelmrepodevwestus2/webrtc-
   service --version=0.1.93 -n webrtc
   ```

6. **Create Gateway elements for deployment color**: This step is to upgrade/deploy the Gateway for inactive color.

   > **Important**
   >
   > CoTurn DNS name is used for Gateway deployment as a parameter in the corresponding `values.yaml` file.

   Run the following command to specify the INACTIVE color of deployment:

   ```
   helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=gateway
   --set-string deployment.color={INACTIVE_COLOR} webrtc-gateway-{INACTIVE_COLOR}
   {HelmRepoPath }/webrtc-service --version={WebRTC Charts Version}
   ```

   Example:

```
helm upgrade --install -f ./k8s/values.yaml--set-string deployment.type=gateway --set-
string deployment.color=blue webrtc-gateway-blue wrtchelmrepodevwestus2/webrtc-
service --version=0.1.93 -n webrtc
```

7. **Create/update Ingress controller rules for Active/Inactive routing for the Gateway deployments**: This step is to install/upgrade ingress without changing the active color. The sampe step is also used for the Cutover.

> **Important**
>
> If you are deploying/upgrading `green`, specify the current ACTIVE color of deployment in the `deployment.color` option which is `blue` and vice versa. If you deploying/upgrading `green` and specify `green` for the **cutover** step, the current active deployment will be switched to the just deployed/upgraded `green`.

> **Important**
>
> You must perform this step even if you do not plan to make cutover right now. This step is to upgrade the ingress and environment.

Run the following command to create/upgrade Ingress controller rules:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=cutover
--set-string deployment.color={ACTIVE_COLOR} webrtc-ingress {HelmRepoPath }/webrtc-
service --version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=cutover --
set-string deployment.color=green webrtc-ingress wrtchelmrepodevwestus2/webrtc-
service --version=0.1.93 -n webrtc
```

Cutover

During cutover, it switches active color of deployment. This step should be performed only after you confirm that the newly installed/upgraded deployment is alive and functional. You must specify the current INACTIVE color of deployment in the `deployment.color` option - deployment that was just deployed/upgraded and tested. Run the following command to specify the cutover:

```
helm upgrade --install -f {Webrtc Values files} --set-string deployment.type=cutover --set-
string deployment.color={INACTIVE_COLOR} webrtc-ingress {HelmRepoPath }/webrtc-service --
version={WebRTC Charts Version}
```

Example:

```
helm upgrade --install -f ./k8s/values.yaml --set-string deployment.type=cutover --set-string
deployment.color=blue webrtc-ingress wrtchelmrepodevwestus2/webrtc-service --version=0.1.93
-n webrtc
```

Important

You need to use PersistentVolume and PersistentVolumeClaim instead of HostPath logs of Gateway pods and CoTurn Pods.

Validate the deployment

Follow the given steps to validate the deployment.

1. **Verify PVCs are created and bound**

   ```
   kubectl get pvc
   ```

   Sample output:

   ```
   NAME                     STATUS   VOLUME                     CAPACITY   ACCESS
    MODES   STORAGECLASS     AGE
    webrtc-coturn-log-pvc    Bound    webrtc-coturn-log-volume    5Gi
    RWX            genesys-webrtc   110s
    webrtc-gateway-log-pvc   Bound    webrtc-gateway-log-volume   5Gi
    RWX            genesys-webrtc   110s
   ```

2. **Validate CoTurn and Gateway services**

   ```
   kubectl get svc
   ```

   Sample output:

   ```
   NAME                          TYPE           CLUSTER-IP      EXTERNAL-IP
    PORT(S)           AGE
    webrtc-coturn-service-blue    LoadBalancer   10.202.51.156   192.168.30.208
    443:31457/TCP     67m
    webrtc-gateway-service-blue   ClusterIP      10.202.47.170             80/TCP,8080/
    TCP   67m
   ```

3. **Query pods in the WebRTC namespace to confirm that pod is created, and in running status**

   ```
   kubectl get pods
   ```

   Sample output:

   ```
   NAME                                   READY   STATUS    RESTARTS   AGE
    webrtc-coturn-blue-b5db74c96-mh9jv    1/1     Running   0          4m20s
    webrtc-gateway-blue-d7ff45677-vbdg9   1/1     Running   0          86s
   ```

4. **Validate Ingress configuration**

   ```
   kubectl get ingress
   ```

   Sample output:

   ```
   NAME                 CLASS
    HOSTS
    ADDRESS   PORTS   AGE
    webrtc-ingress-int      webrtc.apps.vce-c0.eps.genesys.com,webrtc-test.apps.vce-
   ```

```
c0.eps.genesys.com             80      68s
```

5. Validate Ingress Edge route configuration

    ```
    kubectl get route
    ```

    Sample output:

    ```
    NAME                          HOST/PORT                                     PATH
     SERVICES                      PORT   TERMINATION   WILDCARD
     webrtc-gateway-service-blue   webrtc.apps.qrtph6qa.westus2.aroapp.io
     webrtc-gateway-service-blue   web    edge          None
     webrtc-ingress-int-cvdtt      webrtc.apps.qrtph6qa.westus2.aroapp.io        /
     webrtc-gateway-service-blue   web                  None
     webrtc-ingress-int-trcvh      webrtc.apps.qrtph6qa.westus2.aroapp.io        /blue
     webrtc-gateway-service-blue   web                  None
     webrtc-ingress-int-wf6x9      webrtc-test.apps.qrtph6qa.westus2.aroapp.io   /blue
     webrtc-gateway-service-blue   web                  None
    ```

6. **Query Ingress for** made available **WebRTC Web API**

    ```
    kubectl get ingress
    ```

    Copy the WebRTC API from the Ingress output:

    Sample output:

    ```
    NAME                 CLASS
     HOSTS
     ADDRESS   PORTS   AGE
     webrtc-ingress-int      webrtc.apps.vce-c0.eps.genesys.com,webrtc-test.apps.vce-
     c0.eps.genesys.com             80      3h26m
    ```

    Curl WebRTC "ping" API:

    ```
    curl -s webrtc.apps.vce-c0.eps.genesys.com/ping
     {"state":"up","version":"9.0.000.89","path":"blue"}
    ```

Deploy Webphone

Contents

Learn how to deploy Webphone.

Related documentation:

-
-
-

RSS:

- For private edition

Prerequisites

You must deploy the following services before deploying Webphone:

1. GWS
2. GAuth
3. GVP
4. Voice Microservice
5. Tenant Microservice
6. WebRTC

GWS Secret for Webphone must contain the following grants:

```
grant_type=authorization_code
grant_type=urn:ietf:params:oauth:grant-type:token-exchange
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer
grant_type=client_credentials
```

Deploy Webphone

Run the following command to deploy Webphone with single run of Helm.

```
helm upgrade --install -f override_values webphone-service wrtchelmrepo/webphone --
version=100.0.007+0003 -n webrtc
```

Upgrade, rollback, or uninstall WebRTC

Contents

Learn how to upgrade, rollback or uninstall WebRTC.

Related documentation:

-
-
-

RSS:

- For private edition

Upgrade WebRTC

Follow the same process explained in the Deploy section for the upgrade.

Rollback WebRTC

WebRTC uses green-blue upgrade model. This means that the upgrade is performed for the currently inactive color of deployment, and then the cutover is performed. After the cutover, the new version will be active and the previous version gets inactive. To perform a rollback to the previous version, make a cutover. This makes the previous version active.

Uninstall WebRTC

This section describes the steps involved in uninstalling WebRTC.

Uninstall Ingress

1. Run the following helm command to remove the ingress configuration:

   ```
   helm delete webrtc-ingress -n webrtc
   ```

   Validate

   Run the following command to validate if the ingress configuration is removed.

   ```
   kubectl get ingress -n webrtc
   ```

2. Uninstall Gateway deployment
   Run the following helm command to remove the Gateway deployment:

```
helm delete webrtc-gateway-{color} -n webrtc
```

> **Important**
> Update `color` for blue/green requirements.

Validate

Run the following command to validate if the Gateway deployment is removed:

```
kubectl get pods -n webrtc
```

3. Uninstall CoTurn deployment
   Run the following helm command to remove the CoTurn deployment:

   ```
   helm delete webrtc-coturn-{color} -n webrtc
   ```

   Validate

   Run the following command to validate if the CoTurn deployment is removed:

   ```
   kubectl get pods -n webrtc
   ```

4. Uninstall CoTurn and Gateway services
   Run the following helm command to uninstall CoTurn and Gateway services:

   ```
   helm delete webrtc-infra-{color} -n webrtc
   ```

   Validate

   Run the following command to validate if the CoTurn and Gateway services are removed:

   ```
   kubectl get pods -n webrtc
   ```

5. Uninstall Persistent Volumes (PV) and Persistent Volume Claims (PVC)
   Run the following helm command to uninstall Persistent Volumes and Persistent Volume Claims:

   ```
   helm delete webrtc-infra -n webrtc
   ```

   > **Important**
   > This is optional if you want to keep PVs and PVCs.

   Validate

   Run the following command to validate if the PVs and PVCs are removed:

   ```
   kubectl get pv -l service=webrtcoc get pvc -n webrtc
   ```

Configure WebRTC Agents with Genesys Softphone

Contents

Configure WebRTC agents with Genesys Softphone in Genesys cloud using Agent Setup.

Related documentation:

-
-
-

RSS:

- For private edition

Pre-requisites

- Deploy WebRTC in the Tenant’s region. For more information, see Configure and Deploy WebRTC.
- Install Genesys Softphone in the Agent's place. For more information, see Deploying Genesys Softphone.
- Install WWE 9.0+ version, which is a part of GWS.
- Install GWS 9.0+ version in the tenant's region.

Genesys Softphone with WWE works in a Connector mode. In the Connector mode, the configuration is retrieved from the Configuration Server by WWE and sent to the Genesys Softphone.

To configure the Genesys Softphone in WebRTC mode, see Genesys Softphone in WebRTC mode.

Configure WebRTC agent with Genesys Softphone using Agent Setup

To configure WebRTC agent with Genesys Softphone using Agent Setup:

1. Log in to the **Agent Setup** for the corresponding tenant.
2. Select **Users**.
3. Click **Add User**.
4. Update all the required fields and configure the phone number.
5. Select the **Softphone** check box and **Genesys SoftPhone with WebRTC** in the **SIP Phone Type** drop-down to assign all the mandatory DN level options.
6. Click **Save**.

7. Go to **Users** > **Annex** section and click **Add Section**.
8. In the **Group name** text box, type interaction-workspace and click **+** to configure the options. Use the following table to fill the options and its values.

Option name	Option value for WWE9
privilege.sipendpoint.can-use	true
privilege.webrtc.can-use	false
privilege.voice.can-use	true
sipendpoint.transport-protocol	https
sipendpoint.uri	https://localhost:8000
sipendpoint.enable-webrtc-auth	true
sipendpoint.codecs.enabled.audio	opus
sipendpoint.enable-webrtc-signin-with-switchname	true
sipendpoint.proxies.proxy0.http_proxy	{http_proxy_fqdn:http_proxy_port} Example: sipendpoint.proxies.proxy0.http_proxy = proxy.company.com:8080
webrtc.server-urn	webrtc.service-urn = webrtc.service-urn can be templated using “expression.gws-url.capturing-groups” option. This option can be configured on CloudCluster application level and/or Agent Group level. Hence, the customer need not configure it for each agent.

9. Click **Save**.

10. Go to **Desktop Options** > **Genesys Softphone** section.

11. Select the following check boxes:

- **Usage of Genesys Softphone**
- **Uri** with https://localhost:8000

12. Configure other options if needed.

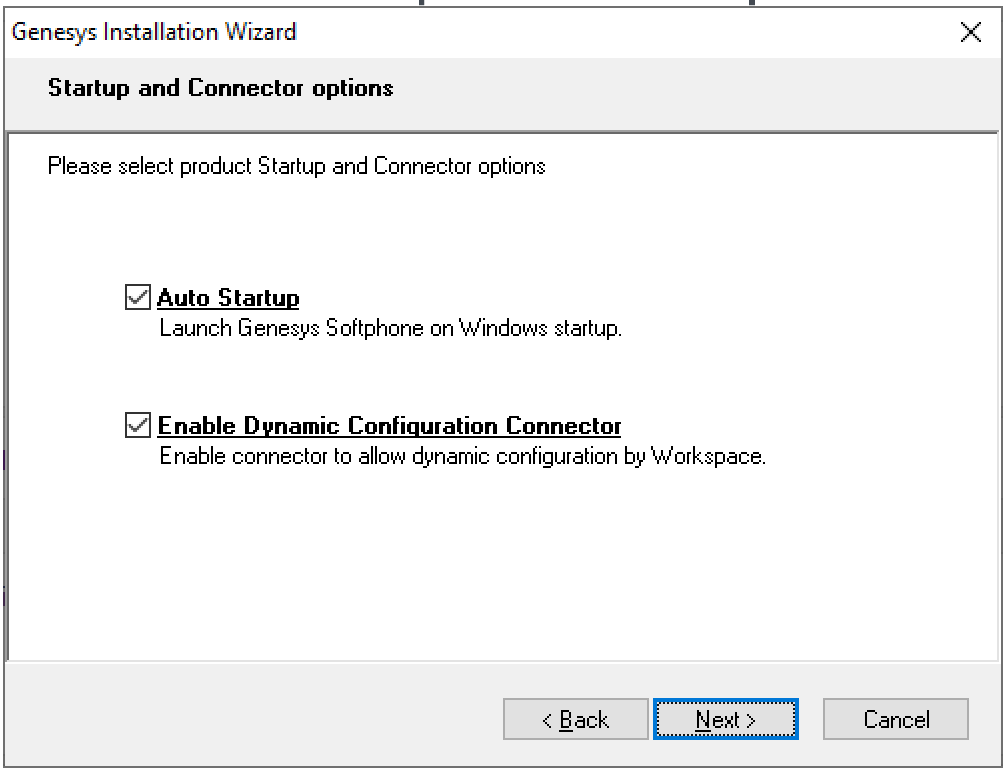
13. Click **Save**.

Configure Genesys Softphone in WebRTC mode

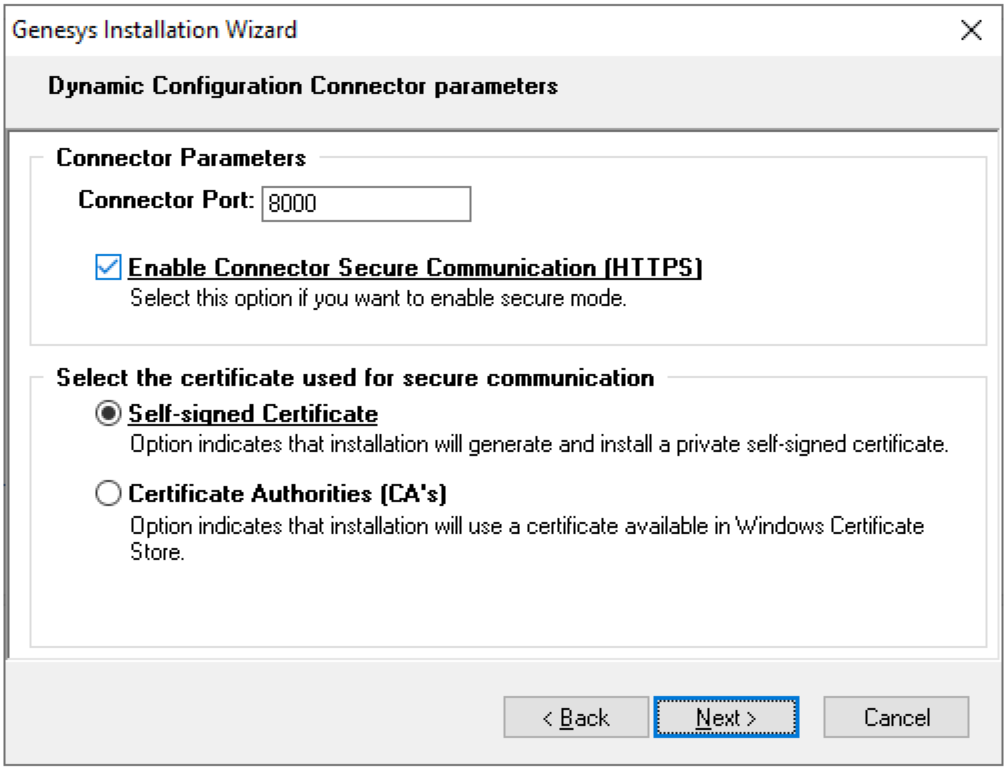
This section includes information on how to configure Genesys Softphone in WebRTC Connector mode.

Enabling Dynamic Configuration Connector in Connector Mode

1. Select **Auto Startup** and **Enable Dynamic Configuration Connector** checkboxes in the **Startup and Connector options** window.



2. Provide a proper connector port and select the **Enable Connector Secure Communication (HTTPS)** checkbox.



3. Select the **Self-signed Certificate** check box if you do not have appropriate certificate available in the Windows Certificate Store. Else, select the **Certificate Authorities (CA's)** check box. Proceed with the regular installation process.

Configure WebRTC agent with browser-based WWE via Agent Setup Application

Contents

Agent does not need a Genesys Softphone as an endpoint for this configuration. But only needs a browser-based WWE, which uses WebRTC capabilities of Chrome, Firefox, and Chromium browsers.

Related documentation:

-
-
-

RSS:

- For private edition

Supported browser versions

- Chrome - 70+
- Firefox - 62+
- Chromium - 85.0.564.41+

Configuration steps

To configure the WebRTC agent, perform the following steps.

1. Log in to the Agent Setup application via Genesys Portal for your Tenant. See Agent Setup guide on how to use Agent Setup application.
2. Select **Users** tab.
3. Click **Add User**.
4. Update all the mandatory fields:
    - First Name
    - Last Name
    - Username
    - Password
    - Password Confirm
    - Phone Number
5. Click **Save**.
6. Go to **Desktop Options** > **Voice**.

7. Select the **Can Use WebRTC** option.
8. Configure **Expression to capture groups in GWS url** and **WebRTC Service URN** options, if required.
9. Click **Save.**

The following table includes the options and its values for WebRTC configuration.

Option Name	Option Value
Expression to capture groups in GWS url	It is a regular expression that allows the workspace to extract some part of its URL to capture the groups containing shared information among services, like the tenant or the region.
WebRTC Service URN	It is a WebRTC service URL that can be templated with the groups captured using the **Expression to capture groups in GWS url** option.

Configure WebRTC Agents with Webphone

Contents

Webphone is a Browser-based standalone WebRTC softphone that can be used as the Agent's device separately from WWE wherever VDI (Citrix, VMware) is used.

Webphone is similar to Zero footprint that provides the ability to separate voice and signaling traffic.

Related documentation:

-
-
-

RSS:

- For private edition

Configure WebRTC agent with Webphone using Agent Setup

To configure WebRTC agents with Webphone, perform the following configurations.

Provision Webphone

The following steps include the configuration for provisioning a Webphone,

1. Log in to **Agent Setup**.
2. Go to **Users** > **Add Users** to create an agent. To know more about user creation, see Manage agents and other users.
3. Enter the mandatory fields to create the user.
4. Navigate to the configuration of the new agent.
5. Enter the agent's phone number and select the Softphone type as **Genesys Softphone / Genesys 420HT / AudioCodes 4xxHD / Polycom**.
6. Go to **User** > **Access Group.**
7. Configure the Access Group for the Agent. To know more about configuring access groups, see Access Groups.

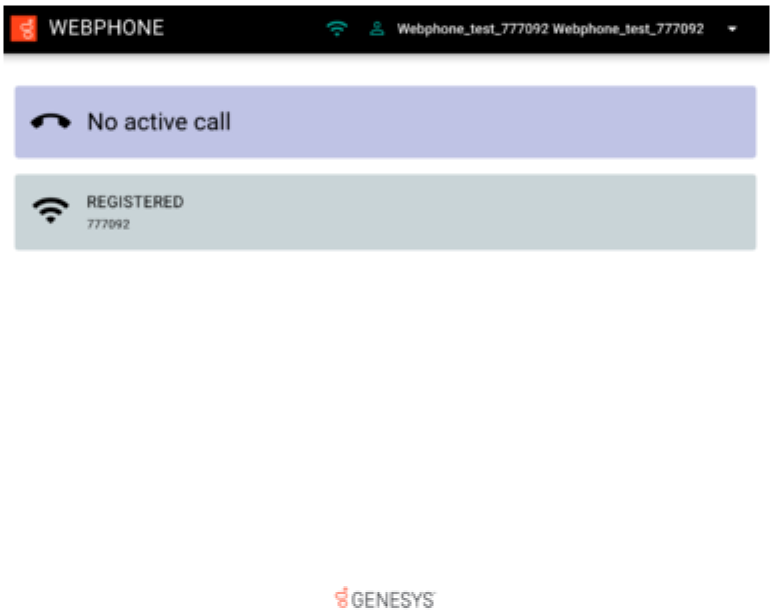
> **Important**
> Make sure that the Access Group assigned to a new user has permissions to access the Folder where the new user is created. You can view the Folder in the **General Info** tab mentioned in the user creation process.

8. Go to **Desktop Options** > **Voice**.

9. Clear the **Can Use WebRTC** check box.
10. Click **Save**.

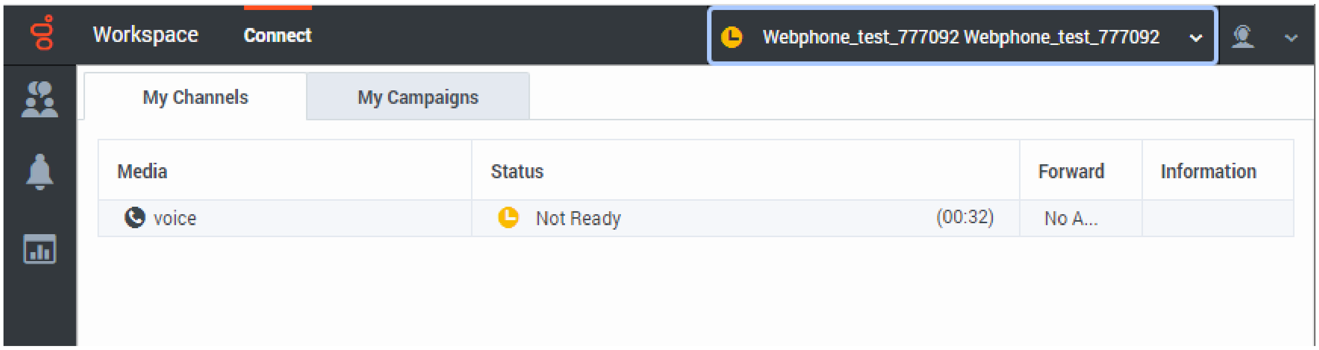
Operate Genesys Webphone

Log in to the Webphone using the credentials you have created. Log in to your Webphone as a WWE Agent now. You can view the following screen when the Webphone is successfully configured and connected to the Webphone service.



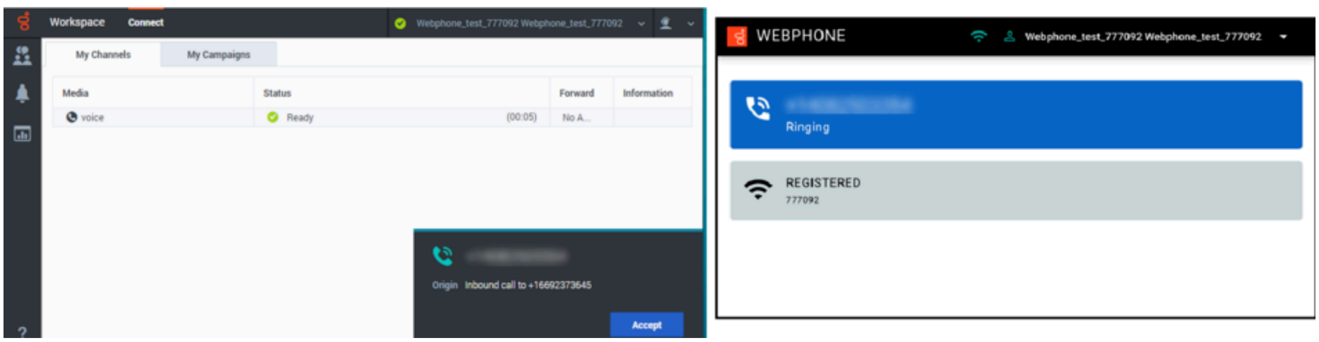
Operate WWE Agent Workspace

Log in to WWE with the user credentials that you created when configuring a WebRTC agent and you can view the following screen to initiate the Webphone call.

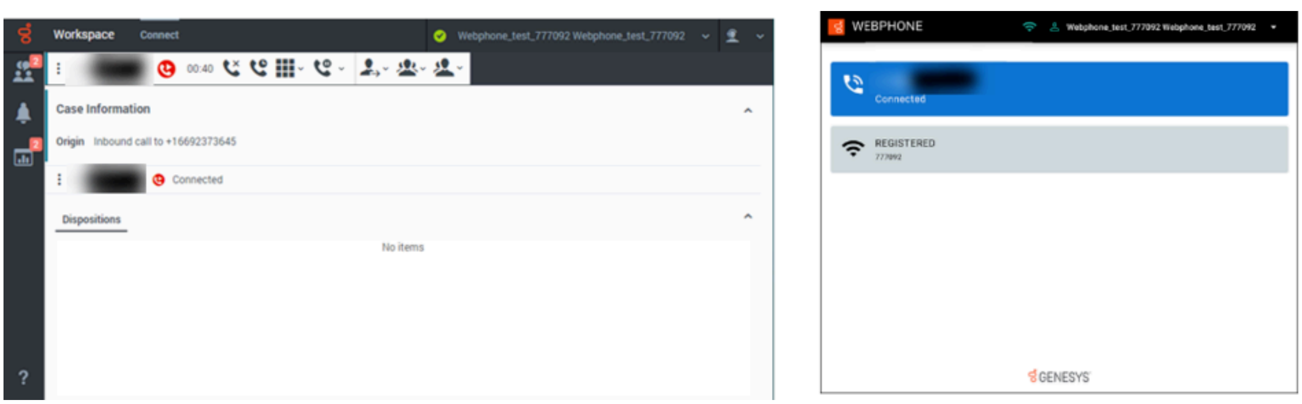


Control calls using Webphone and WWE

The agent can use the WWE desktop for all call control operations as the Webphone does not have any call control interface. The following image displays the inbound call ringing screen with WWE and Webphone.



The following image displays the screens once the call is established with WWE and Webphone.



Observability in WebRTC

Contents

Learn about the logs, metrics, and alerts you should monitor for WebRTC.

Related documentation:

-
-
-

RSS:

- For private edition

Monitoring

Private edition services expose metrics that can be scraped by Prometheus, to support monitoring operations and alerting.

- As described on Monitoring overview and approach, you can use a tool like Grafana to create dashboards that query the Prometheus metrics to visualize operational status.
- As described on Customizing Alertmanager configuration, you can configure Alertmanager to send notifications to notification providers such as PagerDuty, to notify you when an alert is triggered because a metric has exceeded a defined threshold.

The services expose a number of Genesys-defined and third-party metrics. The metrics that are defined in third-party software used by private edition services are available for you to use as long as the third-party provider still supports them. For descriptions of available WebRTC metrics, see:

- WebRTC Gateway Service metrics

See also System metrics.

Enable monitoring

The WebRTC service uses a PodMonitor custom resource definition (CRD). Monitoring is not enabled in the WebRTC service by default. To enable monitoring and expose WebRTC metrics and alerts, you must modify the Helm chart values. Set the following parameters in the **values.yaml** file to `true`:

- monitoring.enabled
- monitoring.prometheusMetrics
- monitoring.prometheusAlerts

For information about overriding Helm chart values before deployment, see Overriding Helm chart values.

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
WebRTC Gateway Service	PodMonitor	10052	/metrics	30s

Configure metrics

No further configuration is required in order to define or expose these metrics.

Alerting

Private edition services define a number of alerts based on Prometheus metrics thresholds.

> **Important**
> You can use general third-party functionality to create rules to trigger alerts based on metrics values you specify. Genesys does not provide support for custom alerts that you create in your environment.

For descriptions of available WebRTC alerts, see:

- WebRTC Gateway Service alerts

Configure alerts

Private edition services define a number of alerts by default (for WebRTC, see the pages linked to above). No further configuration is required.

The alerts are defined as **PrometheusRule** objects in a **prometheus-rule.yaml** file in the Helm charts. As described above, WebRTC does not support customizing the alerts or defining additional **PrometheusRule** objects to create alerts based on the service-provided metrics.

Logging

Refer to the Logging topic for information on configuring logging for WebRTC.

WebRTC Gateway Service metrics and alerts

Contents

Find the metrics WebRTC Gateway Service exposes and the alerts defined for WebRTC Gateway Service.

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval
WebRTC Gateway Service	PodMonitor	10052	/metrics	30s

See details about:

- WebRTC Gateway Service metrics
- WebRTC Gateway Service alerts

Metrics

WebRTC exposes many Genesys-defined as well as system metrics. You can query Prometheus directly to see all the available metrics. The metrics documented on this page are likely to be particularly useful. Genesys does not commit to maintain other currently available WebRTC metrics not documented on this page.

Metric and description	Metric details	Indicator of
wrtc_current_signins Specifies the number of current registered DNs	**Unit:** **Type:** Integer **Label:** **Sample value:** 2	Monitoring
wrtc_current_in_calls Specifies the number of current incoming calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 2	Monitoring
wrtc_current_out_calls Specifies the number of current outgoing calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 5	Monitoring
wrtc_current_audio_calls Specifies the number of current audio calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 5	Monitoring
wrtc_current_video_calls	**Unit:**	Monitoring

Metric and description	Metric details	Indicator of
Specifies the number of current video calls	**Type:** Integer **Label:** **Sample value:** 2	
wrtc_current_xcoding_calls Specifies the number of current xcoding calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 2	Monitoring
wrtc_peak_in_calls Specifies the maximum number of incoming calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 50	Monitoring
wrtc_peak_out_calls Specifies the maximum number of outgoing calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 50	Monitoring
wrtc_peak_audio_calls Specifies the maximum number of audio calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 50	Monitoring
wrtc_peak_video_calls Specifies the maximum number of video calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 50	Monitoring
wrtc_peak_xcoding_calls Specifies the maximum number of xcoding calls	**Unit:** **Type:** Integer **Label:** **Sample value:** 50	Monitoring
wrtc_total_in_calls Specifies the total number of incoming calls	**Unit:** **Type:** Counter **Label:** **Sample value:** 100	Monitoring
wrtc_total_out_calls Specifies the total number of outgoing calls	**Unit:** **Type:** Counter **Label:** **Sample value:** 100	Monitoring
wrtc_total_audio_calls Specifies the total number of audio calls	**Unit:** **Type:** Counter **Label:** **Sample value:** 100	Monitoring
wrtc_total_video_calls	**Unit:**	Monitoring

Metric and description	Metric details	Indicator of
Specifies the total number of video calls	**Type:** Counter **Label:** **Sample value:** 100	
wrtc_total_xcoding_calls Specifies the total number of xcoding calls	**Unit:** **Type:** Counter **Label:** **Sample value:** 100	Monitoring
wrtc_unauthorized_access Specifies number of unauthorized access attempts	**Unit:** **Type:** Counter **Label:** **Sample value:** 20	Monitoring
wrtc_unknown_request Specifies the number of unknown requests received	**Unit:** **Type:** Counter **Label:** **Sample value:** 20	Monitoring
wrtc_double_signin Specifies the number of registration requests that was received for registered DN	**Unit:** **Type:** Counter **Label:** **Sample value:** 20	Monitoring
wrtc_rtp_losts Specifies the number of lost RTP packets	**Unit:** **Type:** Counter **Label:** **Sample value:** 20	Monitoring
wrtc_rtp_errors Specifies the number of RTP receive errors	**Unit:** **Type:** Counter **Label:** **Sample value:** 2	Monitoring
wrtc_rtp_gateway_jitter {over="100"} Audio quality monitoring metrics	**Unit:** **Type:** Counter **Label:** {over="100"} **Sample value:**	Monitoring
wrtc_rtp_gateway_jitter Audio quality monitoring metrics	**Unit:** **Type:** Counter **Label:** {over="300"} **Sample value:**	Monitoring
wrtc_rtp_gateway_jitter Audio quality monitoring metrics	**Unit:** **Type:** Counter **Label:** {over="500"} **Sample value:**	Monitoring
wrtc_rtp_client_jitter	**Unit:**	Monitoring

Metric and description	Metric details	Indicator of
Audio quality monitoring metrics	**Type:** Counter **Label:** {over="100"} **Sample value:**	
wrtc_rtp_client_jitter Audio quality monitoring metrics	**Unit:** **Type:** Counter **Label:** {over="300"} **Sample value:**	Monitoring
wrtc_rtp_client_jitter Audio quality monitoring metrics	**Unit:** **Type:** Counter **Label:** {over="500"} **Sample value:**	Monitoring
wrtc_system_error Specifies the number of failed ICE transactions	**Unit:** **Type:** Integer **Label:** {type="turn_errors"} **Sample value:**	Error
wrtc_system_error Specifies the number of registration transactions which were timed out	**Unit:** **Type:** Integer **Label:** {type="sips", sip=""} **Sample value:** 2	Error
wrtc_system_error Specifies if WebRTC is able to connect to Elasticsearch server or not	**Unit:** **Type:** Integer **Label:** {type="es"} **Sample value:** 1 or 0	Error
wrtc_system_error Specifies the number of error responses received from Elasticsearch server	**Unit:** **Type:** Counter **Label:** {type="es_errors"} **Sample value:** 2	Error
wrtc_system_error Specifies if WebRTC is able to connect to GAuth service or not	**Unit:** **Type:** Integer **Label:** {type="auth"} **Sample value:** 1 or 0	Error
wrtc_system_error Specifies the number of error responses received from GAuth server	**Unit:** **Type:** Counter **Label:** {type="gauth_errors"} **Sample value:** 2	Error
wrtc_system_error Specifies if WebRTC is able to connect to GWS Configuration service or not	**Unit:** **Type:** Integer **Label:** {type=”cfg”} **Sample value:** 1 or 0	Error
wrtc_system_error	**Unit:**	Error

Metric and description	Metric details	Indicator of
Specifies the number of error responses received from GWS Configuration server	**Type:** Counter **Label:** {type=”cfg_errors”} **Sample value:** 2	
wrtc_system_error Specifies if WebRTC is able to connect to GWS Environments service or not	**Unit:** **Type:** Integer **Label:** {type=”env”} **Sample value:** 1 or 0	Error
wrtc_system_error Specifies the number of error responses received from GWS Environments service	**Unit:** **Type:** Counter **Label:** {type=”env_errors”} **Sample value:** 2	Error
wrtc_max_clients_per_instance Specifies the maximum number of clients per instance	**Unit:** **Type:** Constant **Label:** **Sample value:**	Performance
wrtc_max_clients_per_node Specifies the maximum number of clients per node	**Unit:** **Type:** Constant **Label:** **Sample value:**	Performance
wrtc_calls_by_domain Specifies the number of calls by domain	**Unit:** **Type:** Counter **Label:** **Sample value:**	Performance
wrtc_registrations_by_domain Specifies the number of client registrations per domain	**Unit:** **Type:** Counter **Label:** **Sample value:**	Performance
wrtc_failed_registrations_by_domain Specifies the number of failed client registrations per domain	**Unit:** **Type:** Counter **Label:** **Sample value:**	Performance
wrtc_client_errors Specifies the number of double sign-ins being rejected	**Unit:** **Type:** Counter **Label:** (type = double_sign_in_reject) **Sample value:**	Error
wrtc_client_errors Specifies the number of clients being dropped on sign-out	**Unit:** **Type:** Counter **Label:** (type = dropped_on_signout) **Sample value:**	Error
wrtc_client_errors	**Unit:**	Error

Metric and description	Metric details	Indicator of
Specifies the number of client timeout errors	**Type:** Counter **Label:** (type = timeout) **Sample value:**	

Alerts

The following alerts are defined for WebRTC Gateway Service.

Alert	Severity	Description	Based on	Threshold
webrtc-gateway-signins	warning	Specifies the number of sign-ins	wrtc_current_signins	15mins
webrtc-gateway-gauth	warning	Specifies that the Gateway Pod has lost connection to Auth service	wrtc_system_error	Need input
webrtc-gateway-gws	warning	Specifies that the Gateway Pod has lost connection to the Environment Service	wrtc_system_error	Need input
webrtc-gateway-es	warning	Specifies that the Gateway Pod has lost connection to ElasticSearch	wrtc_system_error	Need input

Logging

Contents

Learn how to store logs for WebRTC.

Related documentation:

-
-
-

RSS:

- For private edition

Gateway

Log files are mandatory for troubleshooting process. WebRTC creates logs in both logfiles and Elasticsearch. Logs are created either using `HostPath` or `PersistentVolumeClaim` parameters.

Use the following formula to calculate the approximate log storage volume:

```
(Number of Agents) * (Average call load per hour) * (0.2 constant) * (Number of
active hours)
```

As all operations are of the same order, it can be scaled up or down linearly. For example, if 500 GB is required for 1000 agents with 10 calls per hour for 24 hours, then for 2000 agents with the same load and time we need around 1TB (both are with a slight buffer for idling time).

CoTurn

Coturn creates logs in logfiles or `stdout`. Logs are created either using `HostPath` or `PersistentVolumeClaim` parameters.

A full day (24H) of logging for a 1000 connected agents will give us the required capacity of around 100 GB of storage, considering that we have a buffer for standardization. For easy calculation, the capacity can be scaled linearly from 100 GB for more number of agents. Use the following formula for the exact calculation:

```
3.6MB/hr * number of hours * number of agents
```