Universal Contact Service Private Edition Guide

2/24/2026

Table of Contents

Contents

Find links to all the topics in this guide.

Related documentation:

-
-

RSS:

- For private edition

Universal Contact Service (UCS) is a service available with the Genesys Multicloud CX private edition offering.

Overview

Learn more about Universal Contact Service (UCS), its architecture, and how to support high availability and disaster recovery.

- About Universal Contact Service (UCS)
- Architecture
- High availability and disaster recovery

Configure and deploy

Find out how to configure and deploy Universal Contact Service (UCS).

- Before you begin
- Configure UCS
- Provision UCS
- Deploy Universal Contact Service

Upgrade, roll back, or uninstall

Find out how to upgrade, roll back, or uninstall UCS .

- Upgrade, roll back, or uninstall UCS

Operations

Learn how to monitor Universal Contact Service (UCS) with metrics and logging.

- Observability in Universal Contact Service
- *No results* metrics and alerts

About Universal Contact Service (UCS)

Contents

Learn about Universal Contact Service (UCS) and how it works in Genesys Multicloud CX private edition.

Related documentation:

-
-
-

RSS:

- For private edition

Universal Contact Service (UCS) is a highly scalable, available and serviceable cloud service built with usage of PostgreSQL and Elasticsearch.

UCS covers all tenants in the specific region, having dedicated RDS per tenant. To cover N regions, it will be necessary to install N UCS in each of those regions. UCS does not support cross-region synchronization which means that only single region per tenant is supported.

UCS has an auxiliary component **DataSync** that can migrate data from an existing UCS 8.5 RDS for particular tenant to the newly deployed UCS RDS.

Supported Kubernetes platforms

Universal Contact Service is supported on the following cloud platforms:

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

See the Universal Contact Service Release Notes for information about when support was introduced.

Architecture

Contents

Learn about Universal Contact Service architecture

Related documentation:

-
-
-

RSS:

- For private edition

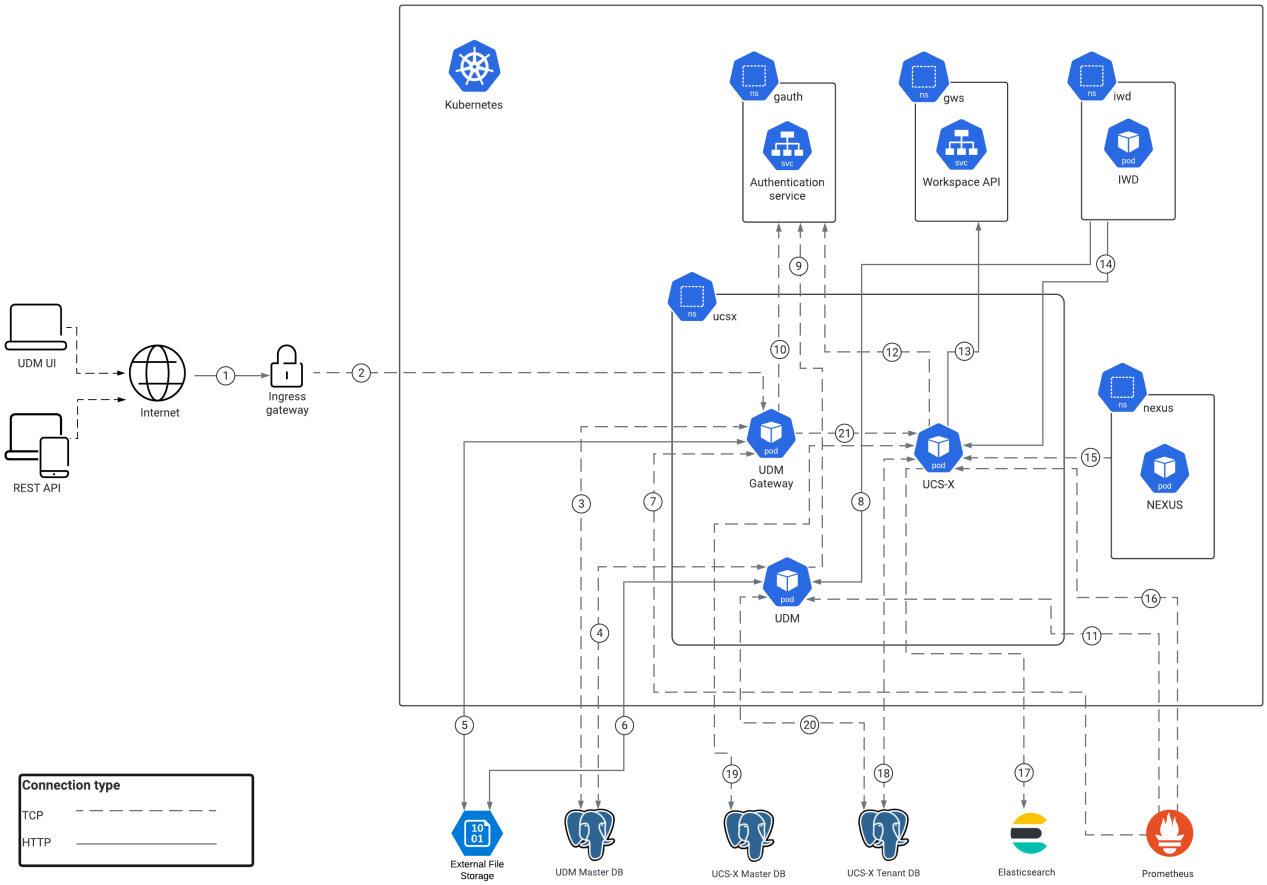
Introduction

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 Universal Contact Service 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 Universal Contact Service as a service in the network. *Egress* means the Universal Contact Service service is the source, and *Ingress* means the Universal Contact Service 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	Browser	Inbound Gateway	HTTP	80	Ingress	Inbound web traffic
2	Ingress Gateway	UDM Gateway	TCP	80	Ingress	Inbound web traffic

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
3	UDM Gateway	UDM Master DB	TCP	5432	Intra-cluster	UDM Gateway reads information about the jobs from the UDM Master DB.
4	UDM	UDM Master DB	TCP	443	Intra-cluster	UDM reads information about the jobs from the UDM Master DB.
5	UDM Gateway	External File Storage	HTTP	443	Intra-cluster	UDM Gateway uploads exported data to the External File Storage.
6	UDM	External File Storage	HTTP	443	Intra-cluster	UDM uploads exported data to the External File Storage.
7	Prometheus	UDM Gateway	TCP	10052	Intra-cluster	Prometheus polls UDM Gateway for metric endpoints.
8	Intelligent Workload Distribution	UDM	HTTP	8080		UDM exports iWD events using the iWD API.
9	UDM	Authentication Service	TCP	80	Intra-cluster	UDM connect to GAUTH for authenticating UDM clients.
10	UDM Gateway	Authentication Service	TCP	80	Intra-cluster	UDM Gatewat connects to GAUTH for authenticating UDM Gateway.
11	Prometheus	UDM	TCP	10052	Intra-cluster	Prometheus

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
						polls UDM for metric endpoints.
12	Universal Contact Service	Authentication Service	TCP	80	Intra-cluster	UCS connects to GAUTH for authenticating UCS-X clients.
13	Universal Contact Service	GWS Workspace Service	HTTP	80	Intra-cluster	Agent Workspace accesses UCS-X through the aggregator (GWS Workspace API). Internal ingress is used to support sticky session for CometD.
14	Intelligent Workload Distribution	Universal Contact Service	HTTP	80	Intra-cluster	iWD stores workitem interactions in UCS-X and iWD also reads contacts from UCS-X.
15	Nexus	Universal Contact Service	TCP	8080	Intra-cluster	Nexus access UCS-X API for storing and reading interactions (chat, socials) and contacts.
16	Prometheus	Universal Contact Service	TCP	10052		Prometheus polls UCS-X for metric endpoints.
17	Universal Contact Service	Elasticsearch	TCP	9200	Intra-cluster	UCS-X logs are passed to Elasticsearch.

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
18	Universal Contact Service	UCSX Tenant Database	TCP	5432	Intra-cluster	UCS-X operations with tenant data (interactions, contacts, categories) are stored in Tenant database.
19	Universal Contact Service	UCSX Master Database	TCP	6432	Intra-cluster	UCS-X stores and accesses configuration data inside the UCS-X Master database: • fetch configuration on instance startup • periodically refresh configuration to get updates in runtime
20	UDM	UCSX Master Database	TCP	5432		UDM exports data from UCS-X Tenant database.
21	UDM Gateway	Universal Contact Service	TCP	443		UDM Gateway retrieves contact information for GDPR related jobs.

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?
Universal Contact Service	N = N (N+1)	Not supported	Primary unit only

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

Before you begin

Contents

Find out what to do before deploying Universal Contact Service (UCS).

Related documentation:

-
-
-

RSS:

- For private edition

Limitations and assumptions

Currently, UCS:

- supports a single-region model of deployment only
- does not support SSL communication with ElasticSearch.
- requires dedicated PostgreSQL deployment per customer.

Download the Helm charts

Download the UCS related Docker containers and Helm charts from the JFrog repository.

See Helm charts and containers for Universal Contact Service for the Helm chart and container versions you must download for your release.

For more information on JFrog, refer to the Downloading your Genesys Multicloud CX containers topic in the *Setting up Genesys Multicloud CX private edition* document.

Third-party prerequisites

- Kubernetes 1.17+
- Helm 3.0

Third-party services

Name	Version	Purpose	Notes
Ingress controller		HTTPS ingress	If UCS is accessed from

Name	Version	Purpose	Notes
		controller.	outside of a K8s cluster, it is recommended to deploy/configure an ingress controller (for example, NGINX), if not already available.
Elasticsearch	7.x	Used for text searching and indexing. Deployed per service that needs Elasticsearch during runtime.	Dedicated (one for UCS cluster) as all indices for all tenants are stored in it. • UCS does not support SSL communication with Elasticsearch.

PostgreSQL	11.x	Relational database.	Dedicated for master DB and for each tenant DB due to high load.
			• DB users must have super user privileges or the following PostgreSQL extensions must be installed before deploying UCS-X service:
			  • pg_prewarm
			  • intarray

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.	You can use any Docker OCI compliant registry.
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.	For UCS, a single regional Google external network LB will be setup as part of the platform setup.

Storage requirements

All data are stored in the PostgreSQL, Elasticsearch, and Nexus Upload Service which are external to the UCS.

Genesys dependencies

UCS requires the following Genesys components:

- Genesys Authentication Service
- GWS Environment Service

GDPR support

As a part of GDPR compliance procedure, the customer would send a request to Care providing the information about the end user. Care would then open a ticket for Engineering team to follow up on the request.

The engineering team would process the request:

GDPR request: Export Data

- Request to UCS to get contact by ID: identify contact (if there is email address or phone number), or `getContact` (if there is a direct contact ID).
- Request to UCS-X to get list of interactions for contact found.
- Perform CSV export and attach resulting file to the ticket.

GDPR request: Forget me

- Request to UCS-X to get contact by ID: identify contact (if there is email address or phone number), or `getContact` (if there is a direct contact ID).
- Request to UCS-X to get list of interactions for contact found.
- Delete all found interactions.
- Re-check that all interactions for contact were removed.
- Delete contact.
- Re-check that contact was removed.
- Update the ticket.

Configure UCS

Contents

Learn how to configure Universal Contact Service (UCS).

Related documentation:

-
-
-

RSS:

- For private edition

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

Create a file named **values.yaml** and set the following values depending on your environment.

- Set the number of running PODs:

```
replicaCount: 2
```

- Set the repository for the images:

```
image:
  registry: base address of UCS image in artifactory.
  repository: ucsx/ucsx
  pullPolicy: IfNotPresent
  pullSecrets: if needed, set to the appropriate value for your environment.
```

- Set the ElasticSearch url, for example, http://ucsx-es-client-service.ucsx.svc.cluster.local:9200:

```
elasticsearch:
  url: Set URL to the ElasticSearch for data
```

- Set the Authentication service information

```
gauth:
  auth:
    url: URL to Genesys Auth service
  env:
    url: URL to GWS Environment service
```

- Set the memory and CPU limits to the values required for your deployment:

```
resources:
  requests:
    memory: "500Mi"
    cpu: "300m"
  limits:
    memory: "1000Mi"
    cpu: "2000m"
```

- Modify the DNS Configuration to match your environment:

```
dnsConfig:
  options:
    - name: ndots
      value: "3"
```

- UCS requires stickiness for some scenarios (from GWS/WWE). You can enable this on the Service level or create ingress rules to enable and configure them.
- The Ingress configuration requires the sticky sessions. You can enable this on the Service level or create ingress rules to enable and configure them. The cookie name should be set to *UCS_SESSIONID*.

Configure Kubernetes

Create a Kubernetes ConfigMap named **ucsx-config** and save the database parameters under the following keys:

- `CMX_MASTER_DB_HOST` - the FQDN of the host where PostgresDB server is running
- `CMX_MASTER_DB_NAME` - the database name
- `CMX_MASTER_DB_PORT` - The port number of the PostgresDB server
- `CMX_MASTER_DB_USER` - the database user

Create a Kubernetes Secret named **ucsx-secret** and save the following secrets under the following keys:

- `CMX_MASTER_DB_PASSWORD` - the password for the database user to access the database
- `CMX_GWS_SERVICE_CREDENTIALS_CLIENT_ID` - the Client ID for GWS Auth
- `CMX_GWS_SERVICE_CREDENTIALS_CLIENT_SECRET` - the Client Secret for GWS Auth

The ConfigMap and Secret can also be created automatically from Helm Chart if the following values are empty:

- `existingSecret` - to create the Secret

- `existingConfig` - to create the ConfigMap

The following values will be added to the Secret:

- `db.password`
- `gauth.auth.clientId`
- `gauth.auth.clientSecret`

The following values will be added to the ConfigMap:

- `db.host`
- `db.name`
- `db.port`
- `db.user`

Configure security

Universal Contact Service (UCS) requires **clientId** and **clientSecret** registered in the Auth Service. These have to be provided during helmchart deployment.

Provision UCS

Contents

- Administrator

Learn how to provision Universal Contact Service (UCS).

Related documentation:

-
-
-

RSS:

- For private edition

UCS provisioning is executed automatically during deployment procedure.

Tenant provisioning

1. Create and configure a **values-override.yaml** file:

```
# * Images
# Replace for your values: registry and secret
image:
  pullPolicy: IfNotPresent
  pullSecrets: [name: ]
  registry:
  repository: deploy_ucsx

# * Command
# "register_ucsx_tenant" or "unregister_ucsx_tenant"
command: "register_ucsx_tenant"

# * Tenant info
tenant:
  id: ""
  ccid: ""

# * UCSX configuration
ucsx:
  addr: ":8500"
  restaddr: "http://:8080"

# * K8s secret and configmap
configNameBase: ucsx-config
secretNameBase: ucsx-secret

# * Authentication
# Set your values.
gauth:
  auth:
    url: "http://"
    # regions:
```

```
    clientId: ""
    clientSecret: ""
  env:
    url: "http://"

# * DB Parameters
# Set your values.
db:
  ssl:  ""
  name: ""
  host: ""
  port: ""
  user: ""
  password: ""

# * WWE
# wwe: {}
wwe:
  skipRegistration: 'true'

# * Pod configuration
podSecurityContext: {}

securityContext: {}

nodeSelector: {}
 priorityClassName: ''
 keepPod: true
```

2. Install the tenant using the `ucsx-addtenant` Helm chart:

```
helm install ucsx-addtenant- ucsxhelmrepo/ucsx-addtenant --version= --namespace=ucsx --wait --
timeout 300s -f values-override.yaml
```

Configure WWE (Workspace API) to consume UCSX service

If the `wwe: skipRegistration` values is set as *true* in the Helm chart,

1. Get a bearer token from the GAUTH service:

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

2. Get the current settings of Workspace API.

```
curl -X GET "https:///environment/v3/contact-centers//settings" -H "Authorization: Bearer "
-H "Content-Type: application/json"
```

3. Configure the Workspace API to consume UCSX service.

```
curl -X POST "https:///environment/v3/contact-centers//settings"
     -H "Authorization: Bearer "
     -H "Content-Type: application/json"
     --data-raw '{
         "data":
            {"name":"workspace-service.ucsservice-uri",
             "location":"/",
             "value":"http://",
```

```
            "shared":false }
      }'
```

4. Configure CORS.

```
curl -X POST "https:///environment/v3/cors"
     -H "Authorization: Bearer "
     -H "Content-Type: application/json"
     --data-raw '{
         "data":
            {
              "origin": "https://",
              "contactCenterId": ""
            }
      }'
```

Validate the provisioning

To check the logs:

```
kubectl get pods ucsx
kubectl logs -pod-id>
```

curl requests

To get a bearer token:

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

To get a cluster version:

```
curl --request POST http:///ucs/v3/rest/request/get-version -H "Authorization: Bearer " -H
"Content-Type: application/json"
```

To get the tenant information:

```
curl -X GET http:///ucs/v3/config/tenants -H "Authorization: Bearer " -H 'Content-Type:
application/json'
```

To show the tenant information using the `cmxctrl` command from the ucsx pod:

```
cmxctrl -g  info
```

Managing Tenants

To unregister a tenant:

```
helm install ucsx-addtenant- ucsxhelmrepo/ucsx-addtenant --version= --namespace=ucsx --wait --
timeout 300s --set command=unregister_ucsx_tenant -f values-override.yaml
```

To uninstall a Helm chart:

```
helm uninstall ucsx-addtenant- --namespace=ucsx
```

To delete a tenant usingthe `cmxctrl` command from the ucsx pod:

```
cmxctrl rm -g
```

Deploy Universal Contact Service

Contents

Learn how to deploy Universal Contact Service (UCS) 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 UCS.

This chart bootstraps an UCS deployment on a Kubernetes cluster using the Helm package manager.

Create a project

Google Kubernetes Engine

1. Log in to the GKE cluster.

```
gcloud container clusters get-credentials gke1
```

2. Create a new manifest for UCS.

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

3. Create the namespace by applying the manifest to the cluster.

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

4. Confirm if the namespace is created.

```
kubectl describe namespace ucsx
```

Azure Kubernetes Service

1. Log in to the AKS cluster.

```
az aks get-credentials --resource-group $RESOURCE_GROUP --name $AKS_CLUSTER_NAME
```

2. Create a new manifest for UCS.

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

3. Create the namespace by applying the manifest to the cluster.

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

4. Confirm if the namespace is created.

```
kubectl describe namespace ucsx
```

Prepare the Helm values file

Create a file named **values-override.yaml** and set the following values depending on your environment.

```
# Default values for ucsx.
# This is a YAML-formatted file.
```

```
# Declare variables to be passed into your templates.

# * Deployment
# Only two possible values are supported: Deployment, ReplicaSet
deploymentType: Deployment

# * Replicacount
replicaCount: 1

# * Images
# Replace for your values: registry and secret
image:
  pullPolicy: IfNotPresent
  pullSecrets: [name: ${DOCKER_REGISTRY_SECRET_NAME}]
  registry: ${DOCKER_SERVER}
  repository: ucsx/ucsx

# * Pod configuration
affinity: {}

nodeSelector: {}

tolerations: []

priorityClassName: ''

podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #   - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

podDisruptionBudget:
  enabled: false
  minAvailable: 1

podLabels: {}

podAnnotations: {}

hpa:
  enabled: false
  targetCPUPercent: 60
  minReplicas: 1
  maxReplicas: 10

resources:
  requests:
    memory: "500Mi"
    cpu: "300m"
  limits:
    memory: "1000Mi"
    cpu: "2000m"

serviceAccount:
  # Specifies whether a service account should be created
  create: false
  # Annotations to add to the service account
```

```
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name:

# * K8s secret and configmap
# If not set it will be created automatically
existingSecret: ucsx-secret
existingConfig: ucsx-config

# * Authentication
# Set your values.
gauth:
  auth:
    url: "http://${GAUTH_AUTH_URL_INTERNAL}:${GAUTH_AUTH_URL_PORT_INTERNAL}"
    clientId: "${GAUTH_CLIENT_ID}"
    clientSecret: "${GAUTH_CLIENT_SECRET}"
  env:
    url: "http://${GAUTH_ENV_URL_INTERNAL}:${GAUTH_ENV_URL_PORT_INTERNAL}"

# * ElasticSearch
# Replace with your values.
elasticsearch:
  url: "http://${ES_URL_INTERNAL}:${ES_PORT_INTERNAL}"

# * DB Parameters
# Set your values.
db:
  ssl:  'false'
  name: "${DB_NAME_SHARED}"
  host: "${DB_HOST_INTERNAL}"
  port: "${DB_PORT_INTERNAL}"
  user: "${DB_USER_SHARED}"
  password: "${DB_PASSWORD_SHARED}"

# * Service
service:
  enabled: true
  name: ucsx
  type: ClusterIP
  externalPort: ${UCSX_ENDPOINT_INTERNAL_EXTERNALPORT}
  env: {}
  annotations: {}

# * Ingress
ingress:
  enabled: true
  annotations: {}
  hosts:
    - host: "${UCSX_ENDPOINT}"
      paths:
        - path: '/ucs/v3/'
          port: ${UCSX_ENDPOINT_INTERNAL_EXTERNALPORT}
        - path: '/ucs/v3/config'
          port: ${UCSX_ENDPOINT_INTERNAL_CONFIGPORT}
  tls: []

# * Monitoring
monitoring:
  # Deploy ServiceMonitor
  enabled: false
  # Create PrometheusRule k8s object with alarm definitions
  alarms: false
```

```
  #  kibanaUrl:
  #  grafanaUrl:
  #  runbookUrl:

# * Network policies
# true or false
networkPolicies:
  enabled: false

# * DNS
dnsConfig:
  options:
    - name: ndots
      value: "3"

# * Healtcheck (Liveness and Readines)
livenessProbe:
  httpGet:
    path: /ucs/v3/healthcheck
    port: rest
  initialDelaySeconds: 5
  timeoutSeconds: 5
  periodSeconds: 30

readinessProbe:
  httpGet:
    path: /ucs/v3/healthcheck
    port: rest
  initialDelaySeconds: 10
  timeoutSeconds: 5
  periodSeconds: 30
```

Deploy the service

Install UCS X:

```
helm upgrade --install ucsx ucsxhelmrepo/ucsx --version= --namespace=ucsx  --set
existingSecret='' --set existingConfig=''-f values-override.yaml
```

Validate the deployment

To validate the install:

```
kubectl get pods -n=ucsx -l "app.kubernetes.io/name=ucsx,app.kubernetes.io/instance=ucsx"
```

Verify that details of the UCS X service deployment information is displayed.

To check the logs:

```
kubectl get pods -n ucsx
kubectl logs
```

curl requests

To get a bearer token:

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

To get a cluster version:

```
curl --request POST http:///ucs/v3/rest/request/get-version -H "Authorization: Bearer " -H
'Content-Type: application/json'
```

To get **/master-dbschema**:

```
curl -X GET http:///ucs/v3/config/master-dbschema -H "Authorization: Bearer " -H 'Content-
Type: application/json'
```

Upgrade, roll back, or uninstall UCS

Contents

Learn how to upgrade, roll back, or uninstall UCS.

Related documentation:

-
-
-

RSS:

- For private edition

> **Important**
>
> The instructions on this page assume you have deployed the services in service-specific 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.

Supported upgrade strategies

Universal Contact Service supports the following upgrade strategies:

Service	Upgrade Strategy	Notes
Universal Contact Service	Rolling Update	

For a conceptual overview of the upgrade strategies, refer to Upgrade strategies in the Setting up Genesys Multicloud CX Private Edition guide.

Timing

A regular upgrade schedule is necessary to fit within the Genesys policy of supporting N-2 releases, but a particular release might warrant an earlier upgrade (for example, because of a critical security fix).

If the service you are upgrading requires a later version of any third-party services, upgrade the third-party service(s) before you upgrade the private edition service. For the latest supported versions of third-party services, see the Software requirements page in the suite-level guide.

Scheduling considerations

Genesys recommends that you upgrade the services methodically and sequentially: Complete the upgrade for one service and verify that it upgraded successfully before proceeding to upgrade the next service. If necessary, roll back the upgrade and verify successful rollback.

Monitoring

Monitor the upgrade process using standard Kubernetes and Helm metrics, as well as service-specific metrics that can identify failure or successful completion of the upgrade (see Observability in Universal Contact Service).

Genesys recommends that you create custom alerts for key indicators of failure — for example, an alert that a pod is in pending state for longer than a timeout suitable for your environment. Consider including an alert for the absence of metrics, which is a situation that can occur if the Docker image is not available. Note that Genesys does not provide support for custom alerts that you create in your environment.

Preparatory steps

Ensure that your processes have been set up to enable easy rollback in case an upgrade leads to compatibility or other issues.

Each time you upgrade a service:

1. Review the release note to identify changes.
2. Ensure that the new package is available for you to deploy in your environment.
3. Ensure that your existing **-values.yaml** file is available and update it if required to implement changes.

Rolling Update

Rolling Update: Upgrade

Execute the following command to upgrade :

```
helm upgrade --install  -f -values.yaml  -n
```

Tip: If your review of Helm chart changes (see Preparatory Step 3) identifies that the only update you need to make to your existing **-values.yaml** file is to update the image version, you can pass the image tag as an argument by using the `--set` flag in the command:

```
helm upgrade --install  -f -values.yaml  --set .image.tag=
```

For example, `helm upgrade -f ./values.yaml ucsx ucsx-100.x.tgz`

Rolling Update: Verify the upgrade

Follow usual Kubernetes best practices to verify that the new service version is deployed. See the information about initial deployment for additional functional validation that the service has upgraded successfully.

Rolling Update: Rollback

Execute the following command to roll back the upgrade to the previous version:

```
helm rollback
```

or, to roll back to an even earlier version:

```
helm rollback
```

Alternatively, you can re-install the previous package:

1. Revert the image version in the `.image.tag` parameter in the **-values.yaml** file. If applicable, also revert any configuration changes you implemented for the new release.
2. Execute the following command to roll back the upgrade:

   ```
   helm upgrade --install  -f -values.yaml
   ```

   Tip: You can also directly pass the image tag as an argument by using the `--set` flag in the command:

   ```
   helm upgrade --install  -f -values.yaml  --set .image.tag=
   ```

For example, `helm rollback ucsx`

Rolling Update: Verify the rollback

Verify the rollback in the same way that you verified the upgrade (see Rolling Update: Verify the upgrade).

Uninstall

> ### Warning
> Uninstalling a service removes all Kubernetes resources associated with that service. Genesys recommends that you contact Genesys Customer Care before uninstalling any private edition services, particularly in a production environment, to ensure that you understand the implications and to prevent unintended consequences arising from, say, unrecognized dependencies or purged data.

Execute the following command to uninstall :

```
helm uninstall  -n
```

For example, `helm uninstall ucsx`

Observability in Universal Contact Service

Contents

Learn about the logs, metrics, and alerts you should monitor for Universal Contact Service.

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 Universal Contact Service metrics, see:

-

See also System metrics.

- *No results* metrics and alerts

Enable monitoring

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
	ServiceMonitor	10052	ucsx.ucsx.svc.cluster.local:10052/ metrics	30 seconds

Configure metrics

UCS provides internal monitoring metrics through a Prometheus endpoint on port 10052.

Alerting

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

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

For descriptions of available Universal Contact Service alerts, see:

-
- *No results* metrics and alerts

Configure alerts

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

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

Logging

UCS writes logs to **stdout**.

No results metrics and alerts

Contents

Find the metrics UCS exposes and the alerts defined for UCS.

Related documentation:

-
-
-

RSS:

- For private edition

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval
UCS	ServiceMonitorpodAnnotations	10052	ucsx.ucsx.svc.cluster.local:10052/metrics	30 seconds

See details about:

- *No results* metrics
- *No results* alerts

Metrics

Metric and description	Metric details	Indicator of
ucsx_performance Basic metric, created with starting metrics service, cannot be disabled.	**Unit:** Percentage **Type:** Gauge **Label:** metric="cpuUsage" **Sample value:** 1	CPU Usage
ucsx_memory Resident Set Size	**Unit:** MB **Type:** Gauge **Label:** **Sample value:** 1024	
ucsx_http_request_duration_count Total number of all HTTP requests	**Unit:** Integer **Type:** Histogram **Label:** **Sample value:** 100	
ucsx_http_request_duration_sum Total duration of all HTTP requests	**Unit:** Milliseconds **Type:** Histogram	

Metric and description	Metric details	Indicator of
	Label: **Sample value:** 500	
ucsx_elastic_search_count Total number of all ElasticSearch requests	**Unit:** Integer **Type:** Histogram **Label:** **Sample value:** 100	
ucsx_elastic_search_sum Total duration of all ElasticSearch requests	**Unit:** Milliseconds **Type:** Histogram **Label:** **Sample value:** 500	
ucsx_tenantdb_health_status Tenant DB status	**Unit:** Integer **Type:** Gauge **Label:** **Sample value:** 1	
ucsx_elasticsearch_health_status ElasticSearch status	**Unit:** Integer **Type:** Gauge **Label:** **Sample value:** 1	
ucsx_masterdb_health_status Master DB status	**Unit:** Integer **Type:** Gauge **Label:** **Sample value:** 1	
ucsx_overload_protection_count Count of the overload protection events	**Unit:** Integer **Type:** Counter **Label:** **Sample value:** 0	

Alerts

The following alerts are defined for *No results*.

Alert	Severity	Description	Based on	Threshold
ucsx_instance_high_cpu_utilization	warning	Triggered when average CPU usage is more than 80%	ucsx_performance	5 minutes
ucsx_instance_high_memory_usage	warning	Triggered when average CPU usage is more than 800 Mb	ucsx_memory	5 minutes
ucsx_instance_high_http_request_rate	warning	Triggered when	ucsx_http_request_duration_count	30 minutes

No results metrics and alerts

Alert	Severity	Description	Based on	Threshold
		request rate is more than 120 requests per seconds on one UCS-X instance		
ucsx_instance_slow_http_response	critical	Triggered when average http response time > 500 ms	ucsx_http_request_duration_sum, ucsx_http_request_duration_count	5 minutes
ucsx_elasticsearch_slow_processing_time	critical	Triggered when Elasticsearch internal processing time > 500 ms	ucsx_elastic_search_sum, ucsx_elastic_search_count	5 minutes
ucsx_tenantdb_health_status	critical	Triggered when there is no connection to tenant DB	ucsx_tenantdb_health_status	2 minutes
ucsx_elasticsearch_health_status	critical	Triggered when there is no connection to ElasticSearch	ucsx_elasticsearch_health_status	2 minutes
ucsx_masterdb_health_status	warning	Triggered when there is no connection to master DB	ucsx_masterdb_health_status	2 minutes
ucsx_instance_overloaded	warning	Triggered when overload protection rate is more than 0	ucsx_overload_protection_counter	5 minutes