Genesys Authentication Private Edition Guide

5/21/2026

Table of Contents

Overview

About Genesys Authentication 6

Architecture 9

High availability and disaster recovery 15

Configure and deploy

Before you begin 16

Configure Genesys Authentication 21

Deploy Genesys Authentication 41

Provision Genesys Authentication 56

Provision SAML-based SSO 65

Upgrade, rollback, or uninstall Genesys Authentication 74

Observability

Observability in Genesys Authentication 77

Authentication Service metrics and alerts 80

Environment Service metrics and alerts 86

Contents

- 1 Overview
- 2 Configure and deploy
- 3 Observability

Find links to all the topics in this guide.

Related documentation:

-
-

RSS:

- For private edition

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

Overview

Learn more about Genesys Authentication and how to get started.

- About Genesys Authentication
- Architecture
- High availability and disaster recovery

Configure and deploy

Find out how to configure and deploy Genesys Authentication.

- Before you begin
- Configure Genesys Authentication
- Deploy Genesys Authentication
- Provision Genesys Authentication
- Provision SAML-based SSO
- Upgrade, rollback, or uninstall Genesys Authentication

Observability

Learn how to monitor Genesys Authentication with metrics and logging.

- Observability in Genesys Authentication
- Authentication Service metrics and alerts
- Environment Service metrics and alerts

About Genesys Authentication

Contents

- 1 Supported Kubernetes platforms

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

Related documentation:

-
-
-

RSS:

- For private edition

Genesys Authentication provides authentication capabilities for Genesys Multicloud CX private edition services and applications. Genesys Authentication is based on the OAuth 2.0 authorization framework, with support for OpenID Connect. It supports the following OAuth grant types:

- Authorization Code
- Client Credentials
- Refresh Token
- Password
- Implicit
- Token Convert
- Assertion

Genesys Authentication confirms a client identity, or a client and user identities, and provides related metadata. It does **NOT** do authorization or handle and manage permissions - this is the responsibility of the authentication service client. The Authentication API service authenticates a user against a tenant's Configuration Server or a tenant's identity provider (IdP), if configured for single sign-on (SSO) use. See Single sign-on for details about how to set up SSO. You can have both Configuration Server and IdP authentication functionalities for a particular tenant. When a client makes a successful authentication attempt, the Authentication API service provides an API access token. The service also verifies existing tokens.

Genesys Authentication has three components, which are always distributed together:

- Authentication API service - Provides the authentication capabilities described above.
- Authentication UI service - A user interface used by many Genesys Multicloud CX private edition applications for log in and change password functionality. See Log in to Genesys Multicloud CX for details.
- Environment API service - An internal service that manages contact centers and environments. An environment contains information about connecting to Configuration Server and can have one or more contact centers.

Supported Kubernetes platforms

Genesys Authentication is supported on the following cloud platforms:

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

See the Authentication, Login, and SSO Release Notes for information about when support was introduced.

Architecture

Contents

- 1 Introduction
- 2 Architecture diagram — Connections
- 3 Connections table

Learn about Genesys Authentication architecture

Related documentation:

-
-
-

RSS:

- For private edition

Introduction

The diagram below shows the architecture of the Genesys Authentication components:

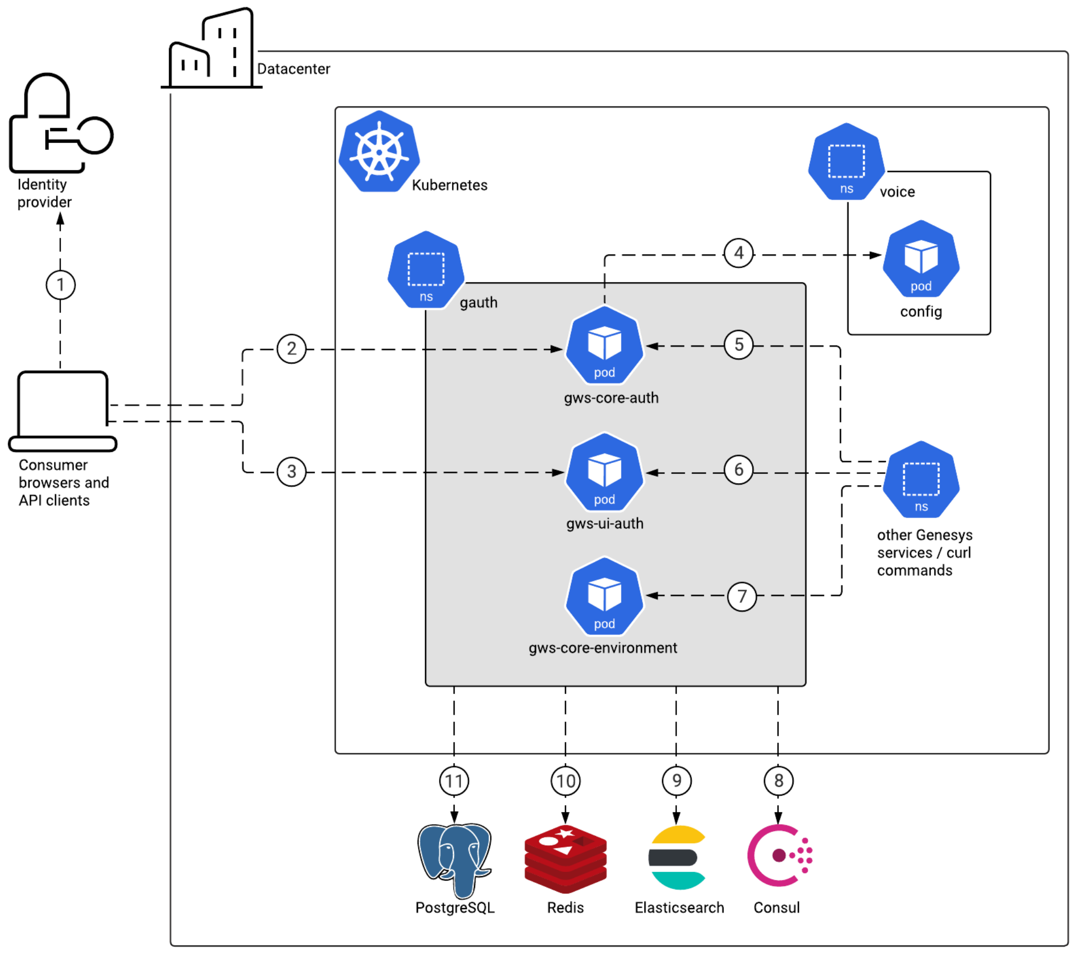
- Authentication API service
- Authentication UI service
- Environment API service

For information about the overall architecture of Genesys Multicloud CX private edition, see the high-level Architecture page.

See also High availability and disaster recovery for information about high availability/disaster recovery architecture.

Architecture diagram — Connections

The numbers on the connection lines refer to the connection numbers in the table that follows the diagram. The direction of the arrows indicates where the connection is initiated (the source) and where an initiated connection connects to (the destination), from the point of view of Genesys Authentication as a service in the network.



Connections table

The connection numbers refer to the numbers on the connection lines in the diagram. The **Source**, **Destination**, and **Connection Classification** columns in the table relate to the direction of the arrows in the Connections diagram above: The source is where the connection is initiated, and the destination is where an initiated connection connects to, from the point of view of Genesys Authentication as a service in the network. *Egress* means the Genesys Authentication service is the source, and *Ingress* means the Genesys Authentication service is the destination. *Intra-cluster* means the connection is between services in the cluster.

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
1	Consumer browser	Identity provider	HTTPS	443	Ingress	For single sign-on support, the

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
						consumer's browser communicates with the identity provider (IdP).
2	Consumer browser and API clients	Authentication Service	HTTPS	443	Ingress	Consumer browsers and API clients use one of the supported OAuth 2.0 grant types to authenticate. See the Authentication API for details.
3	Consumer browser	Authentication UI	HTTPS	443	Ingress	If an application uses the Genesys Authentication UI, users are redirected to the log in page. See for details.
4	Authentication Service	Voice Platform Configuration Server	TCP	8888	Ingress	Data from Configuration Server.
5	Other Genesys services	Authentication Service	HTTP/HTTPS	80/443	Ingress	Genesys services authenticate with Authentication API. Enable Transport Layer Security for this connection with in the **values.yaml** file.
6	Other	Authentication	HTTP/HTTPS	80/443	Ingress	Applications

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
	Genesys services	UI				that use the Genesys Authentication UI. Enable Transport Layer Security for this connection with in the **values.yaml** file.
7	Other Genesys services / curl commands	Environment Service	HTTP/HTTPS	80/443	Ingress	Other Genesys services and the private edition installer (through curl commands) use the Environment API to manage their environments, contact centers, and settings. Enable Transport Layer Security for this connection with in the **values.yaml** file.
8	Genesys Authentication	Consul	HTTPS	443	Egress	Discovery of Configuration Server endpoints. This connection is optional and controlled by the options in the **values.yaml** file.

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
9	Genesys Authentication	Elasticsearch	TCP	9200	Egress	Logging data.
10	Genesys Authentication	Redis	TCP	6379 (non SSL) or 6380 (SSL)	Egress	Session data. SSL is controlled by in the **values.yaml** file.
11	Genesys Authentication	PostgreSQL	TCP	5432	Egress	Configuration data for the Authentication Service and the Environment Service.

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?
Genesys Authentication	N = N (N+1)	Active-spare	Primary or secondary unit

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

Before you begin

Contents

- 1 Download the Helm charts
- 2 Third-party prerequisites
- 3 Storage requirements
- 4 Network requirements
  - 4.1 Ingress
  - 4.2 Cookies
- 5 Browser requirements
- 6 Genesys dependencies

Find out what to do before deploying Genesys Authentication.

Related documentation:

-
-
-

RSS:

- For private edition

Download the Helm charts

Genesys Authentication in Genesys Multicloud CX private edition is made up of three containers, one for each of its components:

- gws-core-auth - Authentication API service
- gws-ui-auth - Authentication UI service
- gws-core-environment - Environment API service

The service also includes a Helm chart, which you must deploy to install all three containers for Genesys Authentication:

- gauth

See Helm charts and containers for Authentication, Login, and SSO for the Helm chart version you must download for your release.

To download the Helm chart, navigate to the **gauth** folder in the JFrog repository. See Downloading your Genesys Multicloud CX containers for details.

Third-party prerequisites

Install the prerequisite dependencies listed in the **Third-party services** table before you deploy Genesys Authentication.

Third-party services

Name	Version	Purpose	Notes
PostgreSQL	11.x	Relational database.	Genesys Authentication supports PostgreSQL

Name	Version	Purpose	Notes
			12.x.
Redis	6.x	Used for caching. Only distributions of Redis that support Redis cluster mode are supported, however, some services may not support cluster mode.	Redis must be in cluster mode.
Consul	1.13.x	Service discovery, service mesh, and key/value store.	
Ingress controller		HTTPS ingress controller.	
HTTPS certificates - Let's Encrypt		Use with cert-manager to provide free rotating TLS certificates for NGINX Ingress Controller. **Note:** Let's Encrypt is a suite-wide requirement if you choose an Ingress Controller that needs it.	
HTTPS certificates - cert-manager		Use with Let's Encrypt to provide free rotating TLS certificates for NGINX Ingress Controller.	
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

Genesys Authentication uses PostgreSQL to store key/value pairs for the Authentication API and Environment API services. It uses Redis to cache data for the Authentication API service.

Network requirements

Ingress

Genesys Authentication supports both internal and external ingress with two ingress objects that are configured with the **ingress** and **internal_ingress** settings in the **values.yaml** file. See Configure Genesys Authentication for details about overriding Helm chart values.

- ingress - External ingress for UIs and external API clients. External ingress can be public.
- internal_ingress - Internal ingress for internal API clients. Internal ingress contains an extended list of API endpoints that are not available for external ingress. Internal ingress should not be public.

These ingress objects support Transport Layer Security (TLS) version 1.2. TLS is enabled by default and you can configure it by overriding the **ingress.tls** and **internal_ingress.tls** settings in **values.yaml**.

For example:

```
 ingress:
  enabled: true
  frontend: gauth.example.com
  tls_enabled: true
  tls:
    - hosts:
        - gauth.example.com
      secretName: gauth-example-com

internal_ingress:
  enabled: true
  frontend: gauth.int.example.com
  tls_enabled: true
  tls:
    - hosts:
        - gauth.int.example.com
      secretName: gauth-int-example-com
```

In the example above:

- **secretName** is the certificate and private key to use for TLS. The secret is a prerequisite and must be created before you deploy Genesys Authentication, unless you have Certificate ClusterIssuer installed and configured in Kubernetes Cluster. In this case, the secret is created by ClusterIssuer.
- **hosts** is a list of the fully qualified domain names that should use the certificate. The list must be the same as the value configured for **ingress.frontend** and **internal_ingress.frontend**.

Cookies

Genesys Authentication components use cookies to identify HTTP/HTTPS user sessions.

Browser requirements

The Authentication UI supports the web browsers listed in the **Browsers** table.

Browsers

Name	Version	Notes
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.
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.
Microsoft Edge (Legacy)	Current release	
Microsoft Edge Chromium	Current release	

Genesys dependencies

Genesys Authentication must be deployed before other Genesys Multicloud CX private edition services. To complete provisioning the service, you must first deploy Web Services and Applications and the Tenant Service. For a look at the high-level deployment order, see Order of services deployment.

Configure Genesys Authentication

Contents

- 1 Add Java KeyStore support (optional)
- 2 Configure a secret to access JFrog
- 3 Override Helm chart values
- 4 Provision Consul-less Deployment
- 5 Configure Kubernetes
  - 5.1 ConfigMaps
  - 5.2 Secrets
- 6 Configure security

Learn how to configure Genesys Authentication.

Related documentation:

-
-
-

RSS:

- For private edition

Complete the steps on this page to configure your Genesys Authentication deployment.

Add Java KeyStore support (optional)

Complete the steps in this section to set up a Java KeyStore (JKS) if you need to configure Genesys Authentication to use JSON Web Token authentication. This method of authentication is currently used for WebRTC.

Create a keystore file:

```
keytool -keystore jksStorage.jks -genkey -alias gws-auth-key -storepass  -keypass  -keyalg RSA
```

Get the Base64 encoded key:

```
cat ./jksStorage.jks | base64
```

The result looks like this:

```
/u3+7QAAAAIAAAABAAAAAQAMZ3dzLWF1dGgta2V5AAABeRmB2Y4AAAUBMIIE/
TAOBgorBgEEASoCEQEBBQAEggTpwQQ5aW5CUYAsf4/IheBuNrlPPyZhUA+NWh3SG52HV3sVjV+p18vKp2k/
q12I9NynoM6R/
DW5bFfEWU1zx3cfXH2kNirRUOIbNZpa43NOroyyF1GSdZFlwa8Kq8Xtp8ZBmiJdSb1n12ODaTKGKv1cb5tsfdzkWs99QeTBGJypHMCdnBvdFB0N
mMACTHk4R9yASsd7fljgNLSn0jhrz9FuxvYgpOVvExiq+sb5YrfbZjtTzZDzFVOu/
2kWzASfZBSiyyxMOr3IhUPkMpIrg+UYkI0tgn/
C3yR1wLr9HElpx8fCu61ORqp8hhp1yvL46K0c6eTa2JcRpO6fmysf2EG0JagG7zNEJHlvtNnt3JpQV06xos2iWsFAtHq+9w8LwvCVbDzx/
UHoCYenIdJ7SBv06mXgKisa3RDIi/y5x5/9T4brgCLUvwI4Z5Rf/oi2Zx5/lXjQXmBPlPAcUVHLr5PvNQUUx5NBr/
ooioD7qka4ADF1/
cx8I2bzqTi+U01fiFdMGRlNlCfcGDMI2h82JUeCswRYi4+dMDiSaGgC2MoL2susLxMYa5CTo9Vs0Y2k+6j8fhIO4h8h0JxdXZODU63OM0cDSUHX
4IhiV3k7W4OHYeXUeDvoNmfo/AriELZl+WgYETiXGsKzxmrsHrBKC0+aT098FwqdY9ACsM/
7WoF2+9eftc7fa2jruutrRjmk0A/
BaIqzboJLFiWaUUGV9gsexEmpGszikQsmOYSIRxY8BYF+SYldehcfcsRRxDnhTaGNV8y2ZnwA61FNPAFps3gaFXeaYsUzlxTSi9m70HJJrUp7JD
KFi9OrEuAdhMJa+iQ21PBZ+iIwxb0y9xMReImoUtoqy6Epre3qMOS6MILLw2bVrxJYo38+hR5uzNdlbsUlpYOoorI1Hp8A/
VEYtG9PDHEhhoqUamdUYUzkFDi9QZfylIgi8Jc4G4PPrPKgMPqqE7sl6bJvoLavU58eHpdWo/Mb9UtdTx+l/
SlulCCE0Xce6M9YE1SyC2B3gd82zNQa81lx+QAY8IaSmX+C2nMz+UeXKngSEzguK6gXg9RwCs8pUavuLQ6uZGkJ+fhDBvDAFgD7hG1XdHs27XGS
DT/KHRB7AHN5/
vQpj6KOscxqmyPrgPY/+TseczEeaQLQ6MfjvXY+AAAAAQAFWC41MDkAAAN7MIIDdzCCAl+gAwIBAgIEYxhLHTANBgkqhkiG9w0BAQsFADBsMRAw
esYcJNEqu1btJLwLvhXb651OyZnsmeNGP2BrNCPXZS6CBReMMKJaZrlCwJQxiSrGPHB/
```

```
gpxKoAowLwl3V7wB2BHKDhrczQBPdvtsfBAzeqpN/
yRpdKZRAtu2LyGqRZKCglSrwYenJFqROdOeworbNmtIKXfQLiamE4KdhzQdPfnyBC7ZWtCIJUp9Va4LmCYD/
ISOmVyfQ9Xql1rRNQLcVaewCKRM2ffBAkx98d3n79XUZDljOzHh+79tCpheuuYfbMQqMCAwEAAaMhMB8wHQYDVR0OBBYEFNtM8mIEb67VYot5tj
Ta4y+B6JcdPjFtII6Pf5W0DDTOa3cHNMeukYn5lBnaMbIKqoxFT7nM7MD3DB+dISvMu8FtVWFwbPzXWhl+Aycuu9ETGlCoJqYfl+vmLyGjJVadc
YbN7be2QIJwmucIZzH7fkU90V+rmVZhl9Bo8ixuIJG/vZTxmEBaDqmhiP4w=
```

Make note of the following values - you need them to configure JKS support in the Helm chart:

- Keystore filename
- Keystore password
- Key alias
- Key password
- Base64 encoded key

Configure a secret to access JFrog

If you haven't done so already, create a secret for accessing the JFrog registry:

```
kubectl create secret docker-registry  --docker-server= --docker-username= --docker-password=
--docker-email=
```

Now map the secret to the default service account:

```
kubectl secrets link default  --for=pull
```

Override Helm chart values

You can specify parameters for the deployment by overriding Helm chart values in the **values.yaml** file. See the **Parameters** table for a full list of overridable values.

For more information about how to override Helm chart values, see Overriding Helm chart values in the Setting up Genesys Multicloud CX Private Edition guide.

Parameters

Parameter	Description	Valid values	Default
gws-core-auth	The gws-core-auth image version tag. For example, 100.0.003.3508.	A valid image version	""
gws-core-environment	The gws-core-environment image version tag. For example, 100.0.003.1866.	A valid image version	""
gws-ui-auth	The gws-ui-auth image version tag. For	A valid image version	""

Parameter	Description	Valid values	Default
	example, 100.0.003.1328.		
image.imagePullSecrets	The secret Kubernetes uses to get credentials to pull images from the registry.	A valid secret	[]
image.pullPolicy	Specifies when Kubernetes pulls images from the registry on start up.	IfNotPresent or Always	"IfNotPresent"
image.registry	Docker registry address	A valid registry URL	""
consul.discovery_register	Specifies whether services are registered in Consul.	true or false	false
consul.discovery_tenants	Enables tenant discovery through Consul.	true or false	true
consul.enabled	Enables a connection to Consul.	true or false	false
consul.host	The host of the local Consul agent.	A valid URL	"http://$(K8_HOST_IP)"
consul.port	The port of the local Consul agent.	A valid port	8500
consul.require_token	Specifies whether Genesys Authentication reads the API token from a Kubernetes secret.	true or false	false
consul.secret.create	Create or use an existing secret with the Consul API token.	true or false	false
consul.secret.name_override	The name of the Kubernetes secret for Consul.	A valid secret name	nil
consul.secret.token	The API token to access Consul.	A valid API token	nil
ingress.enabled	Enables external ingress for Genesys Authentication.	true or false	true
ingress.frontend	The host that is used by external ingress.	A valid host	"gauth.local"
ingress.annotations.	Annotations that are applied to external ingress. See the Kubernetes documentation for details.	A valid set of annotations as "name: value"	nginx.ingress.kubernetes.io/proxy-body-size: "0"
ingress.tls_enabled	Enables Transport Layer	true or false	true

Parameter	Description	Valid values	Default
	Security (TLS) on external ingress.		

ingress.tls	The name of the secret for Secure Sockets Layer (SSL) certificates.	A valid secret name	`- hosts:`
			`  - gauth.local`
			`  secretName:`
			`letsencrypt`

internal_ingress.enabled	Enables internal ingress for Genesys Authentication.	true or false	true
internal_ingress.frontend	The host that is used by internal ingress.	A valid host	"gauth-int.local"
internal_ingress.annotations	Annotations that are applied to internal ingress. See the Kubernetes documentation for details.	A valid set of annotations as "name: value"	nginx.ingress.kubernetes.io/proxy-body-size: "0"
internal_ingress.tls_enabled	Enables Transport Layer Security (TLS) on internal ingress.	true or false	true

internal_ingress.tls	The name of the secret for Secure Sockets Layer (SSL) certificates.	A valid secret name	`- hosts:`
			`  - gauth-int.local`
			`  secretName:`
			`letsencrypt`

monitoring.enabled	Specifies whether to deploy Custom Resource Definitions (CRD) for ServiceMonitors to determine which services should be monitored.	true or false	false
monitoring.interval	The interval at which Prometheus scrapes metrics.	A duration in seconds	"15s"
monitoring.alarms	Specifies whether to deploy CRD for PrometheusRules to define rules for alarms.	true or false	false
monitoring.alarmThresholds.redisKeys	The threshold to trigger an alarm on the total number of keys in Redis.	Number	5000000
monitoring.alarmThresholds.redisMaxMemoryPerentage	The threshold to trigger an alarm for used Redis memory.	Number	85
monitoring.dashboards	Specifies whether to deploy ConfigMaps with	true or false	false

Parameter	Description	Valid values	Default
	Grafana Dashboards.		
monitoring.pagerduty	Enables alarms with a severity of `CRITICAL`.	true or false	true

optional.affinity	Specifies the affinity and anti-affinity for Genesys Authentication pods. See the Kubernetes documentation for details.	Object	`podAntiAffinity:`
			`  preferredDuringSchedulingIgnoredDuringExecution:`
			`    - podAffinityTerm:`
			`        labelSelector:`
			`          matchLabels:`
			`            gauth: '{{ .gauth }}'`
			`            app.kubernetes.io/name: '{{ include "auth.name" . }}'`
			`            app.kubernetes.io/instance: '{{ .Release.Name }}'`
			`        topologyKey: failure-domain.beta.kubernetes.io/zone`
			`      weight: 100`

optional.dnsConfig	Specifies custom DNS settings for Genesys Authentication pods. See the Kubernetes documentation for details.	Object	`options:`
			`  - name: ndots`
			`    value: "3"`

optional.dnsPolicy	Specifies the DNS policy for Genesys Authentication pods. See the Kubernetes documentation for details.	"Default", "ClusterFirst", "ClusterFirstWithHostNet", or "None"	"ClusterFirst"
optional.nodeSelector	The labels Kubernetes uses to assign pods to nodes. See the Kubernetes documentation for details.	Object	`{}`
optional.priorityClassName	The class name Kubernetes uses to determine the priority of a pod relative to other pods. See the Kubernetes documentation for details.	A valid priority class name	""
optional.securityContext	Specifies the privilege and access control	Object	`{}`

Parameter	Description	Valid values	Default
	settings Genesys Authentication pods. See Configure security for details.		

optional.strategy	Specifies details for the rolling update strategy Genesys Authentication uses to upgrade it containers. See the Kubernetes documentation for details.	Object	`type: RollingUpdate`
			`rollingUpdate:`
			`  maxSurge: 10`
			`  maxUnavailable: 0`

optional.tolerations	The tolerations Kubernetes uses for advanced pod scheduling. See the Kubernetes documentation for details.	Object	[]
podDisruptionBudget.create	Specifies whether to create a PodDisruptionBudget. See the Kubernetes documentation for details.	true or false	false
podDisruptionBudget.spec	Specifies the details of your PodDisruptionBudget. See the Kubernetes documentation for details.	A valid spec that defines a value for either minAvailable or maxUnavailable. Do not specify .spec.selector because it is calculated by Helm.	minAvailable: 2
pod_autoscaler.auth.enabled	Enables the Horizontal Pod Autoscaler for the Authentication Service. See the Kubernetes documentation for details.	true or false	false
pod_autoscaler.auth.maxReplicas	Specifies the maximum number of Authentication Service replicas the Horizontal Pod Autoscaler controller will scale.	Number	10

pod_autoscaler.auth.metrics	Specifies resource metrics the Horizontal Pod Autoscaler controller uses to scale Authentication Service pods up or down. See the Kubernetes documentation for	Object	`- type: Resource`
			`  resource:`
			`    name: cpu`
			`    target:`
			`      type:`
			`Utilization`
			`      averageUtilizati`

Parameter	Description	Valid values	Default
	details.		`on: 350%`
pod_autoscaler.environment.enabled	Enables the Horizontal Pod Autoscaler for the Environment Service. See the Kubernetes documentation for details.	true or false	false
pod_autoscaler.environment.maxReplicas	Specifies the maximum number of Environment Service replicas the Horizontal Pod Autoscaler controller will scale.	Number	10

pod_autoscaler.environment.metrics	Specifies resource metrics the Horizontal Pod Autoscaler controller uses to scale Environment Service pods up or down. See the Kubernetes documentation for details.	Object	`- type: Resource`
			`  resource:`
			`    name: cpu`
			`    target:`
			`      type: Utilization`
			`      averageUtilization: 350%`

postgres.deploy	Specifies whether to deploy PostgreSQL. Set this option for lab environments only.	true or false	false
postgres.image	Specifies the Docker image to use in the lab environment if `postgres.deploy=true`.	A Docker image	"postgres:11-alpine"
postgres.configmap.create	Specifies whether Genesys Authentication creates a ConfigMap with PostgreSQL connection parameters. If the value is false, you must create the ConfigMap manually.	true or false	false
postgres.configmap.name_override	The name of the ConfigMap.	A value name	nil
postgres.db	The name of the PostgreSQL database from Create a PostgreSQL database and user.	A valid database name	nil
postgres.host	The host of the PostgreSQL instance.	A valid host	nil
postgres.port	The port of the	A valid port	nil

Parameter	Description	Valid values	Default
	PostgreSQL instance.		
postgres.username	The username to access the PostgreSQL database from Create a PostgreSQL database and user.	A valid username	nil
postgres.password	The password to access the PostgreSQL database from Create a PostgreSQL database and user.	A valid password	nil
postgres.secret.create	Specifies whether to create a Kubernetes secret with user credentials for PostgreSQL. If this value is false, you must create the secret manually.	true or false	false
postgres.secret.name_override	The name of the PostgreSQL secret.	A valid name	nil
redis.cluster_nodes	The Redis nodes in your cluster. For example, "redis-cluster1:7000,redis-cluster2:7002".	A comma-separated list of "host:port" pairs	nil
redis.configmap.create	Specifies whether to create a ConfigMap with connection parameters for Redis. If this value is false, you must create the ConfigMap manually.	true or false	false
redis.configmap.name_override	The name of the Redis ConfigMap.	A valid name	nil
redis.deploy	Specifies whether to deploy a Redis cluster. Set this option for lab environments only.	true or false	false
redis.image	Specifies the Docker image to use in the lab environment if `redis.deploy=true`.	A Docker image	"redis:5-stretch"
redis.password	The Redis password.	A valid password	nil
redis.password_required	Specifies whether Genesys Authentication should read the Redis password from a Kubernetes secret.	true or false	false
redis.secret.create	Specifies whether to create a Kubernetes secret with Redis	true or false	false

Parameter	Description	Valid values	Default
	password. If this value is false, you must create the secret manually.		
redis.secret.name_override	The name of the Redis secret.	A valid name	nil
redis.use_tls	Enable or disable a TLS connection to the Redis cluster.	true or false	false
serviceAccount.create	Specifies whether to create a service account.	true or false	false
serviceAccount.name	The name of the service account to use.	A service account name	""
serviceAccount.annotations	Annotations to add to the service account. See the Kubernetes documentation for details.	A valid set of labels as "name: value"	{}
services.initContainers	Optional init containers to add to Genesys Authentication deployments.	Object	{}
services.location	Location of the deployment. For example, "/USW1".	A valid location.	"/"
services.replicas	The number of Genesys Authentication pod replicas to deploy.	Number	3
services.db.init	Enable automatic updates for the database schema.	true or false	true
services.db.poolCheckoutTimeout	The database pool timeout.	Number	3000
services.db.poolSize	The database pool size.	Number	3
services.auth.loglevel	Specifies the log level for the Authentication Service.	INFO, DEBUG, WARN	DEBUG
services.db.ssl	Enable or disable an SSL connection to PostgreSQL. See the PostgreSQL documentation for details about SSL modes.	disable, prefer, require, verify-ca, or verify-full	"disable"
services.auth.deploymentAnnotations	Annotations for Authentication Service deployment objects. See the Kubernetes documentation for	A valid set of annotations as "name: value"	{}

Parameter	Description	Valid values	Default
	details.		
services.auth.env.GWS_AUTH_SECURITY_HTTP_SCHEME_HEADER_NAME	The name of the header with protocol. This value can be used when HTTPS is terminated by the load balancer.	A valid header name	"X-Forwarded-Proto"
services.auth.env.GWS_AUTH_timeouts_requestTimeoutMs	The Authentication Service request timeout.	A value in milliseconds	30000
services.auth.env.JAVA_TOOL_OPTIONS	Specifies JVM arguments to set in the JAVA_TOOL_OPTIONS environment variable.	Valid JVM arguments	"-XX:+PrintFlagsFinal -XX:+UseG1GC -Dfile.encoding=UTF-8 -XX:+ExitOnOutOfMemoryError -XX:MaxRAMPercentage=80.0"
services.auth.env.GWS_AUTH_logging_level_com_genesys_gws_v3	Specifies the log level for the Authentication Service.	INFO, DEBUG, WARN	DEBUG
services.auth.env.GWS_AUTH_http_headers_frame_options	Specifies the value of the X-Frame-Options HTTP response header.	SAMEORIGIN, DENY, DISABLE, ALLOW	ALLOW
services.auth.jks.enabled	Specifies whether Genesys Authentication uses Java KeyStore. See Add JKS support for details. This value must be set to true for Security Assertion Markup Language single sign-on (SAML SSO) functionality.	true or false	false
services.auth.jks.keyAlias	The name of the key alias in the keystore used by the Authentication Service. This value comes from Add JKS support.	A valid key alias	nil
services.auth.jks.keyPassword	The keystore password from Add JKS support.	A valid keystore password	nil
services.auth.jks.keyStore	The name of the Java keystore file from Add JKS support.	A valid keystore name	"jksStorage.jks"
services.auth.jks.keyStorePassword	The keystore password from Add JKS support.	A valid keystore password	nil
services.auth.jks.secret.create	Specifies whether to create a new secret with the keystore file content and keystore credentials.	true or false	true

Parameter	Description	Valid values	Default
services.auth.jks.keyStoreFileData	The Base64 encoded key value from Add JKS support.	A valid key	nil
services.auth.jks.secret.name	A Kubernetes secret name with the keystore credentials and content.	A valid secret name	nil
services.auth.jks.sso.enabled	Specifies whether to enable SAML SSO functionality.	true or false	false

services.auth.livenessProbe	Specifies parameters for the livenessProbe. See the Kubernetes documentation for details.	Object	`livenessProbe:`
			`  httpGet:`
			`    path: /health`
			`    port: management`
			`  initialDelaySeconds: 120`
			`  periodSeconds: 10`
			`  successThreshold: 1`
			`  timeoutSeconds: 3`
			`  failureThreshold: 3`

services.auth.ports.management	Specifies the management port for Authentication Service.	Number	8081
services.auth.ports.service	Specifies the service port for Authentication Service.	Number	8080

services.auth.readinessProbe	Specifies parameters for the readinessProbe. See the Kubernetes documentation for details.	Object	`readinessProbe:`
			`  httpGet:`
			`    path: /health`
			`    port: management`
			`  initialDelaySeconds: 30`
			`  timeoutSeconds: 3`
			`  periodSeconds: 10`

services.auth.replicas	The number of Authentication Service pod replicas to deploy. This value overrides services.replicas.	Number	nil

services.auth.resources	The requests and limits for Authentication Service pod resources. See the Kubernetes documentation for details.	Object	`requests:`
			`  cpu: 500m`
			`  memory: 4Gi`
			`limits:`
			`  cpu: "4"`
			`  memory: 6Gi`

services.auth.serviceAnnotations	Annotations for Authentication Service service objects. See the Kubernetes	A valid set of annotations as "name: value"	{}

Parameter	Description	Valid values	Default
	documentation for details.		
services.auth_ui.deploymentAnnotations	Annotations for Authentication UI deployment objects. See the Kubernetes documentation for details.	A valid set of annotations as "name: value"	{}
services.auth_ui.ports.service	Specifies the service port for Authentication UI.	Number	8080
services.auth_ui.env.GWS_NGINX_ENABLE_MAPPING	Use Consul to discover Auth Service		"false"
services.auth_ui.livenessProbe	Specifies parameters for the livenessProbe. See the Kubernetes documentation for details.	Object	{}
services.auth_ui.readinessProbe	Specifies parameters for the readinessProbe. See the Kubernetes documentation for details.	Object	{}
services.auth_ui.replicas	The number of Authentication UI pod replicas to deploy. This value overrides services.replicas.	Number	nil

services.auth_ui.resources	The requests and limits for Authentication UI pod resources. See the Kubernetes documentation for details.	Object	`requests:`
			`  cpu: 100m`
			`  memory: 500Mi`
			`limits:`
			`  cpu: 500m`
			`  memory: 1Gi`

services.auth_ui.serviceAnnotations	Annotations for Authentication UI service objects. See the Kubernetes documentation for details.	A valid set of annotations as "name: value"	{}
services.environment.loglevel	Specifies the log level for the Environment Service.	INFO, DEBUG, WARN	INFO
services.environment.deploymentAnnotations	Annotations for Environment Service deployment objects. See the Kubernetes documentation for details.	A valid set of annotations as "name: value"	{}

Parameter	Description	Valid values	Default
services.environment.env.JAVA_TOOL_OPTIONS	Specifies JVM arguments to set in the JAVA_TOOL_OPTIONS environment variable.	Valid JVM arguments	"-XX:+PrintFlagsFinal -XX:+UseG1GC -Dfile.encoding=UTF-8 -XX:+ExitOnOutOfMemoryError -XX:MaxRAMPercentage=80.0"
services.environment.env.GWS_ENVIRONMENT_logging_level_com_genesys_gws_v3	Specifies the log level for the Environment Service.	INFO, DEBUG, WARN	INFO
services.environment.force_writable	Ignore the Data Center topology in a single-region deployment.	true or false	true

services.environment.livenessProbe	Specifies parameters for the livenessProbe. See the Kubernetes documentation for details.	Object	`livenessProbe:`
			`  httpGet:`
			`    path: /health`
			`    port: management`
			`  initialDelaySeconds: 120`
			`  periodSeconds: 10`
			`  successThreshold: 1`
			`  timeoutSeconds: 3`
			`  failureThreshold: 3`

services.environment.ports.management	Specifies the management port for Environment Service.	Number	8081

services.environment.readinessProbe	Specifies parameters for the readinessProbe. See the Kubernetes documentation for details.	Object	`readinessProbe:`
			`  httpGet:`
			`    path: /health`
			`    port: management`
			`  initialDelaySeconds: 30`
			`  timeoutSeconds: 3`
			`  periodSeconds: 10`

services.environment.ports.service	Specifies the service port for Environment Service.	Number	8080
services.environment.replicas	The number of Environment Service pod replicas. This value overrides services.replicas.	Number	nil

services.environment.resources	The requests and limits for Environment Service pod resources. See the Kubernetes documentation for details.	Object	`requests:`
			`  cpu: 500m`
			`  memory: 4Gi`
			`limits:`
			`  cpu: "4"`
			`  memory: 6Gi`

Parameter	Description	Valid values	Default
services.environment.serviceAnnotations	Annotations for Authentication Service service objects. See the Kubernetes documentation for details.	A valid set of annotations as "name: value"	{}
services.secret.admin_password	Encrypted password of the operational user. The password should be encrypted with bcrypt hashing with any number of rounds. You can generate an encrypted password on the following site: https://www.javainuse.com/onlineBcrypt	A valid password	nil
services.secret.admin_username	The username of an operational user.	Any valid username. For example, opsAdmin, clientAdmin, ops.	nil
services.secret.client_id	The ID of an encrypted client secret.	Any valid client ID. For example, external_api_client, nexus_client, authclient.	nil
services.secret.client_secret	The encrypted client secret. The client secret should be encrypted with bcrypt hashing with any number of rounds. You can generate an encrypted client secret on the following site:https://www.javainuse.com/onlineBcrypt	A valid client secret	nil
services.secret.create	Specifies whether to create the Kubernetes secret with the credentials of the operational user and client ID.	true or false	true
services.secret.name_override	The name of the secret.	A valid name	nil
services.secrets.secretProviderClassNames.admin_user	The name of the secretProviderClass with the operational user credentials.	A valid class name	"keyvault-gauth-admin-user"
services.secrets.secretProviderClassNames.client_credentials	The name of the secretProviderClass with the client credentials.	A valid class name	"keyvault-gauth-client-credentials"
services.secrets.secretProviderClassNames.cons	The name of the secretProviderClass with	A valid class name	"keyvault-consul-consul-gauth-token"

Parameter	Description	Valid values	Default
ul_token	the Consul token.		
services.secrets.secretProviderClassNames.jks_credentials	The name of the secretProviderClass with the JKS credentials.	A valid class name	"keyvault-gauth-jks-credentials"
services.secrets.secretProviderClassNames.jks_keyvault	The name of the secretProviderClass with the JKS keystore.	A valid class name	"keyvault-gauth-jks-keyvault"
services.secrets.secretProviderClassNames.pg_user	The name of the secretProviderClass with PostgreSQL credentials.	A valid class name	"keyvault-gauth-pg-user"
services.secrets.secretProviderClassNames.redis_password	The name of the secretProviderClass with the Redis password.	A valid class name	"keyvault-gauth-redis-password"
services.secrets.useSecretProviderClass	Specifies whether to read secrets from the secretProviderClass instead of Kubernetes secrets.	true or false	false
topologySpreadConstraints	In Kubernetes, topology spread constraints are used to control how Pods are spread across the cluster among failure-domains such as regions, zones, nodes, and other user-defined topology domains. This helps to achieve high-availability as well as efficient resource utilization.	Valid topology spread constraints settings. See the Kubernetes documentation for details.	{}
services.service_auth.env.GWS_AUTH_common_configService	The URL of the GWS Service Configuration service (config server cache). Used by the authentication service to read user details, validate credentials, and manage passwords.This is a mandatory setting	A valid HTTP URL pointing to the configuration service (e.g., http://gws-service-configuration:8892 or an internal load balancer URL)	None (must be set)
services.service_auth.env.GWS_AUTH_filterCfgObjects	When enabled, the authentication service filters user authorities (roles and access groups) to return only those assigned to the specific person. This prevents incorrect RBAC behavior caused by cache synchronization delays, where userInfo could temporarily return	true or false	false

Parameter	Description	Valid values	Default
	authorities for all roles/ access groups instead of only assigned ones. Recommended to set to "true" for environments using gws service configuration (config server cache).		
services.useNewAuth	Controls which auth service the main chart's ingress routes traffic to. When `true`, routes /auth/ to `new auth (gauth-service-authentication)`. When `false`, routes to `old auth (gauth-auth)`. **Note:** If on-premise customers also use the gauth-infra-bg chart, this value must be set in sync with active.useNewAuth in that chart to avoid conflicting ingress routing	true or false	false
services.service_auth.externalAuth.enabled	Deploys dedicated new auth ext pods (gauth-service-auth-ext) for handling /auth/v3/oauth/token traffic separately. Pods are deployed but only receive traffic when `services.useNewAuth: true`. **Note:** For on-premise customers using gauth-infra-bg, also set `active.externalAuth: true` in that chart.	true or false	false
services.auth.externalAuth.enabled	Deploys dedicated old auth ext pods (gauth-auth-ext) for handling /auth/v3/oauth/token traffic separately. Pods are deployed but only receive traffic when `services.useNewAuth: false`. **Note:** For on-premise customers using gauth-infra-bg, also set `active.externalAuth: true` in that chart.	true or false	false

Parameter	Description	Valid values	Default
active.useNewAuth	Controls which auth service the infra-bg chart's ingress routes traffic to. When `true`, routes /auth/ to `new auth (gauth-service-authentication)`. When `false`, routes to `old auth (gauth-auth)`. **Note:** Must be set in sync with services.useNewAuth in the gauth chart to avoid conflicting ingress routing.	true or false	false
active.externalAuth	When `true`, creates a separate ingress path for /auth/v3/oauth/token routing to the dedicated ext pods. The target (old ext or new ext) is determined by `active.useNewAuth`. **Note:** Corresponding ext pods must be deployed via services.service_auth.externalAuth.enabled or services.auth.externalAuth.enabled in the gauth chart.	true or false	true

Provision Consul-less Deployment

To perform Consul-less deployment, the customer should use Helm chart **gauth-100.0.100+0250** or newer and remove all Consul related parameters from the override file:

1. `consul` section.
2. `services.secrets.secretProviderClassNames.consul_token` parameter.
3. `auth_ui.consul` section.

If deployment with Consul is desired, charts **gauth-100.0.016+0247** from the release **100.0.016.4359** can be used to deploy newer versions of GAuth components.

Configure Kubernetes

The sections below provide more information about configuring Kubernetes.

ConfigMaps

Genesys Authentication includes separate ConfigMaps for PostgreSQL and Redis configuration.

PostgreSQL - configmap-pg.yaml

```
{{- if or .Values.postgres.configmap.create .Values.postgres.deploy }}
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ include "configmap.postgres" . }}
  namespace: {{ .Release.Namespace | quote }}
  labels:
    {{- include "gauth.labels" . | nindent 4 }}
    gauth: postgres
data:
  db: {{ required "Missing required parameter 'postgres.password'" .Values.postgres.db |
quote}}
  host: {{ default ( include "name.postgres" . ) .Values.postgres.host |quote}}
  port: {{ default ( include "port.postgres.service" . ) .Values.postgres.port |quote }}
  {{- end }}
```

Redis - configmap-redis.yaml

```
{{ if or .Values.redis.configmap.create .Values.redis.deploy }}
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ include "configmap.redis" . }}
  namespace: {{ .Release.Namespace | quote }}
  labels:
    {{- include "gauth.labels" . | nindent 4 }}
    gauth: redis
data:
  cluster_nodes: {{ default ( include "service.redis" . ) .Values.redis.cluster_nodes |
quote}}
  {{end}}
```

Secrets

The following Genesys Authentication services artifacts are stored as Kubernetes secrets:

- Administrator user credentials for the Authentication API and Environment API services.
- OAuth 20 client IDs and client secrets for the Authentication API and Environment API services.
- PostgreSQL database credentials for the Environment API service.
- PostgreSQL database credentials for the Authentication API service.
- Java keystore password for Authentication API service.
- Credentials for access to a password-protected Redis (Access Key) for the Authentication API service.

Configure security

To learn more about how security is configured for private edition, be sure to read the Permissions topic in the *Setting up Genesys Multicloud CX Private Edition* guide.

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.

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

Deploy Genesys Authentication

Contents

- 1 Assumptions
- 2 Prepare your environment
  - 2.1 GKE
  - 2.2 AKS
- 3 Deploy
- 4 Configure external access
  - 4.1 Provision ingresses for GKE or AKS
- 5 Validate the deployment
- 6 Genesys Service Authentication Cut Over Guide
  - 6.1 How Cutover Works
  - 6.2 Verification:
  - 6.3 Prerequisites
  - 6.4 Auth Traffic Routing Tables
  - 6.5 Helm Values Reference:
  - 6.6 Step 1: Deploy gauth chart with both old and new auth services
  - 6.7 Step 2: Verify new auth service health
  - 6.8 Step 3: Pre-cutover functional test
  - 6.9 Step 4: Cutover - Switch traffic to new auth
  - 6.10 Step 5: Validate cutover
  - 6.11 Step 6: Restart Only the Applicable Services
  - 6.12 Step 7: Rollback (if needed)
  - 6.13 Step 8: Cleanup (after successful cutover)
  - 6.14 Additional Cutover Steps for Customers Using gauth-infra-bg Chart
  - 6.15 Step 4 becomes:
  - 6.16 Rollback (Step 7) becomes:

Learn how to deploy Genesys Authentication 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 Genesys Authentication.

Prepare your environment

To prepare your environment for the deployment, complete the steps in this section for Google Kubernetes Engine (GKE).

GKE

Log in to the GKE cluster from the host where you will run the deployment:

```
gcloud container clusters get-credentials
```

Create a new namespace for Genesys Authentication with a JSON file that specifies the namespace metadata. For example, **create-gauth-namespace.json**:

```
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "gauth",
    "labels": {
      "name": "gauth"
    }
  }
}
```

Execute the following command to create the namespace:

```
kubectl apply -f create-gauth-namespace.json
```

Confirm the namespace was created:

```
kubectl describe namespace gauth
```

AKS

Log in to the AKS cluster from the host where you will run the deployment:

```
az aks get-credentials --resource-group  --name  --admin
```

Create a new namespace for Genesys Authentication with a JSON file that specifies the namespace metadata. For example, **create-gauth-namespace.json**:

```
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "gauth",
    "labels": {
      "name": "gauth"
    }
  }
}
```

Execute the following command to create the namespace:

```
kubectl apply -f create-gauth-namespace.json
```

Confirm the namespace was created:

```
kubectl describe namespace gauth
```

Deploy

To deploy Genesys Authentication, you'll need the Helm package and your overrides file. Copy **values.yaml** and the Helm package (**gauth-.tgz**) to the installation location.

For debugging purposes, use the following command to render templates without installing so you can check that resources are created properly:

```
helm template --debug /gauth-.tgz -f values.yaml
```

The result shows Kubernetes descriptors. The values you see are generated from Helm templates, and based on settings from **values.yaml**. Ensure that no errors are displayed; you will later apply this configuration to your Kubernetes cluster.

Now you're ready to deploy Genesys Authentication:

```
helm install gauth ./gauth-.tgz -f values.yaml -n gauth
```

Configure external access

Follow the instructions for either GKE or AKS to make the Genesys Authentication services accessible from outside the cluster.

Provision ingresses for GKE or AKS

After deploying, make Genesys Authentication services accessible from outside the GKE or AKS cluster using the NGINX Ingress Controller.

Create a YAML file called **gauth-ingress.yaml** with the content below. **Note:** Replace **gws.** and **gauth.** with your GWS and Genesys Authentication domains, such as `gws.test.dev`.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gauth-gws-ingress
  namespace: gauth
  annotations:
    # add an annotation indicating the issuer to use.
    cert-manager.io/cluster-issuer: "selfsigned-cluster-issuer"
    # Custom annotations for NGINX Ingress Controller
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/use-regex: "true"
spec:
  rules:
  - host: gws. - e.g. gws.test.dev
    http:
      paths:
        - path: /ui/auth/.*
          backend:
            serviceName:  gauth-auth-ui
            servicePort: 80
        - path: /auth/.*
```

```
          backend:
            serviceName:  gauth-auth
            servicePort: 80
        - path: /environment/.*
          backend:
            serviceName:  gauth-environment
            servicePort: 80
  tls:
  - hosts:
    - gws. - e.g. gws.test.dev
    secretName: gauth-gws-ingress-cert
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gauth-gauth-ingress
  namespace: gauth
  annotations:
    # add an annotation indicating the issuer to use.
    cert-manager.io/cluster-issuer: "selfsigned-cluster-issuer"
    # Custom annotations for NGINX Ingress Controller
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/use-regex: "true"
spec:
  rules:
  - host: gauth. - e.g. gauth.test.dev
    http:
      paths:
        - path: /ui/auth/.*
          backend:
            serviceName:  gauth-auth-ui
            servicePort: 80
        - path: /auth/.*
          backend:
            serviceName:  gauth-auth
            servicePort: 80

        - path: /environment/.*
          backend:
            serviceName:  gauth-environment
            servicePort: 80
  tls:
  - hosts:
    - gauth. - e.g. gauth.test.dev
    secretName: gauth-gauth-ingress-cert
```

Create ingresses with the following command:

```
kubectl apply -f gauth-ingress.yaml -n gws
```

Validate the deployment

Check the installed Helm release:

```
helm list
```

The results should show the Genesys Authentication deployment details. For example:

```
NAME    NAMESPACE       REVISION        UPDATED
STATUS          CHART           APP VERSION
gauth   gauth           1               2021-05-20 11:56:32.5531685 +0530 +0530
deployed        gauth-0.1.77    0.1
```

Check the **gauth** namespace status:

```
helm status gauth
```

The result should show the namespace details with a status of deployed:

```
NAME: gauth
LAST DEPLOYED: Thu May 20 11:56:32 2021
NAMESPACE: gauth
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Check the Genesys Authentication Kubernetes objects created by Helm:

```
kubectl get all -n gauth
```

The result should show all the created pods, service ConfigMaps, and so on.

Finally, verify that you can now access Genesys Authentication at the following URL: https:///ui/auth/sign-in.html

Genesys Service Authentication Cut Over Guide

How Cutover Works

The new auth service (`gauth-service-authentication`) uses the same Postgres database and Redis cluster as the old auth service (`gauth-auth`), but with different table formats and token storage formats.

Database (Postgres)

data	Old Auth table name	New Auth Table name
Clients	oauth_client_details	oauth2_client_details
Contact Centers	client_contact_centers	client_contact_centers2

Automatic migration: When the new auth service starts with `DB_INIT_DB=true` (default), Flyway runs migration `V1.7.1__CopyOauth2ClientDetailsFromOldTable.sql` which copies all client data from old tables to new tables. This happens once on first startup. Subsequent restarts skip it (already applied).

In our helm charts, its enabled by default

```
services:
  # Switch between OLD AUTH (false) and NEW AUTH (true)
  # Set to true to use gauth-service-authentication-eks
  # Set to false to use gauth-auth-eks (default)
  useNewAuth: false

  secrets:
    useSecretProviderClass: false
  # Number of pod replicas, recommended to be N+1
  replicas: 3
  location: /
  db:
    init: true # It enables automatic migration
    poolSize: 3
```

Verification:

1.Check new auth pod logs for:

```
kubectl logs -n gauth  | grep -i "migrat"
```

i. Expected output (first startup):

```
Migrating schema "public" to version "1.7.1 - CopyOauth2ClientDetailsFromOldTable"

Successfully applied 1 migration(s) to schema "public"
```

ii. Or if already migrated (subsequent restarts):

```
Schema "public" is up to date. No migration necessary.
```

2. Also you can verify client data via the operations API on the new auth pod:

```
kubectl port-forward -n gauth  8080:8080
curl -u : http://localhost:8080/auth/v3/ops/clients
```

If clients are returned, the migration was successful.

Redis (Token Storage)

	Old Auth	New Auth
Serialization	JDK binary	Jackson JSON
Key prefix	configurable (empty)	GWS:Auth:V3:
Token keys	plain token values	SHA-256 hashed
Data model	OAuth2AccessToken + OAuth2Authentication	OAuth2Authorization (unified)

Tokens are NOT migrated:

Old tokens stored by old auth cannot be read by new auth due to incompatible serialization formats.

See Step 5 and Step 6 for post-cutover actions required.

Rollback Considerations

If you rollback to old auth (`useNewAuth: false`):

- Tokens created by new auth become invalid (same incompatibility in reverse)
- Dependent services must be restarted again
- Users must re-login again

Prerequisites

- gauth Helm chart version >= 100.0.101+0258 (includes new auth service support)
- gauth-infra-bg Helm chart version >= 100.0.101+36 (only if you use this chart)

Auth Traffic Routing Tables

i.Customers (using gauth chart only)

Single chart manages both pods and ingress. No dependency on gauth-infra-bg.

services.useNewAuth	services.service_auth.externalAuth.enabled	/auth/ routes to	/auth/v3/oauth/token routes to
false	false	Old auth	Old auth (same pods)
false	N/A (services.auth.externalAuth.enabled: true)	Old auth	Old auth ext (dedicated pods)
true	false	New auth	New auth (same pods)
true	true	New auth	New auth ext (dedicated pods)

ii. Customers (using both gauth + gauth-infra-bg charts)

Both charts create ingresses. Values must be kept in sync to avoid conflicting routing.

services.useNewAuth (gauth)	active.useNewAuth (infra-bg)	active.externalAuth (infra-bg)	/auth/ routes to	/auth/v3/oauth/ token routes to
false	false	false	Old auth	Old auth (same pods)
false	false	true	Old auth	Old auth ext (dedicated pods)
true	true	false	New auth	New auth (same pods)
true	true	true	New auth	New auth ext (dedicated pods)

Important: Always set services.useNewAuth and active.useNewAuth to the same value. Mismatched values will result in two ingresses pointing to different auth services, causing unpredictable routing behavior.

Helm Values Reference:

Parameter	Description	Valid Values	Default
services.useNewAuth	Controls which auth service the main chart's ingress routes traffic to. When `true`, routes /auth/ to `new auth (gauth-service-authentication)`. When `false`, routes to `old auth (gauth-auth)`. **Note:** If on-premise customers also use the gauth-infra-bg chart, this value must be set in sync with active.useNewAuth in that chart to avoid conflicting ingress routing	true or false	false
services.service_auth.externalAuth.enabled	Deploys dedicated new auth ext pods (gauth-service-auth-ext) for handling /auth/v3/oauth/token traffic separately. Pods are deployed but only receive traffic when `services.useNewAuth: true`. **Note:** For on-premise customers using gauth-infra-bg, also set `active.externalAuth: true` in that chart.	true or false	false
services.auth.externalAuth.enabled	Deploys dedicated old auth ext pods (gauth-auth-ext) for handling /auth/v3/oauth/token traffic separately. Pods are deployed but only receive traffic when `services.useNewAuth: false`. **Note:** For on-premise customers using gauth-infra-bg, also set `active.externalAuth: true` in that chart.	true or false	false
active.useNewAuth	Controls which auth service the infra-bg chart's ingress routes	true or false	false

Parameter	Description	Valid Values	Default
	traffic to. When `true`, routes /auth/ to new `auth (gauth-service-authentication)`. When `false`, routes to `old auth (gauth-auth)`. **Note:** Must be set in sync with `services.useNewAuth` in the gauth chart to avoid conflicting ingress routing.		
active.externalAuth	When `true`, creates a separate ingress path for /auth/v3/oauth/token routing to the dedicated ext pods. The target (old ext or new ext) is determined by `active.useNewAuth`. **Note:** Corresponding ext pods must be deployed via services.service_auth.externalAuth.enabled or services.auth.externalAuth.enabled in the gauth chart.	true or false	true

Step 1: Deploy gauth chart with both old and new auth services

By default, both old auth and new auth pods are deployed. Traffic routes to old auth initially.

Update your values file:

```
services:
  useNewAuth: false    # Traffic routed to OLD AUTH
  service_auth:
    externalAuth:
      enabled: true    # Deploy new auth ext pods
    jks:
      enabled: true
      secret:
        create: false
        name: gauth-jks
    env:
      GWS_JAVA_OPTIONS_HEAPSIZE: ""
  auth:
    externalAuth:
      enabled: true    # Keep old auth ext pods running
```

Note: The `externalAuth` sections are only needed if you use dedicated external auth pods for the `/auth/v3/oauth/token` endpoint. If you do not use external auth, omit these sections.

Helm chart default value for new auth service is 100.0.001.**0192**

Update this to latest new auth service version 100.0.001.**0193** in your values.yaml

```
gauth-service-authentication: 100.0.001.0193
```

Deploy:

```
helm upgrade --install gauth ./gauth-.tgz -f .yaml -n gauth --wait --timeout 600s
```

Verify all deployments are running:

```
kubectl get deployments -n gauth
```

Expected:

```
gauth-auth-                    2/2
gauth-service-authentication-  2/2
gauth-environment-             2/2
gauth-auth-ui-                 2/2
```

Note: If you use external auth, you will also see `gauth-auth-ext-` and `gauth-service-auth-ext-` deployments.

Step 2: Verify new auth service health

Check that new auth pods are running and healthy:

```
kubectl get pods -n gauth -l gauth=service-auth
kubectl exec -n gauth  -- curl -s http://localhost:8081/health
```

Expected: {"status":"UP"} Verify service started successfully in logs:

```
kubectl logs -n gauth  | grep -i "Started"
```

Expected log message indicating successful startup:

```
Started AuthorizationServerApplication in X seconds
```

If health check fails, check logs for errors:

```
kubectl logs -n gauth  --tail=100
```

Step 3: Pre-cutover functional test

Before switching traffic, verify new auth functionality by calling the token endpoint directly via port-forward:

```
kubectl port-forward -n gauth  8080:8080
```

Then test:

```
curl -X POST http://localhost:8080/auth/v3/oauth/token \
  -d "grant_type=client_credentials&client_id=&client_secret="
```

Expected: A valid JSON response with `access_token`, `token_type`, and `expires_in` fields. If the response contains an error, check the pod logs for details before proceeding with cutover.

Step 4: Cutover - Switch traffic to new auth

Update `gauth` chart values:

```
services:
  useNewAuth: true  # Switch traffic to new auth
```

Deploy:

```
helm upgrade --install gauth ./gauth-.tgz -f .yaml -n gauth --wait --timeout 600s
```

Verify ingress is updated to point to new auth service:

```
kubectl get ingress -n gauth -o yaml | grep -A5 "path: /auth/"
```

Expected output (confirming traffic routes to new auth):

```
path: /auth/
pathType: ImplementationSpecific
backend:
  service:
    name: gauth-service-authentication-
    port:
      number: 8080
```

If you still see `gauth-auth-` as the backend service name, the cutover has not taken effect.

Step 5: Validate cutover

Test token endpoint via the public URL to confirm traffic is flowing through the new auth service:

```
curl -X POST https:///auth/v3/oauth/token \
  -d "grant_type=client_credentials&client_id=&client_secret="
```

Expected: A valid JSON response with `access_token`, `token_type`, and `expires_in` fields. If the response contains an error, check new auth service logs:

```
kubectl logs -n gauth -l gauth=service-auth --tail=100 -f
```

Note: If you use external auth, also check ext pod logs:

```
kubectl logs -n gauth -l gauth=service-auth-ext --tail=100 -f
```

Step 6: Restart Only the Applicable Services

This restart is required because existing cached tokens become invalid after switching to the new authentication service. These services hold old auth tokens that new auth cannot validate.

Example commands to restart service

```
kubectl rollout restart deployment gws-service-configuration -n gws
kubectl rollout restart deployment gws-app-provisioning -n gws
```

i. gws-service-configuration

ii. gws-app-provisoning

ii. GIR(Genesys Interaction Recording) - Speechminer Web service

iv. Nexus

v. CIWD

vi. UCSX (only Azure)

vii. UDM (CDDS-X)

Impact on Existing Logins

- Existing agent/user sessions will not be terminated automatically.
- However, ongoing or new read/write operations may fail after the cutover because old tokens (stored with JDK binary serialization) cannot be read by the new auth service (which uses Jackson JSON serialization).

Note: Users and agents must log in again to resume normal operations:

Examples:

- WWE users: log out and log back in
- Agent Setup users: log out and log back in

Re-login creates new tokens in the new auth's format. After re-login, all operations work normally.

Step 7: Rollback (if needed)

Switch back to old auth:

```
services:
  useNewAuth: false  # Route traffic back to old auth

helm upgrade --install gauth ./gauth-.tgz -f .yaml -n gauth --wait --timeout 600s
```

Traffic instantly routes back to old auth. Old auth pods are already running, no restart needed.

Note: After rollback, tokens created by new auth become invalid. Dependent services must be restarted again (same as Step 5 and Step 6) and Users must re-login again.

Step 8: Cleanup (after successful cutover)

Once new auth is stable, disable old auth external pods:

```
services:
  useNewAuth: true
  auth:
    externalAuth:
      enabled: false  # Remove old auth ext pods
```

Note: This step is only applicable if you use external auth.

Redeploy gauth chart to remove old auth ext pods. Old auth pods remain for safety but receive no traffic.

Additional Cutover Steps for Customers Using `gauth-infra-bg` Chart

Important: If you deploy the `gauth-infra-bg` chart in addition to the `gauth` chart, both charts create ingress resources that control traffic routing. The `services.useNewAuth` value in the `gauth` chart and the `active.useNewAuth` value in the `gauth-infra-bg` chart **must always be kept in sync**. Mismatched values will result in conflicting ingress rules, causing unpredictable routing behavior.

Step 4 becomes:

1. Update `gauth` chart values:

```
services:
  useNewAuth: true
```

2. Update `gauth-infra-bg` chart values:

```
active:
  useNewAuth: true
  externalAuth: true  # Only if you use external auth
```

3. Deploy both charts:

```
# Deploy gauth chart
helm upgrade --install gauth ./gauth-.tgz -f .yaml -n gauth --wait --timeout 600s

# Deploy gauth-infra-bg chart
helm upgrade --install gauth-infra ./gauth-infra-bg-.tgz -f .yaml -n gauth --wait
```

Rollback (Step 7) becomes:

1. Set both values back to `false`:

```
# gauth chart
services:
  useNewAuth: false
```

```
# gauth-infra-bg chart
active:
  useNewAuth: false
```

2. Deploy both charts again.

```
# Deploy gauth chart
helm upgrade --install gauth ./gauth-.tgz -f .yaml -n gauth --wait --timeout 600s

# Deploy gauth-infra-bg chart
helm upgrade --install gauth-infra ./gauth-infra-bg-.tgz -f .yaml -n gauth --wait
```

Provision Genesys Authentication

Contents

- 1 Prerequisites
- 2 Create a new API Client
- 3 Create an authentication token
- 4 Add a Genesys tenant/environment
- 5 Add a contact center
- 6 Add a data center
- 7 Update CORS settings
- 8 Clean up environments and contact centers
    - 8.1 Prerequisites
    - 8.2 Delete an environment
    - 8.3 Delete a contact center

- Administrator

Learn how to provision Genesys Authentication.

Related documentation:

-
-
-

RSS:

- For private edition

Warning

Provisioning for Genesys Authentication is tied closely with other Genesys services. You must install these services before continuing with the steps on this page:

- Tenant Service

Prerequisites

- You have installed the Genesys Authentication services and the following URLs are accessible:
  - /auth/v3/oauth/token
  - /environment/v3/environments
- You have the ops credentials (**services.secret.admin_username** and **services.secret.admin_password**) from the **values.yaml** file.
- The Tenant Service is accessible.
- You have Tenant Configuration Server service details such as hostname or IP, port, username, password, and cloud application name.

Create a new API Client

Make a POST request to create a new API client for Genesys Authentication:

```
curl --location --request POST '/auth/v3/ops/clients' \
--header 'Content-Type: application/json' \
--user ops:ops \ ---------- Cloud ops credentials () from values.yaml. The default value is
```

```
ops:ops
--data-raw '{"data": {
    "name": "external_api_client", ----------
    "clientType": "CONFIDENTIAL",
    "internalClient": true,
    "refreshTokenExpirationTimeout": 43200,
    "client_id": "external_api_client", ----------
    "client_secret": "", ----------
    "authorities": ["ROLE_INTERNAL_CLIENT"],
    "scope": ["*"],
    "authorizedGrantTypes": ["client_credentials", "authorization_code", "refresh_token",
"password"],
    "redirectURIs": ["https://gauth.","https://wwe.","https://gws.","https://prov."],
---------- should add gws/prov external URLS here
    "accessTokenExpirationTimeout": 43200
    }
}'
```

The result includes the **client_id** you need to Create an authentication token:

```
"status": {
        "code": 0
    },
    "data": {
        "clientType": "CONFIDENTIAL",
        "scope": [
            "*"
        ],
        "internalClient": true,
        "authorizedGrantTypes": [
            "refresh_token",
            "client_credentials",
            "password",
            "authorization_code",
            "urn:ietf:params:oauth:grant-type:token-exchange",
            "urn:ietf:params:oauth:grant-type:jwt-bearer"
        ],
        "authorities": [
            "ROLE_INTERNAL_CLIENT"
        ],

        "redirectURIs": [
            "https://gauth.",
            "https://gws.",
            "https://prov.",

        ],
        "accessTokenExpirationTimeout": 43200,
        "refreshTokenExpirationTimeout": 43200,
        "createdAt": 1619796576236,
        "name": "external_api_client",
        "client_id": "external_api_client",
        "client_secret": "secret",
        "encrypted_client_secret": "A34BOmXDedZwbTKrwmd4eA=="
    }
}
```

Create an authentication token

Make the following POST request to create an authentication token:

```
curl --location --user external_api_client:secret --request POST '/auth/v3/oauth/token' \
---------- user is the API client created in the previous step
--data-urlencode 'username=ops' \
--data-urlencode 'client_id=external_api_client' \ ---------- The client ID created in the
previous step
--data-urlencode 'grant_type=password' \
--data-urlencode 'password=ops'
```

The result includes the **access_token** you need to Add a Genesys tenant/environment:

```
{
    "access_token": "5f1ecb33-5c63-4606-8e30-824e494194c6",
    "token_type": "bearer",
    "refresh_token": "f0c7eed6-cc55-426f-9594-7ae14903e749",
    "expires_in": 43199,
    "scope": "*"
}
```

Add a Genesys tenant/environment

Warning

Complete this step after installing the Tenant service.

Make the following POST request to create the Environment tenant:

```
curl --location --request POST '/environment/v3/environments' \

--header 'Content-Type: application/json' \
--header 'Authorization: Bearer f3aa2109-8889-4182-b2b7-d86917c53e4e' \ ----- access token
generated in previous step
--data-raw '{
    "data": {
        "id" :  ---------- Tenant Service UUID you configured as part of the Tenant Service
deployment
        "username": "default", ---------- Tenant Configuration Server service username
        "password": "password", ---------- Tenant Configuration Server service password
        "connectionProtocol": "addp",
        "remoteTimeout": 7,
        "appName": "Cloud", ---------- Cloud app
        "traceMode": "CFGTMBoth",
        "tlsEnabled": false,
        "configServers": [{
            "primaryPort": 8888, ---------- Tenant Configuration Server service port
            "readOnly": false,
            "primaryAddress":
"tenant-839fa06b-6cbd-4af3-97d0-b579b850c4ec.voice.svc.cluster.local", ---------- Tenant
Configuration Server service URI
            "locations": "/USW1"

        }],
        "localTimeout": 5,
        "tenant": "Environment"
    }
}'
```

The result includes the environment ID you need to Add a contact center:

```
{
    "status": {
        "code": 0
    },
    "path": "/environments/d0fb6386-236c-4739-aec0-b9c1bd6173df" ---------- Environment ID
}
```

Add a contact center

Warning

Complete this step after installing the Tenant service.

Make the following POST request to add a contact center to the environment:

```
curl --location --request POST '/environment/v3/contact-centers' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer 9901f8d6-0351-47f8-b718-7db992f53a02' \
--data-raw '{
    "data": {

        "domains": [

        ],
        "environmentId": "343dd264-7c26-4f9e-82c5-26baedbcb797", ---------- Environment ID
created in the previous step
        "auth": "configServer",
        "id" :  ---------- Tenant Service UUID you configured as part of the Tenant Service
deployment
    }
}'
```

The result includes the contact center ID (also known as CCID) you will need to provision other Genesys services:

```
{
    "status": {
        "code": 0
    },
    "path": "/contact-centers/ed4c03f3-6275-4419-8b2b-11d14af10655" ---------- Contact center
ID
}
```

Add a data center

Make the following POST request to add a data center:

```
curl --location --request POST '/environment/v3/data-centers' \
--user ops:ops \
--header 'Content-Type: application/json' \
--data '{
    "data": {
        "location": "/usw1", ---------- The region as per Genesys Multicloud CX name
convention
        "entryPoint": , ---------- For the location above
        "readOnly": false/true ---------- This should be false for a primary or writeable
region only, true for all other regions
    }
}'
```

The result should look like this:

```
"status": {
        "code": 0
}
```

Update CORS settings

Make the following request to enable Cross-Origin Resource Sharing (CORS) and add URLs for each service that requires authentication. **Note:** CORS is required for Universal Contact Service.

Updates to the **value** field, which contains the list of URLs that require CORS permission, override any existing records. To preserve the existing records, make a GET request to collect the URLs and then append the new values as a comma separate list in your POST.

```
curl --location --request POST '/environment/v3/contact-centers//settings' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer 3f26790a-6e5b-4dc7-a139-ae78dab2a331' \ ---------- Bearer
token
--data-raw '
{
   "data":{
      "location":"/",
      "name":"cors-origins",
      "shared":"true",
      "value":",," ---------- URLs that require CORS permission
   }
}'
```

Clean up environments and contact centers

Complete the steps in this section to delete an environment and clean up its related objects, including the contact center and any related CORS settings.

Prerequisites

- You have created an API Client.
- The authentication token you generated previously.

Delete an environment

First, get your environments:

```
curl --location --request GET 'https:///environment/v3/environments'  --header
'Authorization: Bearer '
```

The response includes all environments and related information:

```
{
   "status":{
      "code":0
   },
   "data":{
      "genesysEnvironments":[
         {
            "id":"9350e2fc-a1dd-4c65-8d40-1f75a2e080dd",
            "tenant":"Environment",
            "appName":"Cloud",
            "username":"default",
            "password":"password",
            "connectionProtocol":"addp",
            "localTimeout":60,
            "remoteTimeout":90,
            "traceMode":"CFGTMBoth",
            "tlsEnabled":false,
            "configServers":[
               {
                  "primaryAddress":"tenant-9350e2fc-a1dd-4c65-8d40-1f75a2e080dd-
cfgmaster.service.dc1.consul",
                  "primaryPort":8888,
                  "readOnly":false,
                  "locations":"/USW1",
                  "readFromDb":false,
                  "useConfigExporter":false,
                  "initDb":false
               }
            ],
            "proxyPort":0
         },
         {
            "id":"6350e2fc-a1dd-4c65-8d40-1f75a2e080dd",
            "tenant":"Environment",
            "appName":"Cloud",
            "username":"default",
            "password":"password",
            "connectionProtocol":"addp",
            "localTimeout":5,
            "remoteTimeout":7,
            "traceMode":"CFGTMBoth",
            "tlsEnabled":false,
            "configServers":[
               {
                  "primaryAddress":"tenant-9350e2fc-
a1dd-4c65-8d40-1f75a2e080dd.voice.svc.gke2-useast1.gcpe002.gencpe.com",
                  "primaryPort":8888,
                  "readOnly":false,
                  "locations":"/USW1",
                  "readFromDb":false,
                  "useConfigExporter":false,
                  "initDb":false
               }
            ],
            "proxyPort":0
         }
      ]
   }
}
```

Make note of **genesysEnvironments.id** and use this ID to delete the environment:

```
curl --location --request DELETE 'https:///environment/v3/environments/' --header
'Authorization: Bearer '
```

The response:

```
{
   "status":{
      "code":0
   }
}
```

Repeat the delete request for each environment. Confirm all environments are deleted by calling GET /environment/v3/environments again.

Delete a contact center

First, get your contact centers:

```
curl --location --request GET 'https:///environment/v3/contact-centers/' --header
'Authorization: Bearer '
```

The response includes all contact centers and related information:

```
{
   "status":{
      "code":0
   },
   "data":{
      "contactCenters":[
         {
            "id":"9350e2fc-a1dd-4c65-8d40-1f75a2e080dd",
            "environmentId":"9350e2fc-a1dd-4c65-8d40-1f75a2e080dd",
            "domains":[
               "t100"
            ],
            "auth":"configServer"
         },
         {
            "id":"6350e2fc-a1dd-4c65-8d40-1f75a2e080dd",
            "environmentId":"6350e2fc-a1dd-4c65-8d40-1f75a2e080dd",
            "domains":[
               "t200"
            ],
            "auth":"config"
         }
      ]
   }
}
```

Make note of **contactCenters.id** and use this ID to delete the contact center:

```
curl --location --request DELETE 'https:///environment/v3/contact-centers/' --header
'Authorization: Bearer '
```

The response:

```
{
   "status":{
      "code":0
   }
}
```

Repeat the delete request for each contact center. Confirm all contact centers are deleted by calling GET /environment/v3/contact-centers/ again.

Provision SAML-based SSO

Contents

- 1 Prerequisites
- 2 Configure SAML-based SSO
    - 2.1 Configure global settings
    - 2.2 Configure regional settings
    - 2.3 Upload IdP metadata for the region
    - 2.4 Enable SAML
    - 2.5 Settings propagation to secondary regions
    - 2.6 Configure CORS
- 3 Update configuration
- 4 SAML metadata
    - 4.1 IdP metadata
    - 4.2 SP metadata
    - 4.3 Manual metadata entries
- 5 Troubleshooting

Learn how to provision Security Assertion Markup Language-based single sign-on for private edition and mixed mode deployments when you do not have access to Agent Setup.

Related documentation:

-
-
-

RSS:

- For private edition

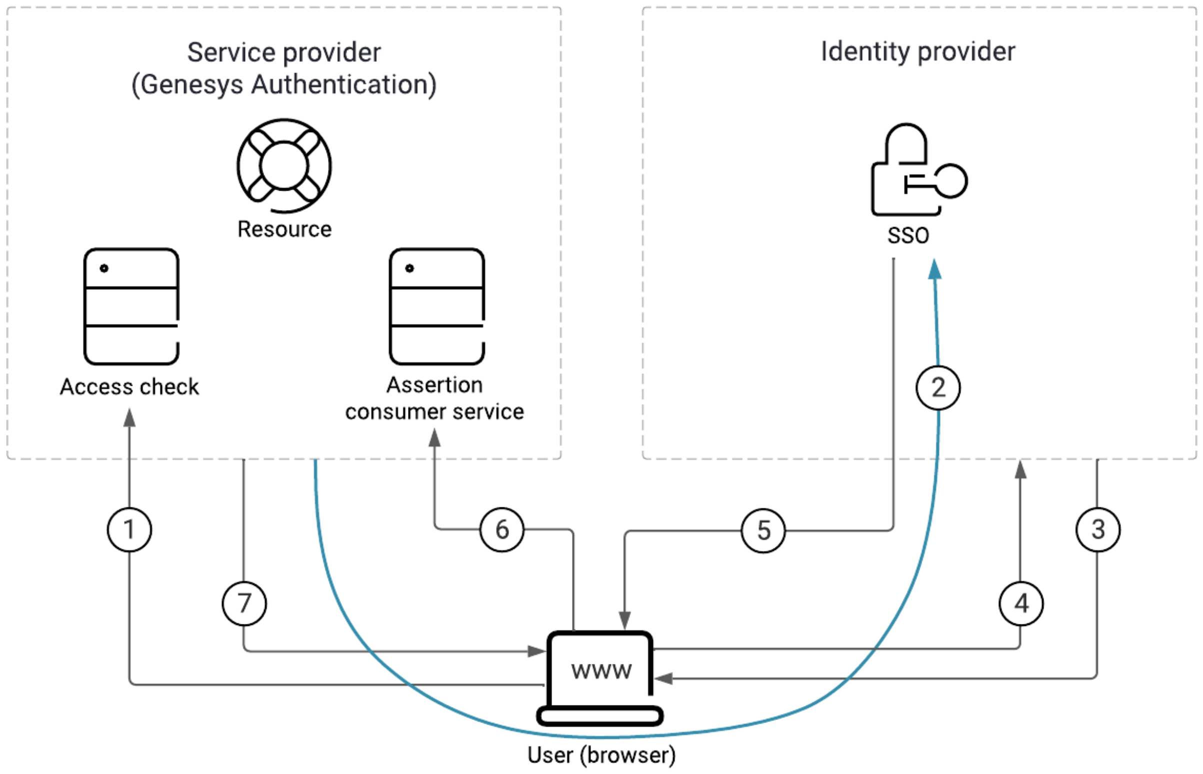
This topic describes how to configure SAML 2.0 single sign-on integration between Genesys Authentication and third-party identity providers (IdP), such as Okta or Google.

Warning

These instructions are for private edition or mixed mode deployments when Agent Setup is not available in your environment. If Agent Setup is available, see Single Sign-On.

Genesys Authentication works as a SAML service provider entity (SP). It accepts authentication assertions according to the SAML protocol and, if the assertion is valid, redirects to the application that started communication. In general, complete this configuration for each region in your deployment where you need SSO integration. However, there are a few global settings that are applicable to all regions—see Configure global settings for details.

The following diagram shows the communication flow for SAML-based SSO. All communication goes through the user's browser and there is no direct traffic or firewall filtering between the SP and the IdP.



Here's a breakdown of the SAML SSO process illustrated in the diagram:

1. The user requests access to a resource.
2. The SP redirects a SAML request to the IdP.
3. The IdP challenges the user for credentials.
4. The user provides the credentials and logs in.
5. The IdP sends a signed SAML response to the browser.
6. The browser posts the SAML response to the SP. Note: This diagram show SAML POST binding, which is selected by default. For a SAML redirect binding, #5 and #6 are merged into one arrow, similar to #2.
7. The SP supplies the resource to the user.

Prerequisites

You must have the following prerequisites to set up SAML-based SSO:

- Genesys Administrator Extension
- The identity provider metadata XML file generated by your IdP server. This file contains configuration

and integration details for SAML SSO. For more information, see SAML metadata.

- The fully qualified domain name URL of your Genesys Authentication deployment. All endpoints in the SP metadata generated by Genesys Authentication use this URL.
- The administrator credentials: services.secret.admin_username and services.secret.admin_password from the **values.yaml** file.
- curl or any REST client.

Configure SAML-based SSO

To configure SAML SSO for your deployment, complete the steps in this section. In the table below, you can find details about the parameters used in the configuration instructions.

Parameter	Description
	The Genesys Authentication internal ingress URL, as configured in internal_ingress.frontend.
	The Genesys Authentication external ingress URL, as configured in ingress.frontend.
	Your contact center ID.
	The deployment region. For example, USW1.
	The user name of the operations administrator, as configured in services.secret.admin_username.
	The password of the operations administrator, as configured in services.secret.admin_password.

Configure global settings

In Genesys Administrator Extension, create an access group for the SSO integration and add users to the group. Genesys recommends that you do the configuration with a test group and test users until you confirm that SSO is working correctly.

Next, configure the access group you want to use for the SSO integration. The **value** can be a comma-separated list.

```
curl -X POST -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /environment/
v3/contact-centers//settings --data '
{"data":
    {
      "name": "samlAuthenticationAccessGroups",
      "location": "/",
      "value": "Test users",
      "category": "saml"
    }
}'
```

If needed, exclude an access group from SSO.

```
curl -X POST -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /environment/
v3/contact-centers//settings --data '
```

```
{"data":
    {
        "name": "internalUserAccessGroups",
        "location": "/",
        "value": "Internal Users,Super Administrators",
        "category": "saml"
    }
}
```

Optional—set the SAML user name option to identify the subject of a SAML assertion. This specifies which attribute in a SAML response is used as the user ID. The default value is .

```
curl -X POST -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /environment/
v3/contact-centers//settings --data '
{
    "data":
    {
        "name":"userNameAttributeKey",
        "location":"/",
        "value":,
        "category": "saml"
    }
}
```

Optional—set the external ID option. If set to true, a user is identified by matching the user name from the SAML response with the **external ID** field from Configuration Server. If false, a user is identified by the **username** field in Configuration Server.

```
curl -X POST -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /environment/
v3/contact-centers//settings --data '
{
    "data":
    {
        "name":"useExternalUserId",
        "location":"/",
        "value":"true",
        "category": "saml"
    }
}'
```

Optional—change the default SSO binding. Currently, Genesys Authentication supports POST (default) and Redirect bindings.

```
curl -X POST -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /environment/
v3/contact-centers//settings  --data '
{
    "data":
    {
        "name":"ssoBinding",
        "location":"/",
        "value":"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect",
        "category": "saml"
    }
}'
```

Configure regional settings

Specify the settings for each region in your deployment. You must have a least one region.

```
curl -X POST -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /environment/
v3/contact-centers//settings --data '
{
    "data":
    {
        "name":"serviceProviderBaseURL",
        "location":,
        "value":,
        "category": "saml"
    }
}'
```

Note: must start with "/". For example, /USW1.

Upload IdP metadata for the region

Some IdP servers, like Okta, require you to submit service provider metadata before they generate IdP metadata. In this case, see SAML metadata before completing the following step.

Once you have the IdP metadata from your identity provider, upload it to Genesys Authentication.

```
curl -X POST -H "Content-Type: text/html" -H 'Authorization: Basic ' -i /environment/v3/
contact-centers//saml/ -d @
```

Note: is the name of your metadata file.

Enable SAML

To enable SAML, first get the data for your contact center.

```
curl  -H 'Authorization: Basic ' -i /environment/v3/contact-centers/
```

The response:

```
{

  "status": {
    "code": 0
  },
  "data": {
    "id": "526af7ee-a71a-44a0-9eea-695eb46478d6",
    "environmentId": "608b741c-99f3-4bb8-8456-4639088aff96",
    "domains": ["somedomain.com"],
    "auth": "configServer"
  }
}
```

Copy the data object and change the value of **auth** to `saml`. Now POST the data back to the server:

```
curl -X PUT -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /environment/v3/
contact-centers/ --data '
{
"data": {
    "environmentId": "608b741c-99f3-4bb8-8456-4639088aff96",
    "domains": ["somedomain.com"],
    "auth": "saml"
  }
}'
```

Settings propagation to secondary regions

In multi-regional deployments, Genesys Authentication data propagates to the secondary region according to the data replication or propagation interval.

Configure CORS

Make sure to configure CORS settings to allowlist your IdP server endpoint URL. See Update CORS settings for details.

Update configuration

You can update configuration by following the steps in Configure SAML-based SSO and then reloading the configuration.

```
curl -X POST -H 'Content-Type: application/json' -H 'Authorization: Basic ' -i /auth/v3/ops/
saml/contact-centers/ --data '
{
  "data":
     {
        "operation":"refresh"
     }
}'
```

SAML metadata

Genesys Authentication works with two kinds of SAML metadata:

- Identity provider (IdP) metadata
- Service provider (SP) metadata

IdP metadata

IdP metadata is a prerequisite to configure SAML-based SSO with Genesys Authentication. Some IdP servers (Okta, for example) might require you to submit SP metadata before they can generate IdP metadata. In this case, you must upload the IdP metadata to the Genesys Authentication service later in the configuration.

Make sure your IdP metadata is up to date with any changes that might affect communication between Genesys Authentication and the IdP server. For example, if you change to a different IdP or a certificate expires for your existing IdP.

Genesys stores IdP metadata as a plain text file in the Web Services and Applications Configuration database.

For example:

```
/environment/v3/contact-centers//saml/ -u :
```

SP metadata

You usually don't need the SP metadata. Retrieve it only when it is required to generate IdP metadata AND you don't want to supply metadata entries to the IdP manually.

Genesys Authentication generates SP metadata automatically when configuration is successful for a particular region. You can access SP metadata as follows:

```
/auth/v3/saml/metadata/alias/sp---0
```

Manual metadata entries

To supply metadata entries to the IdP manually, you need the following information:

- The SP entity ID, also known as the Audience or Reference URI. This is the unique identifier of the service provider. For Genesys Authentication, you can calculate this ID as `sp---0`. Here's an example with a CCID of d49eab9b-ac85-4ad7-b9db-4197e6bc8020 and the region as USW1: `sp-d49eab9b-ac85-4ad7-b9db-4197e6bc8020-USW1-0`
- The single sign-on URL, also known as the AssertionConsumerService URI. For Genesys Authentication, the URL format `/auth/v3/saml/SSO/alias/`. Here's an example with the SP entity ID from the previous step: `https://auth.myexamplecompany.com/auth/v3/saml/SSO/alias/sp-d49eab9b-ac85-4ad7-b9db-4197e6bc8020-USW1-0`
- The single logout URL, also known as the SingleLogoutService URI. For Genesys Authentication, the URL format is `/auth/v3/saml/SingleLogout/alias/`. Here's an example with the SP entity ID from the previous step: `https://auth.myexamplecompany.com/auth/v3/saml/SSO/alias/sp-d49eab9b-ac85-4ad7-b9db-4197e6bc8020-USW1-0`
- The signature certificate, also known as an X509 certificate, from a certificate authority.

Troubleshooting

The first step in troubleshooting SSO issues is to check the SAML settings:

```
curl -X GET -H 'Authorization: Basic ' -i '/environment/v3/contact-
centers//settings?category=saml'
```

If you're seeing errors, particularly intermittent errors, try reloading the configuration after checking the following:

- Make sure the IdP metadata is valid, including valid certificates.
- If the IdP delegates authentication to other entities, make sure that your CORS settings include all fully qualified domain names in the authentication path.

Upgrade, rollback, or uninstall Genesys Authentication

Contents

- 1 Upgrade Genesys Authentication from version 100.0.007.3692
- 2 Upgrade Genesys Authentication
- 3 Rollback Genesys Authentication
- 4 Uninstall Genesys Authentication

Learn how to upgrade, rollback or uninstall Genesys Authentication.

Related documentation:

-
-
-

RSS:

- For private edition

Upgrade Genesys Authentication from version 100.0.007.3692

If you have installed Genesys Authentication version 100.0.007.3692, before upgrading to a new version you must first prepare the PostgreSQL database.

Run the following psql commands:

```
env=# SET search_path TO env;
SET
env=# delete from schema_version where version = '18';
DELETE 1
env=# delete from schema_version where version = '18.1';
DELETE 1
env=# drop table openid_providers;
DROP TABLE
```

Now follow the instructions in Upgrade Genesys Authentication to complete the upgrade to the new version.

Upgrade Genesys Authentication

Genesys Authentication supports a Rolling Update strategy to upgrade its services. To upgrade Genesys Authentication, first Override the Helm chart values.

Next, run the following command to upgrade:

```
helm upgrade -f values.yaml gauth ./gauth
```

Rollback Genesys Authentication

To rollback Genesys Authentication, you can either use the `helm rollback` command or the `helm upgrade` command and specify the previous values.

An example using `helm rollback`:

```
helm rollback gauth
```

An example using `helm upgrade`:

```
helm upgrade -f previous-values.yaml gauth ./gauth
```

Uninstall Genesys Authentication

Use the following command to uninstall Genesys Authentication:

```
helm uninstall gauth
```

Observability in Genesys Authentication

Contents

- 1 Monitoring
  - 1.1 Enable monitoring
  - 1.2 Configure metrics
- 2 Alerting
- 3 Logging

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

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 Genesys Authentication metrics, see:

- Authentication Service metrics
- Environment Service metrics

See also System metrics.

Enable monitoring

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
Authentication Service	Annotations	8081	/prometheus	Real-time
Environment Service	Annotations	8081	/prometheus	Real-time

Configure metrics

The metrics that are exposed by the Genesys Authentication services are available by default. No further configuration is required in order to define or expose these metrics. You cannot define your own custom metrics.

The Metrics pages linked to above show some of the metrics the Genesys Authentication services expose. You can also query Prometheus directly or via a dashboard to see all the metrics available from the Genesys Authentication services.

Alerting

No alerts are defined for Genesys Authentication.

Logging

Genesys Authentication outputs logs to stdout. You can extract these logs using log collectors such as logstash and Elasticsearch. For more information about logging in private edition, see Logging overview and approaches.

You can set the log level for the Authentication Service and the Environment Service using Helm chart parameters:

- services.auth.env.GWS_AUTH_logging_level_com_genesys_gws_v3
- services.environment.env.GWS_ENVIRONMENT_logging_level_com_genesys_gws_v3

Authentication Service metrics and alerts

Contents

- 1 Metrics
- 2 Alerts

Find the metrics Authentication Service exposes and the alerts defined for Authentication Service.

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval
Authentication Service	Annotations	8081	/prometheus	Real-time

See details about:

- Authentication Service metrics
- Authentication Service alerts

Metrics

Authentication Service 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 Authentication Service metrics not documented on this page.

The following system metrics are likely to be most relevant:

- api_requests_seconds_count_total
- api_requests_seconds_sum_total
- jvm_threads_deadlocked
- jvm_gc_pause_seconds_count
- jetty_threads_current
- jvm_memory_used_bytes

Metric and description	Metric details	Indicator of
gws_responses_total The number of responses grouped by HTTP code.	**Unit:** **Type:** Counter **Label:** • Code - The response status code. • Group - The goup of response codes. The values are: 4xx,5xx	

Metric and description	Metric details	Indicator of
	Sample value:	
auth_auth_system_login_errors_total The number of system login errors, excluding expired passwords, incorrect user names and so on.	**Unit:** **Type:** Counter **Label:** • contactCenter – The contact center ID. • environment – The environment ID. **Sample value:**	
psdk_conn_error_total The number of errors that occurred when the Authentication Service connected to Configuration Servers.	**Unit:** **Type:** Counter **Label:** Environment – The environment ID. **Sample value:**	
auth_context_error_total The number of errors during Configuration Server context initialization.	**Unit:** **Type:** Counter **Label:** environment – The environment ID. **Sample value:**	
auth_cs_connection_timeouts_total The number Configuration Server connection timeouts.	**Unit:** **Type:** Counter **Label:** environment – The environment ID. **Sample value:**	
auth_cs_command_timeouts_total The number of Configuration Server command timeouts.	**Unit:** **Type:** Counter **Label:** environment – The environment ID. **Sample value:**	
auth_cs_protocol_errors_total The number of Configuration Server protocol timeouts.	**Unit:** **Type:** Counter **Label:** environment – The environment ID. **Sample value:**	
auth_saml_response_errors The number of Security Assertion Markup Language (SAML) errors.	**Unit:** **Type:** Counter **Label:** contactCenter - The contact center ID. **Sample value:**	

Alerts

The following alerts are defined for Authentication Service.

Alert	Severity	Description	Based on	Threshold
GAUTH-Blue-CPU-Usage	High	A Genesys Authentication pod has CPU usage above 300% during the last 5 minutes.	container_cpu_usage_seconds_total	More than 300% in 5 minutes
GAUTH-Blue-Memory-Usage	High	A Genesys Authentication pod has memory usage above 70% in the last 5 minutes.	container_memory_usage_bytes, container_spec_memory_limit_bytes	More than 70% in 5 minutes
GAUTH-Blue-Pod-NotReady-Count	High	Genesys Authentication has 1 pod ready in the last 5 minutes.	kube_deployment_spec_replicas, kube_deployment_status_replicas_available	1 in 5 minutes
GAUTH-Blue-Pod-Restarts-Count	High	A Genesys Authentication pod has restarted 1 or more times during the last 5 minutes.	kube_pod_container_status_restarts_total	1 or more in 5 minutes
GAUTH-Blue-Memory-Usage-CRITICAL	Critical	A Genesys Authentication pod has memory usage above 90% in the last 5 minutes.	container_memory_usage_bytes	More than 90% in 5 minutes
GAUTH-Blue-Pod-Restarts-Count-CRITICAL	Critical	A Genesys Authentication pod has restarted more than 5 times in the last 5 minutes.	kube_pod_container_status_restarts_total	More than 5 in 5 minutes
GAUTH-Blue-Pods-NotReady-CRITICAL	Critical	Genesys Authentication has 0 pods ready in the last 5 minutes.	kube_deployment_status_replicas_available, kube_deployment_spec_replicas	0 in 5 minutes
auth_jvm_threads_deadlocked	Critical	Deadlocked JVM threads exist.	jvm_threads_deadlocked	0
auth_high_jvm_gc_pause_seconds_count	Critical	JVM garbage collection occurs	jvm_gc_pause_seconds_count	More than 10 in 30 seconds

Alert	Severity	Description	Based on	Threshold
		more than 10 times in the last 30 seconds.		
auth_high_5xx_responces_count	Critical	Genesys Authentication has received more than 10 5xx responses.	gws_responses_total	More than 10
auth_high_500_responces_count	Critical	Genesys Authentication has received more than 10 500 responses.	gws_responses_total	More than 10
auth_auth_login_errors	Critical	Genesys Authentication has received more than 20 login errors for the call center ID in the last 60 seconds.	auth_system_login_errors_total	More than 20 in 60 seconds
auth_total_count_of_errors_in_PSDK_connections	High	Genesys Authentication received more than 3 errors in PSDK connections in the last 30 seconds. A spike might indicate a problem with the backend or a network issue. Check the logs for details.	psdk_conn_error_total	More than 3 in 30 seconds
auth_total_count_of_errors_during_context_initialization	High	Genesys Authentication received more than 10 errors in the last 30 seconds during context initialization. A spike might indicate a network or configuration problem. Check the logs for details.	auth_context_error_total	More than 10 in 30 seconds
auth_saml_response_errors	High	Genesys Authentication received more than 20 SAML errors for the contact center ID in the last 60	auth_saml_response_errors	More than 20 in 60 seconds

Alert	Severity	Description	Based on	Threshold
		seconds.		
auth_saml_timing_errors	High	Genesys Authentication received more than 20 SAML timing errors for the contact center ID in the last 60 seconds.	auth_saml_timing_errors	More than 20 in 60 seconds

Environment Service metrics and alerts

Contents

- 1 Metrics
- 2 Alerts

Find the metrics Environment Service exposes and the alerts defined for Environment Service.

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval
Environment Service	Annotations	8081	/prometheus	Real-time

Metrics

Environment Service 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 Environment Service metrics not documented on this page.

The following system metrics are likely to be most relevant:

- api_requests_seconds_count_total
- api_requests_seconds_sum_total
- jvm_threads_deadlocked
- jvm_gc_pause_seconds_count
- jetty_threads_current
- jvm_memory_used_bytes

Metric and description	Metric details	Indicator of
gws_responses_total Total count of HTTP responses with the specified code.	**Unit:** Number **Type:** Counter **Label:** • Code - The HTTP response code, such as 500, 502, or 401. • Group - The group of the HTTP code, such as 4xx or 5xx. **Sample value:** 23	Errors

Alerts

No alerts are defined for Genesys Authentication.