Digital Channels Private Edition Guide

12/15/2025

Table of Contents

Contents

Find links to all the topics in this guide.

Related documentation:

-
-
-

RSS:

- For private edition

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

Overview

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

- About Digital Channels
- Architecture
- High availability and disaster recovery

Configure and deploy

Find out how to configure and deploy Digital Channels.

- Before you begin
- Configure Digital Channels
- Deploy Digital Channels
- Upgrade, rollback, or uninstall Digital Channels

Integrate and provision

Learn how to integrate with the tenant and provision API keys and SMS.

- Provisioning overview
- Pre-configure tenant objects
- Enable a tenant for Digital Channels
- Provision API keys
- Setting up Integration for Inbound and Outbound SMS
- Setting up Integration for Outbound Email Campaigns

Upgrade, roll back, or uninstall

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

- Upgrade, rollback, or uninstall Digital Channels

Configure and deploy AI Connector

Find out how to configure and deploy AI Connector.

- About AI Connector
- Before you begin
- Configure AI Connector
- Deploy
- Provisioning overview

Observability

Learn how to monitor Digital Channels with metrics and logging.

- Observability in Digital Channels
- Digital Channels metrics and alerts

About Digital Channels

Contents

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

Related documentation:

-
-
-
-

RSS:

- For private edition

Digital Channels powers your customer interactions across the chat and SMS channels. It provides a platform that enables you to grow sales, create more targeted marketing campaigns, and deliver exceptional customer service. The Digital Channels service processes, manages and archives customer and agent interactions across media.

Chats and SMS are treated just like regular Genesys interactions. When customers communicate with your company on one of these channels, Genesys matches them against customers already in the contact database. If there's a match, the agent handling the interaction has access to all previous interactions with the contact. Until the interaction is marked Done, agents can also return to the chat conversation at any time in the future — for example, they might need to take time to find additional information for the contact or initiate a business process in your company.

Supported Kubernetes platforms

Digital Channels is supported on the following cloud platforms

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

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

Architecture

Contents

Learn about Digital Channels architecture

Related documentation:

-
-
-
-

RSS:

- For private edition

Introduction

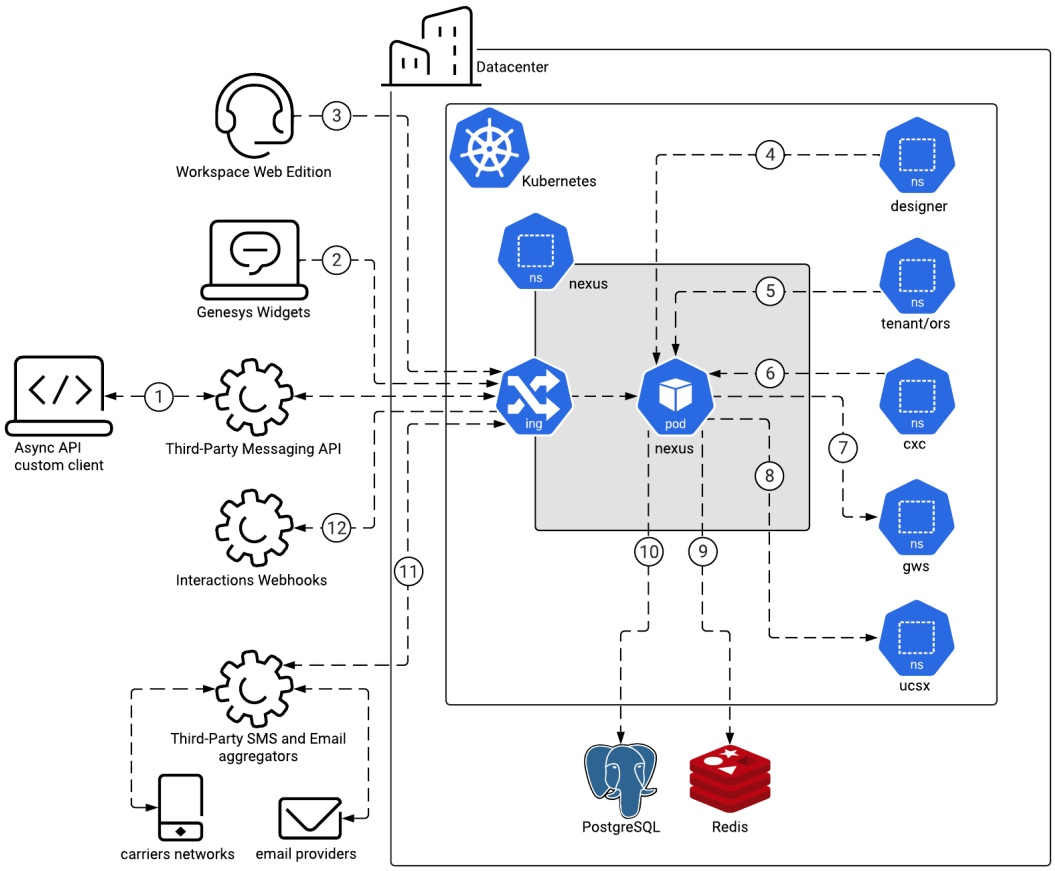
The following diagram shows the high-level architecture of Digital Channels (labelled as "nexus").

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 Digital Channels 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 Digital Channels as a service in the network. *Egress* means the Digital Channels service is the source, and *Ingress* means the Digital Channels 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	Async API custom client	Digital Channels through Third-Party	HTTPS	443	Ingress/ Egress	Delivers messages to Digital Channels

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
		Messaging API				and sends messages from Digital Channels.
2	Genesys Widgets	Digital Channels	HTTPS	443	Ingress	Chat interaction data.
3	Workspace Web Edition	Digital Channels	HTTPS/WSS	443	Ingress	Data for chat, social media, and SMS interactions.
4	Designer	Digital Channels	HTTP	80	Intra-cluster	Routing information for various Digital Channels.
5	Tenant Service	Digital Channels	HTTP	80	Intra-cluster	Chat communication for routing and self-service purposes
6	CX Contact	Digital Channels	HTTP/WS	80	Intra-cluster	Outbound campaign data and record statuses.
7	Digital Channels	Genesys Web Services and Applications	HTTP/ CometD	80	Intra-cluster	Interaction management and configuration access.
8	Digital Channels	Universal Contact Service	HTTP	80	Intra-cluster	Access to contact and interaction history.
9	Digital Channels	Redis	Redis		Intra-cluster	Shared real-time data for ongoing chat sessions.
10	Digital Channels	PostgreSQL	Postgres		Intra-cluster	Provisioning data and authentication information for third-party bots.
11	Digital	Third-Party	HTTPS	443	Ingress	

Connection	Source	Destination	Protocol	Port	Classification	Data that travels on this connection
	Channels	SMS and Email aggregators				
12	Digital Channels	Interactions Webhooks	HTTPS	443	Egress	Notifications about state changes in chat and secure email interactions.

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?
Digital Channels	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 Digital Channels.

Related documentation:

-
-
-
-

RSS:

- For private edition

Limitations and assumptions

Digital Channels for private edition has the following limitations:

- Supports only a single-region model of deployment.
- Social media requires additional components that are not included in Digital Channels.

Download the Helm charts

Digital Channels in Genesys Multicloud CX private edition includes the following containers:

- nexus
- hubpp
- tenant_deployment

The service also includes a Helm chart, which you must deploy to install all the containers for Digital Channels:

- nexus

See Helm charts and containers for Digital Channels for the Helm chart version you must download for your release.

To download the Helm chart, navigate to the **nexus** folder in the JFrog repository. For information about how to download the Helm charts, see Downloading your Genesys Multicloud CX containers.

Third-party prerequisites

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

Third-party services

Name	Version	Purpose	Notes
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.	Digital Channels supports Redis deployed in either cluster (TLS) or non-cluster (non-TLS) mode.
PostgreSQL	11.x	Relational database.	No support for enforced SSL.
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.	

Name	Version	Purpose	Notes
Command Line Interface		The command line interface tools to log in and work with the Kubernetes clusters.	

Storage requirements

Digital Channels uses PostgreSQL and Redis to store all data.

Network requirements

For general network requirements, review the information on the suite-level Network settings page.

Genesys dependencies

Digital Channels has dependencies on the following Genesys services:

- Genesys Authentication
- Web Services and Applications
- Tenant Microservice
- Universal Contact Service
- Designer

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

Configure Digital Channels

Contents

Learn how to configure Digital Channels.

Related documentation:

-
-
-
-

RSS:

- For private edition

Complete the steps on this page to configure your Digital Channels deployment.

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.

Parameters

Parameter	Description	Valid values	Default
global.imageRegistry	The Docker registry from which Kubernetes pulls images.	A valid registry URL	nil
global.imagePullSecrets	An array of global docker-registry secret names.	An array of secret names	[] (does not add image pull secrets to deployed pods)
global.storageClass	The global storage class used for dynamic provisioning.	A valid storage class	nil
image.registry	The Nexus image registry.	A valid registry URL	TBD
image.repository	The Nexus image name.	A valid image name	nexus/nexus
image.pullPolicy	Specifies when Kubernetes pulls images from the registry on start up.	IfNotPresent or Always	IfNotPresent
imagePullSecrets	An array of docker-registry secret names.	An array of secret names	[] (does not add image pull secrets to deployed pods)

Parameter	Description	Valid values	Default
nameOverride	A string to partially override the nexus.fullname template. This string is prepended to the release name.	String	nil
fullnameOverride	A string to fully override the nexus.fullname template.	String	nil
nexus.redirectProtocol	Defines the Web Services and Applications to Nexus redirect protocol (HTTP or HTTPS).	A valid protocol	http://
nexus.fqdn	The internal or external URI of the nexus services.	http://nexus.nexus.svc or http(s)://	nil
nexus.redis.enabled	Specifies whether to use Redis. You must not changes this from the default value of true.	true	true
nexus.redis.nodes	A comma-separated list of Redis nodes to connect.	A valid URL	redis://nexus-redis-master.default.svc.cluster.local:6379
nexus.redis.useCluster	Specifies whether to deploy Redis as a cluster.	true or false	false
nexus.redis.enableTls	Specifies whether to use TLS on the Redis connection.	true or false	false
nexus.redis.password	The password for Redis authentication.	A valid password	""
nexus.db.host	The Postgres service URL.	A valid URL	nexus-postgres-postgresql.default.svc.cluster.local
nexus.db.port	The Postgres service port.	A valid port	5432
nexus.db.user	The user assigned for the Nexus application to access Postgres.	A valid user	nexus
nexus.db.password	The password assigned for the Nexus application to access Postgres.	A valid password	nexus
nexus.db.enableSSL	Enable an SSL connection to PostgreSQL.	true or false	false
podSecurityContext.runAsNonRoot	Specifies whether the	true or false	true

Parameter	Description	Valid values	Default
	container must run as a non-root user.		
podSecurityContext.runAsUser	The user ID to run the entry point of the container process.	A valid user ID	500
podSecurityContext.runAsGroup	The group ID to run the entry point of the container process.	A valid group ID	500
podSecurityContext.fsGroup	A supplemental group ID that applies to all containers in a pod.	A valid group ID	500
resources	The requests and limits for CPU and memory usage in Kubernetes. See the Kubernetes documentation for details.		requests: { cpu: "300m", memory: "512Mi" }
affinity	Specifies the affinity and anti-affinity for Digital Channels pods. See the Kubernetes documentation for details.	Object	{}
nodeSelector	The labels Kubernetes uses to assign pods to nodes. See the Kubernetes documentation for details.	Object	{}
tolerations	The tolerations Kubernetes uses for advanced pod scheduling. See the Kubernetes documentation for details.	Object	[]
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	""
monitoring.enabled	Specifies whether to deploy Custom Resource Definitions (CRD) for ServiceMonitors to determine which services should be	true or false	false

Parameter	Description	Valid values	Default
	monitored.		
service.type	The Kubernetes service type.	See the Kubernetes documentation for details.	LoadBalancer
service.port	The Kubernetes service HTTP port.	A valid port	80
service.httpsPort	The Kubernetes service HTTPS port.	A valid port	443
service.nodePorts.http	The Kubernetes service HTTP node port.	A valid port	""
service.nodePorts.https	The Kubernetes service HTTPS node port.	A valid port	""
service.externalTrafficPolicy	Enables client source IP preservation. See the Kubernetes documentation for details.	Cluster or Local	Cluster
service.loadBalancerIP	The IP address of the load balancer service.	A valid IP address	""
ingress.enabled	Enables the ingress controller resource.	true or false	false
ingress.annotations	The ingress annotations.	A valid set of annotations as "name: value"	[]
ingress.certManager	Add annotations for cert-manager.	true or false	false
ingress.hosts[0].name	The hostname of your Nexus installation.	A valid hostname	nexus.local

ingress.hosts[0].paths	The internal or external URI of the nexus services.	`paths:`
		`  - path: '/chat/v3/'`
		`  port: http`
		`  - path: '/nexus/v3/'`
		`  port: http`
		`  - path: '/ux/'`
		`  port: http`
		`  - path: '/admin/'`
		`  port: http`
		`  - path: '/auth/'`
		`  port: http`
		`  - path: '/health/'`
		`  port: http`
		
		or
		
		`paths:`
		`  - path: '/'`
		`  port: http`
			/

Parameter	Description	Valid values	Default
ingress.hosts[0].tls	Specifies whether to use TLS backend in ingress.	true or false	false
ingress.hosts[0].tlsHosts	An array of TLS hosts for ingress record. If nil, this value defaults to the value of ingress.hosts[0].name.	Valid hosts	nil
ingress.hosts[0].tlsSecret	The TLS secret (certificates).	A valid secret	nexus.local-tls-secret
ingress.secrets[0].name	The TLS secret name.	A valid name	nil
ingress.secrets[0].certificate	The TLS secret certificate.	A valid certificate	nil
ingress.secrets[0].key	The TLS secret key.	A valid key	nil
podAnnotations	Custom annotations for each pod.	A valid set of labels as "name: value"	{}

Configure security

To learn more about how security is configured for private edition, be sure to read Permissions and OpenShift security settings.

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.

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

Deploy Digital Channels

Contents

Learn how to deploy Digital Channels 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 Digital Channels.

Prepare your environment

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

GKE

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

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

AKS

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

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

Create a JSON file called **create-nexus-namespace.json** with the following content:

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

Use the JSON file to create a new namespace for Digital Channels:

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

Now, confirm the created namespace:

```
kubectl describe namespace nexus
```

Add Helm repo and execute Helm upgrade to create persistent volumes and persistent volume claims

```
helm repo add nexushelmrepo https://pureengage.jfrog.io/artifactory/helm-dev   --
username={jfrog_user} --password={jfrog_token}
```

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=
```

Deploy

To deploy Digital Channels, you need the Helm package and override files you downloaded in a previous step. Copy **values.yaml** and the Helm package (**nexus-.tgz**) to the installation location.

You must override the following key sections in **values.yaml**:

- image.*
- nexus.fqdn

- nexus.redis.*
- nexus.db.*
- ingress.*

Here's an example of how your **values.yaml** file might look:

```
deploymentType: Deployment
replicaCount: 1
image:
  registry: pureengage-docker-staging.jfrog.io
  repository: nexus/nexus
  pullPolicy: IfNotPresent
imagePullSecrets: [ mycred ]
nameOverride: ""
fullnameOverride: ""
existingSecret:
existingConfig:
nexus:
  fqdn: "http://digital."
  redirectProtocol: "http://"
  redis:
    enabled: true
    nodes: "redis://:"
    useCluster: true
    enableTls: false
    password: $nexus_redis_password
  db:
    host: ""
    port:
    user: ""
    password: nexus_db_password
    enableSsl: false
  social:
    apikey: ""
    retryTimeout: 10000
service:
  enabled: true
  type: ClusterIP
ingress:
  tls:
    - hosts:
      - digital.
      secretName: letsencrypt
  enabled: true
  hosts:
    - host: digital.
      paths:
        - path: '/chat/v3/'
          port: http
        - path: '/nexus/v3/'
          port: http
        - path: '/ux/'
          port: http
        - path: '/admin/'
          port: http
        - path: '/auth/'
          port: http
monitoring:
  enabled: true
  alarms: true
```

Run the following command to install Digital Channels:

```
helm install nexus ./nexus-.tgz --set version= -f values.yaml
```

Validate the deployment

To validate the deployment, send the following GET request:

```
$nexusURL/health/detail
```

Where **$nexusURL** is the fully qualified domain name (FQDN) for Digital Channels.

The response should look like this:


```
{
        "buildInfo": {
                "@genesys/nexus-admin-ux": "^1.0.21",
                "@genesys/nexus-ux": "^2.2.46",
                "version": "9.0.001.01.95292",
                "changeset": "8c3a2b34888d41b318d949a4bda2903368cf3bda",
                "timestamp": "Thu Oct 21 13:55:13 UTC 2021"
        },
        "startTime": "2021-10-22T10:40:27.456Z",
        "os": {
                "upTime": 1142897,
                "freemem": 105664512,
                "loadavg": [1.32, 0.68, 0.43],
                "totalmem": 2052542464
        },
        "upTime": 279363374,
        "memoryUsage": {
                "rss": 306659328,
                "heapTotal": 196685824,
                "heapUsed": 162114568,
                "external": 2490373,
                "arrayBuffers": 818214
        },
        "cache": {
                "ready": true,
                "state": "READY"
        },
        "db": {
                "ready": true
        },
        "state": "green"
}
```


The deployment was successful if `state="green"`. You can also confirm that `db.ready=true` and `cache.ready=true`.

Next steps

Complete the steps in the "Integrate and provision" chapter to finish deploying Digital Channels. See

Provisioning overview for details.

Upgrade, rollback, or uninstall Digital Channels

Contents

Learn how to upgrade, rollback or uninstall Digital Channels.

Related documentation:

-
-
-
-

RSS:

- For private edition

Upgrade Digital Channels

Use **helm upgrade** to upgrade to a new revision.

Rollback Digital Channels

Use **helm rollback** to rollback to the previous revision.

Uninstall Digital Channels

Use **helm delete** to uninstall the deployment.

Provisioning overview

Contents

Learn about the steps involved in provisioning Digital Channels.

Related documentation:

-
-
-
-

RSS:

- For private edition

After you configure and deploy Digital Channels, you must complete a number of provisioning steps to enable Digital Channels to work with other Genesys services.

1. Pre-configure tenant objects
2. Enable a tenant for Digital Channels
3. Provision API keys
4. Setting up Integration for Inbound and Outbound SMS (optional)
5. Setting up Integration for Outbound Email Campaigns (optional)

Pre-configure tenant objects

Contents

Learn how to configure your tenant resources for Digital Channels.

Related documentation:

-
-
-
-

RSS:

- For private edition

Complete the steps on this page to configure your tenant resources for Digital Channels.

> **Important**
> “Nexus” is the simplified name we use for the digital channels application and nodes, so you’ll see that name referenced throughout this document.

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
Digital Channels FQDN URL	$nexusURL	HTTPS URL string	https://nexus.mydomain.com	The fully qualified domain name (FQDN) for Digital Channels.
Contact center ID	$ccId	string	578ec98e-f07c-46ad-9675-f36c2263bd9f	The contact center ID provisioned in Web Services and Applications. If you don't have this ID, see Get contact center ID from GWS.

Parameter	Variable	Type	Example	Notes
Tenant name	$tenantName	string	t1001	CUSTOMER_NAME that used in tenant service during deployment.

Create a user

Use Agent Setup to create a user with these properties:

- Make sure **Agent** is not checked.
- Set **Name**, **First Name**, **Last Name** and **Employee ID** to `nexus`.
- Set the password.
- Set the System Administrator and Superuser.
- Add the user to the Super Administrator, Administrator and User access groups in the **Member Of** tab.

Configure the Genesys Web Services application

Contact your Genesys representative to add the following configuration options for Web Services and Applications:

Enable Nexus UX

Section	Option name	Value
[NexusCommunication]	label	Communication
	url	https://$nexusURL/ux/comm?customername=$tenantName&ccid=$ccId
	sandbox	ALLOW-SCRIPTS,ALLOW-FORMS,ALLOW-POPUPS,ALLOW-SAME-ORIGIN,ALLOW-DOWNLOADS
[NexusConversation]	label	Conversation
	url	https://$nexusURL/ux/conv?iid=$Interaction.Id$&customername=$tenantName&ccid=$
	sandbox	ALLOW-SCRIPTS,ALLOW-FORMS,ALLOW-POPUPS,ALLOW-SAME-ORIGIN,ALLOW-DOWNLOADS
[interaction-workspace]	workspace.web-content	NexusCommunication
	interaction.web-content	NexusConversation
	service-client-api.accepted-web-content-origins	*
	service-client-api.allow-full-api	true
	privilege.chat.can-place-on-hold-async	false

> **Important**
> When enabling the Conversation tab you must also add the Communication tab to ensure proper functionality of the solution. If you want to disable the Communication tab for your agents, you must hide it instead of removing it. To hide a tab, add the **mode** option with the value set to `HIDDEN` on either the NexusCommunication section or the NexusConversation section.

Configure the Universal Contact Server application

Contact your Genesys representative to add the following configuration options for Universal Contact Service:

> **Important**
> The following configuration is only applicable to UCS 8.5.x.

Enable Nexus UX

Section	Option name	Value
[index]	enabled	true
[index.contact]	enabled	true
	storage-path	

Update the Environment tenant

Contact your Genesys representative to add the following option for the Environment tenant:

Section	Option name	Value
[nexus]	url	$nexusURL

Create transactions

Use Agent Setup to create the following transactions in the Environment tenant:

- List 'NexusEndpoints'

- List 'NexusServices'

Create scripts

Contact your Genesys representative to create the following scripts in the Environment tenant:

asynchold queue

Property	Value
Name	asynchold
Type	Interaction Queue
Tenant	Environment
State enabled	checked

asynchold View script

General Tab

Property	Value
Name	asynchold/scheduled_view
Type	Interaction Queue View
Tenant	Environment
State enabled	checked

Annex Tab

Section	Property	Value
[Namespace]	Name	scheduled_view
[View]	Condition	
	Order	
	Queue	asynchold
	scheduling-mode	scheduled-and-unscheduled
	freeze-interval	30

asynchold application script

General Tab

Property	Value
Name	asynchold.ER
Type	Enhanced Routing

Property	Value
Tenant	Environment
State enabled	checked

Annex Tab

Section	Property	Value
[Application]	url	$nexusURL/scxml/ redirect_queue.scxml

asynchold Submitter script

General Tab

Property	Value
Name	asynchold.IS
Type	Interaction Submitter
Tenant	Environment
State enabled	checked

Annex Tab

Section	Property	Value
[Submitter]	View	asynchold/scheduled_view
	Strategy	asynchold.ER

undelivered queue

Property	Value
Name	undelivered
Type	Interaction Queue
Tenant	Environment
State enabled	checked

undelivered View script

General Tab

Property	Value
Name	undelivered/scheduled_view
Type	Interaction Queue View
Tenant	Environment
State enabled	checked

Annex Tab

Section	Property	Value
[Namespace]	Name	scheduled_view
[View]	Condition	
	Order	
	Queue	undelivered
	scheduling-mode	scheduled-and-unscheduled

undelivered application script

General Tab

Property	Value
Name	undelivered.ER
Type	Enhanced Routing
Tenant	Environment
State enabled	checked

Annex Tab

Section	Property	Value
[Application]	url	$nexusURL/scxml/ undelivered.scxml

undelivered Submitter script

General Tab

Property	Value
Name	undelivered.IS
Type	Interaction Submitter
Tenant	Environment
State enabled	checked

Annex Tab

Section	Property	Value
[Submitter]	View	undelivered/scheduled_view
	Strategy	undelivered.ER

Create an SMS interaction subtype

Contact your Genesys representative to create the following Business Attribute Value in the "Interaction Subtype" Business Attribute:

Name	Display name	Description
SMS	SMS	The SMS text message. This interaction subtype is only required if you are using SMS.

Enable a tenant for Digital Channels

Contents

Learn how to enable your tenant for Digital Channels.

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
> “Nexus” is the simplified name we use for the Digital Channels application and nodes, so you’ll see that name referenced throughout this document.

Complete the steps on this page to provision your tenant and set up Digital Channels to work with Web Services and Applications (GWS).

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
GWS URL	$gwsURL	HTTPS URL string	https://gws.mydomain.com	Host of the Web Services and Applications load balancer.
Auth URL	$authURL	HTTPS URL string	https://auth.mydomain.com	Host of the Auth Services load balancer.
GWS admin username	$gwsAdminName	string	ops	The GWS ops credentials. See Provision Genesys Web Services and Applications.
GWS admin password	$gwsAdminPass	string	ops	The GWS ops credentials. See Provision Genesys Web Services and Applications.
Tenant User	$tenantUser	string	nexus	The Person username from your tenant configuration. See Create a Person object.
Tenant User Password	$tenantUserPassword	string	nexus_password	The Person password from your tenant configuration. See Create a Person object.
Contact Center ID	$ccId	string	578ec98e-f07c-46ad-9675-f36c22f3ba9f	The contact center ID provisioned in Web Services and Applications. If you don't have this ID, see Get contact center ID from GWS below.
Tenant Name	$tenantName	string	premise_tenant	The tenant name you used as in the Enable Nexus UX step.
Digital Channels FQDN URL	$nexusURL	HTTPS URL string	https://nexus.mydomain.com	The fully qualified domain

Parameter	Variable	Type	Example	Notes
				name (FQDN) for Digital Channels.
UCS URL	$ucsURL	HTTPS URL string	https://ucs.mydomain.com	A URL pointing on the Universal Contact Service (UCS) entry point.

> **Important**
> Use a REST client or curl utility to make the requests explained on this page. Make sure to substitute the variables - prefixed with '$' - with their values.

Get contact center ID from GWS

If you don't have the contact center ID (as specified in the **Prerequisites** table above), you can get it with the following request:

```
curl --user $gwsAdminName:$gwsAdminPass --request GET '$authURL/environment/v3/contact-centers'
```

The expected response:

```
{
    "data": {
        "contactCenters": [
            {
                "id": "",
                "environmentId": "",
                "domains": [

                ],
                "auth": "configServer"
            }
        ]
    }
}
```

Add GWS to the list of available GWS services for Nexus

Complete the steps in this section to add GWS to the list of available GWS services for Nexus.

Create the authentication client

To create the authentication client, send a POST request to GWS that includes a body parameter called **data** in JSON format. Give **data** the following properties:

Property	Value
clientType	CONFIDENTIAL
internalClient	true
authorizedGrantTypes	refresh_token, password, client_credentials, authorization_code
redirectURIs	$nexusURL

Property	Value
authorities	ROLE_INTERNAL_CLIENT
scope	*
description	nexus_client
name	nexus_client
client_id	nexus_client
client_secret	- randomly generated and saved for further use
accessTokenExpirationTimeout	43200
refreshTokenExpirationTimeout	2592000

Sample request

```
curl --user ops:ops --request POST '$gwsURL' \
--header 'Content-Type: application/json' \
--data '{"data": {
    "internalClient": true,
    "name": "nexus_client",
    "clientType": "CONFIDENTIAL",
    "client_id": "nexus_client",
    "client_secret": "",
    "authorities": ["ROLE_INTERNAL_CLIENT"],
    "scope": ["*"],
    "authorizedGrantTypes": ["client_credentials", "authorization_code", "refresh_token", "implicit", "password"],
    "redirectURIs": ["$nexusURL"],
    "accessTokenExpirationTimeout": 43200,
    "refreshTokenExpirationTimeout": 2592000
    }
}'
```

In the above example, the `ops:ops` GWS superuser is different for your environment.

Expected response

The expected response is **200 OK**.

Verify client

To verify that authentication was successful, send a POST request to GWS:

```
curl --user nexus_client: \
--request POST '$authURL/auth/v3/oauth/token?grant_type=client_credentials>ope=*&client_id=nexus_client&client_secret='
```

This is the response:

```
{"access_token":"","token_type":"bearer","expires_in":43199,"scope":"*"}
```

Add GWS to the nex_gapis table for Nexus

Execute the following query in the PostgreSQL command line interface:

```
NSERT INTO nex_gapis (url, clientid, apikey, clientsecret, created) VALUES ('$gwsURL', 'nexus_client', 'NA', '', now());
```

Provision Digital Channels in GWS

To provision Digital Channels in GWS, send a POST request to GWS:

```
curl --user $gwsAdminName:$gwsAdminPass --request POST '$authURL/environment/v3/contact-
centers/$ccId/settings' \
--header 'Content-Type: application/json' \
--data '{
    "data":
    {
        "name":"chatservice-uri",
        "location":"/",
        "value":"$nexusURL",
        "shared":false
    }
 }
```

Get the authentication token

To get the authentication token, send a POST request to GWS:

```
curl --user nexus_client: --request POST '$authURL/auth/v3/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'username=$ccId\\$tenantUser' \
--data-urlencode 'client_id=nexus_client' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'password=$tenantUserPassword'
```

The expected response:

```
{"access_token":"","token_type":"bearer","refresh_token":"","expires_in":43199,"scope":"*"}
```

As an output of this step, you will have the GWS access token:

Parameter	Variable	Type	Example	Notes
Nexus API Key	$outGWSAccessToken	UUID string	9b7682b7-cbce-422f-9bbb-ecda85e61695	The access_token from the response example.

Provision the Universal Contact Service connection

You must complete this step if you are using Universal Contact Service in your environment.

Make the following POST request to create the PlatformUCS service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/PlatformUCS \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "$ucsURL",
    "secret": {},
    "data" : {}
}'
```

Provision the tenant in Digital Channels

To provision the tenant in Digital Channels, make the following POST request:

```
curl --request POST '$nexusURL/nexus/v3/provisioning/tenants' \
--header 'Authorization: Bearer $outGWSAccessToken' \
--header 'Content-Type: application/json' \
--data '{
  "id": "$ccId",
  "name": "$tenantName",
  "type": "PureEngage",
  "backendurl": "$gwsURL",
  "username": "$tenantUser",
  "token": "$tenantUserPassword",
  "genesystenantid": 1
}'
```

The expected response:

```
{
    "status": {
        "code": 0
    },
    "data": {
        "xapikey": "$nexusApiKey"
    },
    "operationId": "ec90f3d2-f4b5-47fd-9004-2c309398ab38"
}
```

As an output of this step, you will have the specific tenant API key:

Parameter	Variable	Type	Example	Notes
Nexus API Key	$nexusApiKey	UUID string	9b7682b7-cbce-422f-9bbb-ecda85e61695	The **xapikey** from the response example. For creating chat.

Enable routing using Designer Applications

Use Agent Setup to add the following values to the "DesignerEnv" transaction in the Environment tenant. If the transaction doesn't exist, create it with a type of List and "DesignerEnv" as Name and Alias.

Property	Value
baseurl	/nexus/v1
password	
url	/nexus/v3

Provision API keys

Contents

Learn how to provision API keys for Digital Channels.

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
> “Nexus” is the simplified name we use for the Digital Channels application and nodes, so you’ll see that name referenced throughout this document.

Complete the steps on this page to provision API keys for Digital Channels.

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
Auth URL	$authURL	HTTPS URL string	https://auth.mydomain.com	Host of the Auth Services load balancer.
Contact Center ID	$ccId	string	578ec98e-f07c-46ad-9675-f36c22f3ba9f	The contact center ID provisioned in Web Services and Applications. If you don't have this ID, see Get contact center ID from GWS below.
GWS admin username	$gwsAdminName	string	ops	The GWS ops credentials. See Provision Genesys Web Services and Applications.
GWS admin password	$gwsAdminPass	string	ops	The GWS ops credentials. See Provision Genesys Web Services and Applications.
Digital Channels FQDN URL	$nexusURL	HTTPS URL string	https://nexus.mydomain.com	The fully qualified domain name (FQDN) for Digital Channels.
Tenant Name	$tenantName	string	premise_tenant	The tenant name you used as in the Enable Nexus UX step.
Tenant User	$tenantUser	string	nexus	The Person username from your tenant configuration. See Create a Person object.
Tenant User Password	$tenantUserPassword	string	nexus_password	The Person password from your tenant configuration. See Create a Person object.

Tenant API keys

> **Important**
> Use a REST client or curl utility to make the requests explained on this page. Make sure to substitute the variables - prefixed with '$' - with their values.

Each API key you provision has a tenant or a set of tenants to which it is provisioned. The API key is also associated with a set of permissions. In the example below, we create a tenant API key with consumer chat API key permissions. Complete the steps below to provision an API key. First, get the authentication token by sending a POST request to GWS:

```
curl --user nexus_client: --request POST '$authURL/auth/v3/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'username=$ccId\\$tenantUser' \
--data-urlencode 'client_id=nexus_client' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'password=$tenantUserPassword'
```

The expected response:

```
{"access_token":"","token_type":"bearer","refresh_token":"","expires_in":43199,"scope":"*"}
```

As an output of this step, you will have the GWS access token for the tenant:

Parameter	Variable	Type	Example	Notes
Tenant GWS access token	$outGWSAccessToken	UUID string	9b7682b7-cbce-422f-9bbb-ecda85e61695	The access_token from the response example.

To provision the API key, send a POST request to Digital Channels:

```
curl --request POST '$nexusURL/nexus/v3/apikeys' \
--header 'Authorization: $outGWSAccessToken' \
--header 'Content-Type: application/json' \
--data '{
    "enabled": true,
    "tenant": "$ccId",
    "name": "API key for $tenantName",
    "permissions": [
        "nexus:consumer:chat"
    ]
}'
```

The expected response:

```
{
    "id": "",
```

```
    "tenant": "$ccId",
    "name": "API key for $tenantName",
    "permissions": [
        "nexus:consumer:chat"
    ],
    "enabled": true,
    "created": "",
    "revoked": false,
    "expires": false,
    "createdby": "",
    "capacity": 120,
    "frequency": 60
}
```

As an output of this step, you will have the tenant API key:

Parameter	Variable	Type	Example	Notes
Tenant API Key	$apikey	UUID string	9b7682b7-cbce-422f-9bbb-ecda85e61695	The id from the response example

Cluster API keys

Complete the following steps to provision a cluster (multi-tenant) API key.

First, get the cluster authentication token by sending a POST request to GWS:

```
curl --user nexus_client: --request POST '$authURL/auth/v3/oauth/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'username=$gwsAdminName' \
--data-urlencode 'client_id=nexus_client' \
--data-urlencode 'grant_type=password' \
--data-urlencode 'password=$gwsAdminPass'
```

The expected response:

```
{"access_token":"","token_type":"bearer","refresh_token":"","expires_in":43199,"scope":"*"}
```

As an output of this step, you will have the cluster GWS access token:

Parameter	Variable	Type	Example	Notes
Cluster GWS access token	$outClusterGWSAccessToken	UUID string	9b7682b7-cbce-422f-9bbb-ecda85e61695	The access_token from the response example.

To provision the API key, send a POST request to Digital Channels:

```
curl --request POST '$nexusURL/nexus/v3/apikeys' \
--header 'Authorization: $outClusterGWSAccessToken' \
--header 'Content-Type: application/json' \
--data '{
    "enabled": true,
    "tenant": "*",
    "name": "Cluster API key",
    "permissions": [
        "nexus:cluster:*"
```

```
    ]
}'
```

The expected response:

```
{
    "id": "",
    "tenant": "*",
    "name": "Cluster API key",
    "permissions": [
        "nexus:cluster:*"
    ],
    "enabled": true,
    "created": "",
    "revoked": false,
    "expires": false,
    "createdby": "",
    "capacity": 120,
    "frequency": 60
}
```

As an output of this step, you will have the cluster API key:

Parameter	Variable	Type	Example	Notes
Cluster API Key	$clusterApikey	UUID string	9b7682b7-cbce-422f-9bb1-ecda85e61695	The id from the response example

Designer API key

First, follow the instructions above to create a dedicated Designer API key. Set `permissions` as:

```
"permissions": [ "nexus:designer:chat" ]
```

Genesys recommends creating a separate key from the consumer chat API key so that you can update your consumer API key without impacting Designer applications.

```
curl --request POST '$nexusURL/nexus/v3/apikeys' \
--header 'Authorization: $outGWSAccessToken' \
--header 'Content-Type: application/json' \
--data '{
    "enabled": true,
    "tenant": "$ccId",
    "name": "Designer API key",
    "permissions": [
        "nexus:designer:chat"
    ]
}'
```

The expected response:

```
{
    "id": "",
    "tenant": "$ccId",
    "name": "Designer API key",
    "permissions": [
```

```
        "nexus:designer:chat"
    ],
    "enabled": true,
    "created": "",
    "revoked": false,
    "expires": false,
    "createdby": "",
    "capacity": 120,
    "frequency": 60
}
```

As an output of this step, you will have the specific tenant API key:

Parameter	Variable	Type	Example	Notes
Designer API Key	$designerApikey	UUID string	9b7682b7-cbce-422f-9bb6-ecda85e61695	The id from the response example

Setting up Integration for Inbound and Outbound SMS

Contents

- Administrator
- Developer

Learn how to enable SMS connectivity for inbound conversations and outbound campaigns.

Related documentation:

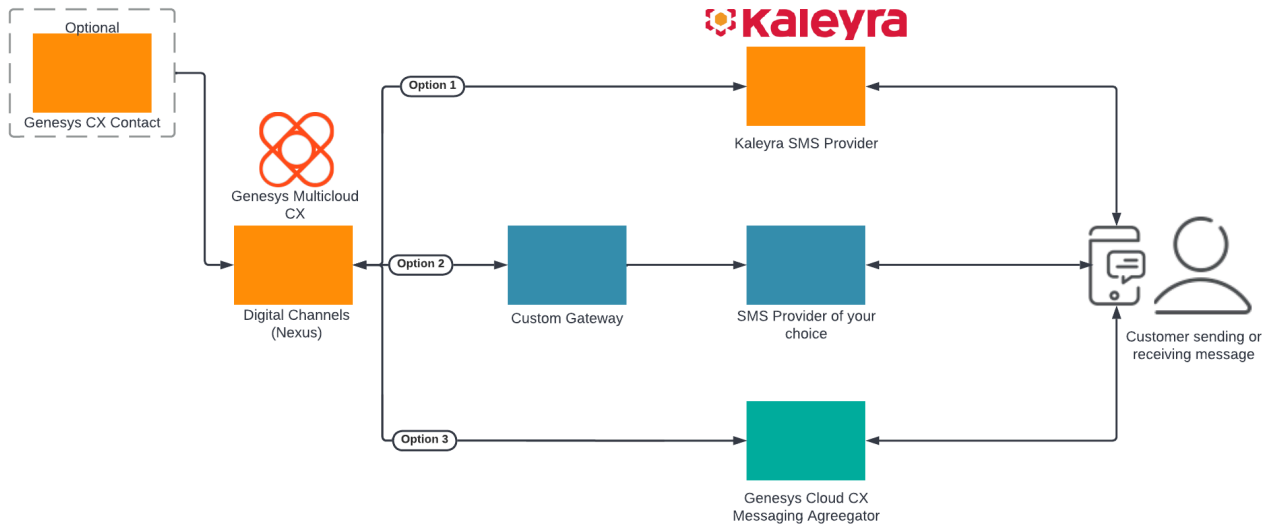
-
-
-
-

RSS:

- For private edition

> **Important**
> "Nexus" is the simplified name we use for the Digital Channels application and APIs, so you'll see that name referenced in this document.

Genesys Multicloud CX supports SMS interactions using one of these three options: built-in connector to Kaleyra SMS text messaging service, a custom gateway built with a third-party messaging API, or a built-in connector to Genesys Cloud CX SMS aggregation service.



After completing the setup steps on this page, you will be able to:

- Receive SMS messages from your customers on your corporate short code or long code and allow a Designer application or agent to respond to them.

- Send SMS messages in your outbound campaigns through CX Contact.

To get started, complete the configuration steps for **one** of the following scenarios:

- Set up Digital Channels to use Kaleyra SMS OR
- Set up Digital Channels to use a Custom Gateway OR
- Set up Digital Channels to use Genesys Cloud CX SMS Agreegation

Option 1: Set up Digital Channels to use Kaleyra SMS

Complete the steps in this section to set up Digital Channels to use the built-in connector to Kaleyra.

1. Review the prerequisites table.
2. Use the Digital Channels provisioning API to create Digital Channels services definition.
3. Manage and provision sender IDs.

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
Company's sender id	$tfn	string	1650466114	The sender id can be a short code (5 digits) or a long code (10 digits).
Contact Center Id	$ccid	UUID string	45acae06-6b7c-4f97-9c76-471cbd2bef7	This value comes from your Web Services and Applications deployment.
Digital Channels URL	$baseURL	string	http://digital.example.com	Publicly available URL for Digital Channels API
Kaleyra API URL	$kaleyraURL	string	http://directtext.mgage.com	URL for Kaleyra SMS API.
Kaleyra user name	$kaleyraUserName	string	Secret	The username for an account with permission to send SMS.
Kaleyra password	$kaleyraPassword	string	Secret	The password for an account with permission to send SMS.

Create Digital Channels services definitions

Enable Digital Channels to use the built-in connector to Kaleyra SMS services (Kaleyra was formerly known as mGage). You must configure the following services in Digital Channels within your tenant:

- SMS - Enables SMS media for the tenant and selects Kaleyra as the default provider.
- mGageSMS - Integrates your Genesys Multicloud CX tenant to Kaleyra for SMS communication.

> **Important**
>
> Use a REST client or curl utility to provision the following services in Digital Channels using the provisioning API. Make sure to substitute the variables - prefixed with '$' - with their values. Note: You must create these services once per tenant.

Create the **SMS** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/SMS \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "N/A",
    "data" : { "provider": "mGage" },
    "secret": {}
}'
```

Create the **mGageSMS** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/mGageSMS \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
        "url": "$kaleyraURL"
        "data": { },
        "secret": {
             "username": "$kaleyraUserName",
             "password": "$kaleyraPassword"         } }'
```

Adding sender IDs to Kaleyra

If you need to add a new short code or text-enabled phone number, contact your Genesys representative to complete this step.

Manage and provision sender IDs

Provision sender IDs in Digital Channels.

Option 2: Set up Digital Channels to use a Custom Gateway

Complete the steps in this section to set up Digital Channels to use a custom SMS gateway provider.

1. Review the prerequisites table.
2. Use the Digital Channels provisioning API to create Digital Channels services definitions.
3. Manage and provision sender IDs..

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
Company's sender id	$asyncPhoneNumber	string	16504661149	
Third-party Messaging Webhook URL	$asyncWebhookURL	string	https://genesys-webhook.company.com	The FQDN of the third-party service implementing the Third-Party Messaging Webhook.
Third-Party Messaging API secret key	$asyncAPISignatureKey	string	Secret	The key used by the third-party service to calculate the signature for calls to the Third-Party Messaging API.
Third-Party Messaging Webhook secret key	$asyncWebhookSignatureKey	string	Secret	The key used by Digital Channels to calculate the signature for calls to the third-party service through the webhook.

Create Digital Channels services definitions

In this step, you will enable Digital Channels to use a custom gateway for the SMS provider of your choice. You must configure the following services in Digital Channels within your tenant:

- Async - Integrates the Genesys Multicloud CX tenant to the custom gateway used to communicate with the SMS provider of your choice.
- SMS - Enables SMS media for the tenant and selects the Async provider.

> **Important**
>
> Use a REST client or curl utility to provision the following services in Digital Channels using the provisioning API. Make sure to substitute the variables - prefixed with '$' - with their values. You must create these services once per tenant.

Create the **Async** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/Async \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
        "data": {
          "channels": [
            {
              "channelId": "$asyncPhoneNumber",
              "webhook": { "url": "$asyncWebhookURL" }
            },
            {
              "channelId": "$asyncEmailDomain",
              "webhook": { "url": "$asyncWebhookURL" }
            }
          ]
        },
        "secret": {
          "channels": [
            {
              "channelId": "$asyncPhoneNumber",
              "webhook": { "secret": "$asyncWebhookSignatureKey" },
              "api": { "secret": "$asyncAPISignatureKey" }
            },
            {
              "channelId": "$asyncEmailDomain",
              "webhook": { "secret": "$asyncWebhookSignatureKey" },
              "api": { "secret": "$asyncAPISignatureKey" }
            }
          ]
        }
      }'
```

Create the **SMS** service:

```
curl -X POST \
```

```
  $nexusURL/nexus/v3/provisioning/services/$ccid/SMS \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "N/A",
    "data" : { "provider": "Async" },
    "secret": {}
}'
```

Manage and provision sender IDs

If you need to register a new or a existing short code or a text-enabled phone number, contact your Genesys representative to complete this step.

Provision sender IDs in Digital Channels.

> **Important**
>
> An SMS must have a "provider" property equal to "Async" to use the Third-Party Messaging API implementation.

Add custom HTTP headers

You can add custom HTTP headers with static values to the webhooks sent by Digital Channels to the third-party messaging aggregator. On the transaction **NexusServices > [your async provider]**, add a property that starts with the "header:" prefix and set it to your static value. For example, `header:custom-header-for-async-1 = 12345`. The webhook from Digital Channels will include this property name and value in the header:

```
HTTP
custom-header-for-async-1: 12345
X-Hub-Signature:
X-B3-TraceId:
Content-Type: application/json

{
  "messages": [
    {
        ... RichMedia message ...
    }
  ]
}
```

Option 3: Set up Digital Channels to use Genesys Cloud CX SMS

Aggregation

Complete the steps in this section to set up Digital Channels to use Genesys Cloud CX SMS Aggregation as the SMS gateway.

1. Review the prerequisites table.
2. Contact your Genesys representative to create a Genesys Cloud CX organization and get administrator user credentials. Your Genesys representative also must add the Genesys SMS Aggregation product to your organization.
3. Create the Digital Channels integration in Genesys Cloud CX. This will give you a clientId and clientSecret to authenticate API calls with the Digital Channels provisioning API.
4. Create a Digital Channels API key for Genesys Cloud CX.
5. Use the Digital Channels provisioning API to create Digital Channels services definition.
6. Manage and provision sender IDs.

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
GWS tenant Contact Center ID	$ccid	UUID string	45acae06-6b7c-4f97-9c76-471cbd2bef7	This value comes from your Web Services and Applications deployment.
Nexus Provider	$nexusProvider	string	Portico	This option depends on the provider's value of the service. Supported values: - Portico - Async - mGage

Request a Genesys Cloud CX organization

Contact your Genesys representative to create a Genesys Cloud CX organization and get administrator user credentials. Your Genesys representative also must add the Genesys SMS Aggregation product to your organization.

Make sure your Genesys representative provides you with the details in the **Genesys Cloud CX information** table.

Genesys Cloud CX information

Parameter	Variable	Type	Example	Notes
Genesys Cloud CX Organization ID	$orgId	UUID string	47d8329d-1c28-4c86-9374-5596bddfee15	Your Genesys Cloud CX organization ID.

Parameter	Variable	Type	Example	Notes
Genesys Cloud CX Organization admin user credentials	$orgUsername $orgPassword	string	admin-user / admin-password	The username and password for an account with administrative permissions for this organization.
Genesys Cloud CX Login URL	$gcLoginURL	HTTPS URL string	https://login.mypurecloud.com	Your Genesys Cloud CX login URL (depends on your organization region).
Genesys Cloud CX API URL	$gcAPIURL	HTTPS URL string	https://api.mypurecloud.com	Your Genesys Cloud CX login URL (depends on your organization region).

Create the Digital Channels integration in Genesys Cloud CX

Complete the steps in this section as an administrator user in Genesys Cloud CX to create the integration client credentials that will be used by Digital Channels to access Genesys Cloud CX APIs to send and receive messages. You're going to create a new role, assign it to your admin user, and create the access credentials.

First, create the new role:

1. Navigate to $gcLoginURL (for example, https://login.mypurecloud.com) and log in to Genesys Cloud CX with your $orgUsername/$orgPassword.
2. Go to **Admin**.
3. Under **People and Permissions**, click **Roles/Permissions**.
4. Click **Add Role** and give it a name. For example, `Nexus Messaging`.
5. Under **Permissions**, search for **messaging** and select **messaging > All Permissions** and **messagingProvisioning > All Permissions**. Save your changes.

Next, assign the role to your administrator user:

1. Click **Admin**.
2. Under **People and Permissions**, click **People**.
3. Search for your admin user.
4. Under **Roles,** switch the view to **All** and search for the name of your new role (`Nexus Messaging`). Click to enable the role and then save your changes.
5. Log out and log in again to enable the permissions.

Now create access credentials for the Digital Channels integration.

1. Click **Admin**.
2. Under **Integrations**, click **OAuth**.
3. Click **Add Client.**
4. Under **Client Details**, set **App Name** to `Nexus Messaging Integration` and select the Client Credentials **Grant Type**.
5. Click **Roles** and assign the `Nexus Messaging` role. Save your changes.
6. Go back to **Client Details** and copy the values for **clientId** and **clientSecret**.

As the output of this step, you will have the access credentials:

Parameter	Variable	Type	Example
Nexus Integration client ID	$clientId	UUID string	4da40a9de-b113-4024-8ba9-c9dd89c91f67
Nexus Integration client secret	$clientSecret	string	aKSXEgLO57cm6FqxD4hrjkcW-iuWiXhd0uF0WOcZUm2

Create a Digital Channels API key for Genesys Cloud CX

To create an API key that will be used by Genesys Cloud CX to send requests to Digital Channels, follow the steps in Provision API keys. Make sure to use the following parameters:

```
"tenant": "*"
"name":"Portico Cluster API Key"
"permissions" : ["nexus:cluster:*"]
```

As an output of this step, you will have the API key:

Parameter	Variable	Type	Example
Messaging Cluster API Key	$apikey	UUID string	9b7682b7-cbce-422f-9bbb-ecda85e61695

Create Digital Channels services definitions

Enable Digital Channels to use Genesys Cloud CX SMS Aggregation as the SMS provider. You must configure the following services in Digital Channels within your tenant:

- PurecloudIDP - Integrates the Genesys Multicloud CX tenant to the Genesys Cloud CX organization.
- SMS - Enables SMS media for the tenant and selects the provider.
- PorticoSMS - Enables the SMS service through Genesys Cloud CX.

> **Important**
>
> Use a REST client or curl utility to provision the following services in Digital Channels using the provisioning API. Make sure to substitute the variables - prefixed with '$' - with their values. Note: You must create these services once per tenant.

Create the **PurecloudIDP** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/PurecloudIDP \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "$gcLoginURL",
    "secret": {"clientId": "$clientId", "clientSecret": "$clientSecret"},
    "data" : {}
}'
```

Create the **SMS** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/SMS \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
```

```
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "N/A",
    "secret": {},
    "data" : {"tokenProvider": "Purecloud"}
}'
```

Create the **PorticoSMS** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/PorticoSMS \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "$gcAPIURL",
    "secret": {},
    "data": {}
}'
```

Manage provision sender IDs

You can purchase new SMS numbers or re-use your existing text-enabled numbers.

If you need to register an existing phone number or new short code, contact your Genesys representative to complete this step. Otherwise, follow the steps below to use the Genesys SMS Aggregation API from Genesys Cloud CX to purchase and register toll-free numbers from the pool of available numbers. **Note:** Each purchased number will incur additional costs to your account.

Retrieve the Genesys Cloud CX token

Any Genesys Cloud CX operation has to include a security token that remains valid for a configured amount of time. When the token expires, you must retrieve it again in order to send new requests.

To retrieve your token, use Basic Authentication where the username is **$clientId** and the password is **$clientSecret**.

```
curl -X POST \
  $gcLoginURL/oauth/token \
  -H 'Authorization: Basic YOUR_BASIC_AUTHENTICATION_SECRETS' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'grant_type=client_credentials'
```

The response contains:

- **access_token** - You must include this in all subsequent requests.
- **expires_in** - Indicates how long the token is valid.
- **token_type** - Indicates how this token needs to be included in subsequent requests (bearer token).

List purchased and registered numbers

Run the following command to list your purchased and registered numbers:

```
curl -X GET \
  $gcAPIURL/api/v2/messaging/sms/provisioning/tollfreenumbers/ \
  -H 'Authorization: Bearer $access_token'
```

Search for available toll free-number

To order a new number, first search for available numbers and then select one of the options.

```
curl -X GET \
  $gcAPIURL/api/v2/messaging/sms/provisioning/tollfreenumbers/available \
  -H 'Authorization: Bearer $access_token'
```

The response contains a few currently available numbers - choose the one you like. We'll use the variable **$tfn** to represent this number.

Order a toll-free number number

When you send this request, use the **comment** and **emailAddress** fields to help Genesys Customer Care quickly identify the best contact person if there's an issue with the SMS service. For example, you can include your organization name in the **comment** field.

```
curl -X POST \
  $gcAPIURL/api/v2/messaging/sms/provisioning/tollfreenumbers/ \
  -H 'Authorization: Bearer $access_token' \
  -H 'Content-Type: application/json' \
  -d '[
    {
        "tollfreeNumber": "$tfn",
        "comment": "Nexus Premise ACME Corp",
        "moUrl": "$nexusURL/nexus/v3/sms/message",
        "drUrl": "$nexusURL/nexus/v3/sms/receipt",
        "webhookUsername": "$ccid",
        "webhookPassword": "$GMAKey",
        "emailAddress": "sms.admin@acme.test.com"
    }
]'
```

Finally, provision your new numbers in Digital Channels.

Provision sender IDs in Digital Channels

Complete the following steps for each short code or text enabled toll-free number you want to use to send and receive SMS messages.

1. Log in to Designer and create an application to route SMS interactions. Create a chat endpoint and assign it to your application, making note of the name you use. We'll use the variable **$designerEndpointName** to represent this value.
2. Log in to Platform Administration and go to **Environment > Transactions > NexusServices > Options**.
3. Create a section that starts with 'chat.' and includes some text that represents the purpose of number.

For example, `chat.SMS_Main_Corporate` or `chat.SMS_CustomerSupport`. Note: The 'chat.' prefix is required and the rest of the value can be made up of letters and underscore or dashes, but not spaces.

4. In this new section, create the following options:
   - channelId = **$tfn**
   - channelType = sms
   - endpoint = chat.**$designerEndpointName**
   - interactionSubtype = SMS
   - interactionType = Inbound
   - media = chat
   - cxcOnly = false
     Note: Set `cxcOnly` to **true** for CX Contact integrations. If **true**, incoming SMS messages are not processed as chat messages and are not delivered to an agent. The default value is **false**.
   - provider = $nexusProvider (Default: Portico, if the value is missing)

Setting up Integration for Outbound Email Campaigns

Contents

- Administrator
- Developer

Learn how to enable Email connectivity for outbound campaigns.

Related documentation:

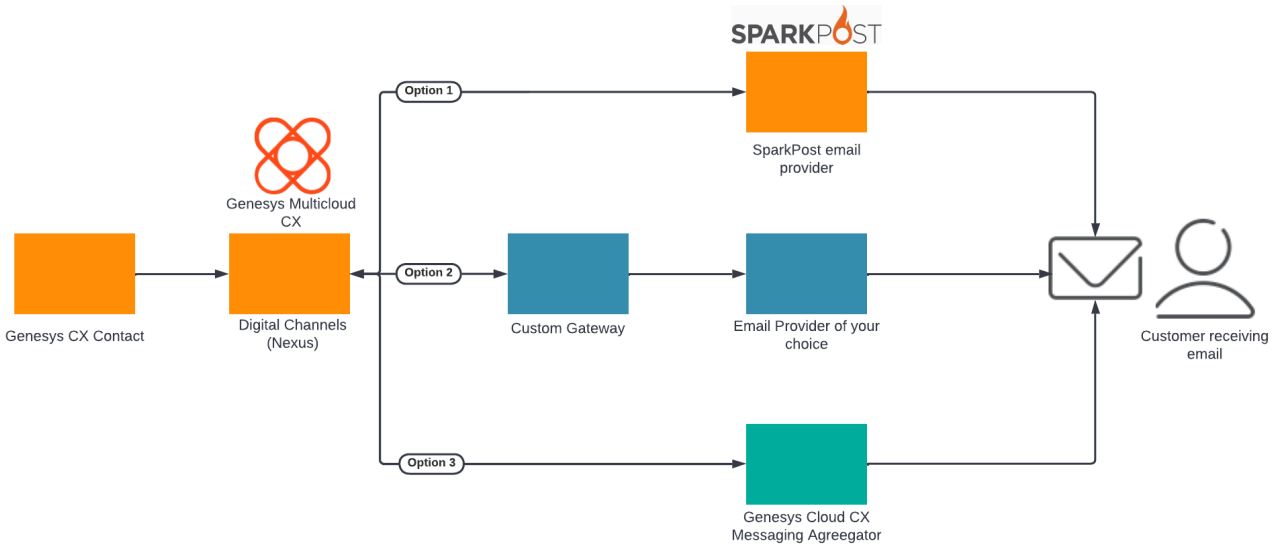
-
-
-
-

RSS:

- For private edition

> **Important**
> "Nexus" is the simplified name we use for the Digital Channels application and APIs, so you'll see that name referenced in this document.

Genesys Multicloud CX supports outbound email interactions using one of these three options: built-in connector to SparkPost email provider, a custom gateway built with a third-party email API or orbuilt-in connector to Genesys Cloud CX aggregation service.



After completing the setup steps on this page, you will be able to:

- Send Email in your outbound campaigns through CX Contact.

To get started, complete the configuration steps for **one** of the following scenarios:

- Set up Digital Channels to use SparkPost Email OR
- Set up Digital Channels to use a Custom Gateway OR
- Set up Digital Channels to use Genesys Cloud CX Agreegation

Option 1: Set up Digital Channels to use SparkPost Email

Complete the steps in this section to set up Digital Channels to use SparkPost email provider.

1. Review the prerequisites table.
2. Use the Digital Channels provisioning API to create Digital Channels services definitions.
3. Provision email domains.

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
Contact Center Id	$ccid	UUID string	45acae06-6b7c-4f97-9c76-471cbd2bef7	This value comes from your Web Services and Applications deployment.
Digital Channels URL	$baseURL	string	http://digital.example.com	Publicly available URL for Digital Channels API.
SparkPost API URL	$sparkpostURL	string	Secret	URL for SparkPost email API.
SparkPost apikey	$sparkpostApikey	UUID string	Secret	API key for using SparkPost email API.

Create Digital Channels services definitions

In this step, you will enable Digital Channels to use SparkPost email services. You must configure the following services in Digital Channels within your tenant:

- Email - Enables outbound email service through the third-party provider.
- SparkPostEmail - Integrates your Genesys Multicloud CX tenant to SparkPost for email communication.

> **Important**
>
> Use a REST client or curl utility to provision the following services in Digital Channels using the provisioning API. Make sure to substitute the variables - prefixed with '$' - with their values. You must create these services once per tenant.

Create the **Email** service:

```
curl -X POST \ $nexusURL/nexus/v3/provisioning/services/$ccid/Email \
     -H 'Content-Type: application/json' \
     -H 'x-api-key: $apiKey' \
     -H 'x-ccid: $ccid' \ -d '{ "url" : "N/A", "secret": {}, "data" : { "provider":
"SparkPost" }'
```

Create the **SparkPostEmail** service:

```
curl -X POST \ $nexusURL/nexus/v3/provisioning/services/$ccid/SparkPostEmail \
     -H 'Content-Type: application/json' \
     -H 'x-api-key: $apiKey' \
     -H 'x-ccid: $ccid' \
     -d '{ "url" : "$sparkpostURL", "secret": { "apikey": "$sparkpostApikey"}, "data": {} }'
```

Manage email domains

Provision email domains in Digital Channels.

Option 2: Set up Digital Channels to use a Custom Gateway

Complete the steps in this section to set up Digital Channels to use a custom email gateway provider.

1. Review the prerequisites table.
2. Use the Digital Channels provisioning API to create Digital Channels services definitions.
3. Provision email domains.

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
Company's email domain	$asyncEmailDomain	string	company.com	
Third-party Messaging Webhook URL	$asyncWebhookURL	string	https://genesys-webhook.company.com	The FQDN of the third-party service implementing the Third-Party Messaging Webhook.
Third-Party Messaging API secret key	$asyncAPISignatureKey	string	Secret	The key used by the third-party service to calculate the signature for calls to the Third-Party Messaging API.
Third-Party Messaging Webhook secret key	$asyncWebhookSignatureKey	string	Secret	The key used by Digital Channels to calculate the signature for calls to the third-party service through the webhook.

Create Digital Channels services definitions

In this step, you will enable Digital Channels to use a custom gateway for the email provider of your choice. You must configure the following services in Digital Channels within your tenant:

- Async - Integrates your Genesys Multicloud CX tenant to the custom gateway used to communicate with the email provider of your choice.
- Email - Enables outbound email service through the third-party provider.

> **Important**
>
> Use a REST client or curl utility to provision the following services in Digital Channels using the provisioning API. Make sure to substitute the variables - prefixed with '$' - with their values. You must create these services once per tenant.

Create the **Async** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/Async \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
        "data": {
          "channels": [
            {
              "channelId": "$asyncPhoneNumber",
              "webhook": { "url": "$asyncWebhookURL" }
            },
            {
              "channelId": "$asyncEmailDomain",
              "webhook": { "url": "$asyncWebhookURL" }
            }
          ]
        },
        "secret": {
          "channels": [
            {
              "channelId": "$asyncPhoneNumber",
              "webhook": { "secret": "$asyncWebhookSignatureKey" },
              "api": { "secret": "$asyncAPISignatureKey" }
            },
            {
              "channelId": "$asyncEmailDomain",
              "webhook": { "secret": "$asyncWebhookSignatureKey" },
              "api": { "secret": "$asyncAPISignatureKey" }
            }
          ]
        }
      }'
```

Create the **Email** service:

```
curl -X POST \
```

```
  $nexusURL/nexus/v3/provisioning/services/$ccid/Email \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "N/A",
    "data" : { "provider": "Async" },
    "secret": {}
}'
```

Manage email domains

Provision email domains in Digital Channels.

Option 3: Set up Digital Channels to use Genesys Cloud CX Email Aggregation

Complete the steps in this section to set up Digital Channels to use Genesys Cloud CX Messaging Aggregation as the Email gateway.

1. Review the prerequisites table.
2. Contact your Genesys representative to create a Genesys Cloud CX organization and get administrator user credentials. Your Genesys representative also must add the Genesys Cloud CX product to your organization.
3. Create the Digital Channels integration in Genesys Cloud CX. This will give you a clientId and clientSecret to authenticate API calls with the Digital Channels provisioning API.
4. Create a Digital Channels API key for Genesys Cloud CX.
5. Use the Digital Channels provisioning API to create Digital Channels services definition.
6. Provision email domains.

Prerequisites

Review the **Prerequisites** table and make sure you have all the listed information before you get started. The values in this table are referenced later by the name in the Variable column.

Prerequisites

Parameter	Variable	Type	Example	Notes
GWS tenant Contact Center ID	$ccid	UUID string	45acae06-6b7c-4f97-9c76-471cbd2bef7	This value comes from your Web Services and Applications deployment.

Request a Genesys Cloud CX organization

Contact your Genesys representative to create a Genesys Cloud CX organization and get administrator user credentials. Your Genesys representative also must add the Genesys Cloud CX product to your organization.

Make sure your Genesys representative provides you with the details in the **Genesys Cloud CX information** table.

Genesys Cloud CX information

Parameter	Variable	Type	Example	Notes
Genesys Cloud CX Organization ID	$orgId	UUID string	47d8329d-1c28-4c86-9374-5596bddfee15	Your Genesys Cloud CX organization ID.
Genesys Cloud CX Organization admin user credentials	$orgUsername $orgPassword	string	admin-user / admin-password	The username and password for an account with administrative permissions for this organization.
Genesys Cloud CX Login URL	$gcLoginURL	HTTPS URL string	https://login.mypurecloud.com	Your Genesys Cloud CX login URL (depends on your organization region).
Genesys Cloud CX API URL	$gcAPIURL	HTTPS URL string	https://api.mypurecloud.com	Your Genesys Cloud CX login URL (depends on your

Parameter	Variable	Type	Example	Notes
				organization region).

Create the Digital Channels integration in Genesys Cloud CX

Complete the steps in this section as an administrator user in Genesys Cloud CX to create the integration client credentials that will be used by Digital Channels to access Genesys Cloud CX APIs to send and receive messages. You're going to create a new role, assign it to your admin user, and create the access credentials.

First, create the new role:

1. Navigate to $gcLoginURL (for example, https://login.mypurecloud.com) and log in to Genesys Cloud CX with your $orgUsername/$orgPassword.
2. Go to **Admin**.
3. Under **People and Permissions**, click **Roles/Permissions**.
4. Click **Add Role** and give it a name. For example, `Nexus Messaging`.
5. Under **Permissions**, search for **messaging** and select **messaging > All Permissions** and **messagingProvisioning > All Permissions**. Save your changes.

Next, assign the role to your administrator user:

1. Click **Admin**.
2. Under **People and Permissions**, click **People**.
3. Search for your admin user.
4. Under **Roles,** switch the view to **All** and search for the name of your new role (`Nexus Messaging`). Click to enable the role and then save your changes.
5. Log out and log in again to enable the permissions.

Now create access credentials for the Digital Channels integration.

1. Click **Admin**.
2. Under **Integrations**, click **OAuth**.
3. Click **Add Client.**
4. Under **Client Details**, set **App Name** to `Nexus Messaging Integration` and select the Client Credentials **Grant Type**.
5. Click **Roles** and assign the `Nexus Messaging` role. Save your changes.
6. Go back to **Client Details** and copy the values for **clientId** and **clientSecret**.

As the output of this step, you will have the access credentials:

Parameter	Variable	Type	Example
Nexus Integration client ID	$clientId	UUID string	4da40a9de-b113-4024-8ba9-c9dd89c91f67
Nexus Integration client secret	$clientSecret	string	aKSXEgLO57cm6FqxD4hrjkcW-iuWiXhd0uF0WOcZUm2

Create a Digital Channels API key for Genesys Cloud CX

To create an API key that will be used by Genesys Cloud CX to send requests to Digital Channels, follow the steps in Provision API keys. Make sure to use the following parameters:

```
"tenant": "*"
 "name":"Portico Cluster API Key"
 "permissions" : ["nexus:cluster:*"]
```

As an output of this step, you will have the API key:

Parameter	Variable	Type	Example
Messaging Cluster API Key	$apikey	UUID string	9b7682b7-cbce-422f-9bbb-ecda85e61695

Create Digital Channels services definitions

Enable Digital Channels to use Genesys Cloud CX Messaging Aggregation as the Email provider. You must configure the following services in Digital Channels within your tenant:

- PurecloudIDP - Integrates the Genesys Multicloud CX tenant to the Genesys Cloud CX organization.
- PorticoEmail - Enables the outbound email service through Genesys Messaging Aggregation.

> **Important**
>
> Use a REST client or curl utility to provision the following services in Digital Channels using the provisioning API. Make sure to substitute the variables - prefixed with

> '$' - with their values. Note: You must create these services once per tenant.

Create the **PurecloudIDP** service:

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/PurecloudIDP \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "$gcLoginURL",
    "secret": {"clientId": "$clientId", "clientSecret": "$clientSecret"},
    "data" : {}
}'
```

Create the **PorticoEmail** service if you use Genesys Multicloud CX Contact email campaigns.

```
curl -X POST \
  $nexusURL/nexus/v3/provisioning/services/$ccid/PorticoEmail \
  -H 'Content-Type: application/json' \
  -H 'x-api-key: $apiKey' \
  -H 'x-ccid: $ccid' \
  -d '{
    "url" : "$gcAPIURL",
    "secret": {},
    "data": {}
}'
```

Provision email domains in Digital Channels

Complete the steps in this section if you are integrating with CX Contact and plan to use email campaigns. These steps explain how to choose an email domain that you control (you should be able to update DNS record sets for this domain) and want to use in email campaigns as the "sent from" address.

If you are using Genesys Cloud CX as the provider, contact your Genesys representative to have them provision an email domain in Genesys Cloud CX for your organization. Once completed, you will receive a set of secrets you must use to update your domains records. After this update, contact your Genesys representative to validate the secrets and confirm domain ownership. Now you can provision an email service channel.

Complete the following for each domain only after your domain records have been updated, validated, and provisioned in Genesys Cloud CX.

1. In Agent Setup, navigate to the "NexusServices" transaction you created previously.
2. Create a section that starts with 'cxc.' and includes some text that represents the domain. For example, `cxc.Coraporate_Promotions`. Note: The 'cxc.' prefix is required and the rest of the value can be made up of letters and underscore or dashes, but not spaces.
3. In this new section, create the following options:
   - channelId = **$emailDomain**
   - channelType = email
   - provider = Async (Only set this option if you use a custom gateway as the email aggregator.)

About AI Connector

Contents

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

Related documentation:

-
-
-
-

RSS:

- For private edition

The AI Connector interfaces with third-party natural language processing and virtual agent frameworks to stream audio and chat messages, and then deliver the results to the Genesys agent desktop or bot conversations. It makes the interface available as an HTTP webhook endpoint where voice and digital channels can send interaction events.

> **Important**
> The term “Athena” is used in some places in this article. It is the simplified name used for the AI Connector application and nodes.

Supported Kubernetes platforms

AI Connector is supported on the following cloud platform:

- Google Kubernetes Engine (GKE)

Before you begin

Contents

Find out what to do before deploying AI Connector.

Related documentation:

-
-
-
-

RSS:

- For private edition

Limitations and assumptions

AI Connector for private edition has the following limitation:

- Supports only a single-region model of deployment.

Download the Helm charts

AI Connector in Genesys Multicloud CX private edition includes the following container:

- athena

The service also includes a Helm chart, which you must deploy to install all the containers for AI Connector:

- athena

See Helm charts and containers for Digital Channels for the Helm chart version you must download for your release.

To download the Helm chart, navigate to the **athena** folder in the JFrog repository. For information about how to download the Helm chart, see Downloading your Genesys Multicloud CX containers.

Third-party prerequisites

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

Third-party services

Name	Version	Purpose	Notes
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.	AI Connector supports Redis deployed in either cluster (TLS) or non-cluster (non-TLS) mode.
PostgreSQL	11.x	Relational database.	No support for enforced SSL.
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.	

Name	Version	Purpose	Notes
Command Line Interface		The command line interface tools to log in and work with the Kubernetes clusters.	

Storage requirements

AI Connector uses PostgreSQL and Redis to store all data.

Network requirements

For general network requirements, review the information on the suite-level Network settings page.

Genesys dependencies

AI Connector has dependencies on the following Genesys service:

- Digital Channels

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

Configure AI Connector

Contents

Complete the steps on this page to configure your AI Connector deployment.

Related documentation:

-
-
-
-

RSS:

- For private edition

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.

Parameters

Parameter	Description	Valid values	Default
version	The AI Connector version.	A valid release version	nil
nameOverride	A string to partially override the athena.fullname template. This string is prepended to the release name.	String	nil
fullnameOverride	A string to fully override the athena.fullname template.	String	nil
image.registry	The AI Connector image registry.	A valid registry URL	nil
image.repository	The AI Connector image name.	A valid image name	nexus/athena
image.pullPolicy	Specifies when Kubernetes pulls images from the registry on start up.	IfNotPresent or Always	IfNotPresent

Parameter	Description	Valid values	Default
image.pullSecrets	An array of docker-registry secret names.	An array of secret names	[] (does not add image pull secrets to deployed pods)
serviceAccount.create	Specifies whether a service account must be created.	false or true	false
serviceAccount.name	The name of the service account to use. If this is not set and create is true, a name is generated using the fullname template.	String	""
serviceAccount.annotations	Annotations to add to the service account.	A valid set of labels as "name: value"	{}
podAnnotations	Custom annotations for each pod.	A valid set of labels as "name: value"	{}
podSecurityContext.runAsNonRoot	Specifies whether the container must run as a non-root user.	true or false	true
podSecurityContext.runAsUser	The user ID to run the entry point of the container process.	A valid user ID	500
podSecurityContext.runAsGroup	The group ID to run the entry point of the container process.	A valid group ID	500
podSecurityContext.fsGroup	A supplemental group ID that applies to all containers in a pod.	A valid group ID	500
configChecksum	Adds SHA-256 checksum of the ConfigMap to the deployment annotations.	true of false	true
secretChecksum	Adds SHA-256 checksum of the Secret to the deployment annotations.	true or false	true
containerPort	TCP port the service is listening on.	A valid port	4084
service.enabled	Enables the Kubernetes service.	true or false	true
service.type	The Kubernetes service type.	See the Kubernetes documentation for details.	ClusterIP
service.annotations	The service annotations.	A valid set of annotations as "name: value"	{}
service.port	The Kubernetes service	A valid port	80

Parameter	Description	Valid values	Default
	HTTP port.		
ingress.enabled	Enables the ingress controller resource.	true or false	false
ingress.annotations	The ingress annotations.	A valid set of annotations as "name: value"	[]
ingress.hosts[0].host	The hostname of your AI Connector installation.	A valid hostname	athena.local
ingress.hosts[0].paths	The paths (within the URL structure) to your AI Connector.	A valid list of paths	[]
ingress.tls[0].secretName	Kubernetes secret name with server.crt certificate and server.key private key file (Only required if you want to configure TLS for ingress resources).	A valide name	nil
ingress.tls[0].hosts	The hostname used to generate cert that matches the TLS certificate (required if you want to configure TLS for ingress resources).	A valid hostname	nil
resources	The requests and limits for CPU and memory usage in Kubernetes. See the Kubernetes documentation for details.	Object	{}
nodeSelector	The labels Kubernetes uses to assign pods to nodes. See the Kubernetes documentation for details.	Object	{}
tolerations	The tolerations Kubernetes uses for advanced pod scheduling. See the Kubernetes documentation for details.	Object	{}
affinity	Specifies the affinity and anti-affinity for Digital Channels pods. See the Kubernetes documentation for details.	Object	{}

Parameter	Description	Valid values	Default
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	""
monitoring.enabled	Specifies whether to deploy Custom Resource Definitions (CRD) for ServiceMonitors to determine which services should be monitored.	true or false	false
athena.server.apiPrefix	The prefix for all API endpoints exposed by the service.	String or comma-separated list of strings	/athena/v1,/nexus/v3
athena.nexus.url	The Nexus service URL.	A valid URL	http://nexus-production.nexus.svc.cluster.local
athena.nexus.apiPrefix	The prefix of Nexus API endpoints.	String	/nexus/v3
athena.nexus.apiKey	A Nexus API key used by AI Connector to fetch tenant settings and services from Nexus. See the Digital Channels documentation for details.	A valid cluster Nexus API key with permissions "nexus:cluster:*" assigned	""
athena.nexus.timeout	Timeout for requests to Nexus (access tokens validation, configuration retrieval).	Integer	10000
athena.db.host	The Postgres service URL.	A valid URL	postgres
athena.db.port	The Postgres service port.	A valid port	5432
athena.db.user	The user assigned for the AI Connector application to access Postgres.	A valid user	nexus
athena.db.password	The password assigned for the AI Connector application to access Postgres.	A valid password	""
athena.db.database	The database for the AI Connector application to use in Postgres.	A valid database	nexus

Parameter	Description	Valid values	Default
athena.db.ssl	Use secured connection to Postgres.	true or false	false
athena.redis.nodes	A comma-separated list of Redis nodes to connect.	A valid URL	redis://redis:6379
athena.redis.password	The password for Redis authentication.	A valid password	""
athena.redis.cluster	Specifies whether to deploy Redis as a cluster.	true or false	false
athena.redis.tls	Specifies whether to use TLS on the Redis connection.	true or false	false
athena.google.speechApiKey	API key for Google speech recognition API used to transcribe speech to text for bot providers that do not natively support voice input (LUIS, DialogEngine), not needed for Dialogflow ES, CX and Lex.	A valid Google speech recognition API key	""

Configure security

To learn more about how to configure security for private edition, be sure to read Permissions and OpenShift security settings.

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.

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

Deploy

Contents

Learn how to deploy 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.

Deploy AI Connector

> **Important**
> Make sure to review Before you begin for the full list of prerequisites required to deploy Digital Channels AI Connector.

Prepare your environment

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

GKE

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

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

Create a JSON file called **create-nexus-namespace.json** with the following content:

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

Use the JSON file to create a new namespace for Digital Channels AI Connector:

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

Now, confirm the created namespace:

```
kubectl describe namespace athena
```

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=
```

Deploy

To deploy AI Connector, you'll need the Helm package and override files you downloaded in a previous step. Copy **values.yaml** and the Helm package (**athena-.tgz**) to the installation location.

You must override the following key sections in **values.yaml:**

- image.*
- athena.nexus.*
- athena.redis.*
- athena.db.*
- ingress.*

Here's an example of how your **values.yaml** file might look:

```
# Default values for athena.
```

```
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

version: "100.0.124.3419" # AI Connector Version

nameOverride: ""

fullnameOverride: ""

replicaCount: 1

image:
  registry: "pureengage-docker-staging.jfrog.io"
  repository: nexus/athena
  pullPolicy: IfNotPresent
  pullSecrets:
    - name:

serviceAccount:
  create: false
  name: ""
  annotations: {}

podAnnotations: {}

podLabels: {}

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

securityContext: {}

configChecksum: true

secretChecksum: true

containerPort: 4084

service:
  enabled: true
  type: ClusterIP
  annotations: {}
  port: 80

ingress:
  enabled: false
  annotations: {}
  hosts:
    - host: athena.local
      paths: []
  tls: []
  #  - secretName: athena-tls-secret
  #    hosts:
  #      - athena.local

resources: {}
  # limits:
  #   cpu: 100m
  #   memory: 128Mi
  # requests:
```

```
  #   cpu: 100m
  #   memory: 128Mi

nodeSelector: {}

tolerations: []

affinity: {}

priorityClassName: ""

dnsPolicy: ClusterFirst

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

monitoring:
  enabled: false

athena:
  server:
    apiPrefix: "/nexus/v3"
  nexus:
    url: ""
    apiPrefix: "/nexus/v3"
    apiKey: ""
    timeout: 10000
  db:
    host: ""
    port: 5432
    user: ""
    password: ""
    database: ""
    ssl: false
  redis:
    nodes: "redis://:6379"
    password: ""
    cluster: true
    tls: false
  google:
    speechApiKey: ""
```

Run the following command to install AI Connector:

```
helm upgrade --install  /athena-.tgz -f values.yaml
```

Validate the deployment

To validate the deployment, first run the following code snippet

```
kubectl port-forward service/athena :80
```

Then, send the GET request on the following URL:

```
$athenaURL/health/detail
```

where **$athenaURL** is the fully qualified domain name (FQDN) for AI Connector.

The response should look like this:


```
{
    "buildInfo": {
        "version": "100.0.001.97446",
        "changeset": "565f432fa8f4555276b55e8237cebcfb201b986e",
        "timestamp": "Mon Jan 17 10:21:22 UTC 2022"
    },
    "startTime": "2022-03-17T13:15:22.873Z",
    "upTime": 49338032,
    "os": {
        "hostname": "athena-6bb9c5c68f-bz449",
        "upTime": 52366.39,
        "freemem": 1316397056,
        "loadavg": [0.35, 0.37, 0.63],
        "totalmem": 4124729344
    },
    "memoryUsage": {
        "rss": 178757632,
        "heapTotal": 83382272,
        "heapUsed": 80987072,
        "external": 1890524,
        "arrayBuffers": 126610
    },
    "redis": {
        "state": "READY",
        "latency": 5
    },
    "db": {
        "latency": 202
    },
    "isReady": true
}
```


The deployment is successful if `state="green"`. You can also confirm that `db.ready=true` and `redis.ready=true`.

Uninstall

Execute the following command to uninstall AI Connector:

```
helm delete  -n
```

Next steps

Complete the steps in Provisioning overview to finish deploying AI Connector.

Provisioning overview

Contents

Learn about the steps involved in provisioning Digital Channels AI Connector.

Related documentation:

-
-
-
-

RSS:

- For private edition

Provisioning AI Connector webhook to Digital Channels

To receive and analyse chat messages, AI Connector functions as a webhook for interactions in Digital Channels. As Digital Channels is unaware of where AI Connector is available, register AI Connector with Digital Channels. This process includes two parts:

1. Verify AI Connector Webhook
2. Registering Webhook to Digital Channels

Verifying AI Connector webhook

AI Connector webhook is deployed at **`${CONF.athena.server.apiPrefix}/agent-assist-chat/messages`**

Verifying the webhook returns the HTTP status code as **200** with status ***PAYLOAD_SIGNATURE_CHECK_FAILED***

Registering webhook to Digital Channels

The webhook target name for Digital Channels is **ai-connector**

It is a one-time process (per contact center) required to ensure that the AI Connector webhook URL is known and trusted to Digital Channels service. For registering the AI Connector webhook, send a request with the admin-level access credentials to Digital Channels.

Request:

```
curl --request POST 'https://{nexusUrl}/nexus/v3/provisioning/
services/{ccid}/WebhookTransportTarget-ai-connector' \
--header 'Authorization: Bearer {accessToken}' \
--header 'Content-Type: application/json' \
--data '{
```

```
    "url": "{AI Connector webhook URL}"
}'
```

Observability in Digital Channels

Contents

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

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 Digital Channels metrics, see:

-

See also System metrics.

Enable monitoring

Digital Channels uses ServiceMonitor custom resource definitions (CRDs) and it also supports custom annotations for pods. To enable monitoring for Digital Channels, set the **monitoring.enabled** Helm chart parameter to `true`. To set custom annotations, use the **podAnnotations** Helm chart parameter. See Override Helm chart values for more information about these parameters.

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
	Both — ServiceMonitor and annotations	4004	nexus.nexus.svc.cluster.local/ metrics	15 seconds

Configure metrics

The metrics that are exposed by Digital Channels 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 Digital Channels exposes. You can also query Prometheus directly or via a dashboard to see all the metrics available from Digital Channels.

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 Digital Channels alerts, see:

-

Configure alerts

Private edition services define a number of alerts by default (for Digital Channels, 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, Digital Channels does not support customizing the alerts or defining additional **PrometheusRule** objects to create alerts based on the service-provided metrics.

Logging

Digital Channels 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 the logging overview in the Operations guide.

Digital Channels metrics and alerts

Contents

Find the metrics *No results* exposes and the alerts defined for *No results*.

Related documentation:

-

Service	CRD or annotations?	Port	Endpoint/Selector	Metrics update interval
Digital Channels	Both — ServiceMonitor and annotations	4004	nexus.nexus.svc.cluster.local/metrics	15 seconds

See details about:

- Digital Channels metrics
- Digital Channels alerts

Metrics

Digital Channels exposes many Genesys-defined 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 Digital Channels metrics not documented on this page.

Metric and description	Metric details	Indicator of
nexus_errors_total The total number of requests that resulted in an error.	**Unit:** **Type:** Number **Label:** **Sample value:** 100	
nexus_request_total The total number of requests.	**Unit:** **Type:** Number **Label:** **Sample value:** 1000	
nexus_process_resident_memory_bytes The total byes of memory Digital Channels consumed.	**Unit:** **Type:** Number **Label:** **Sample value:** 100000	
nexus_redis_connections_established The current number of established Redis	**Unit:** **Type:** Gauge	

Metric and description	Metric details	Indicator of
connections.	**Label:** **Sample value:** 0	
nexus_redis_connections_reconnecting The current number of reconnecting Redis connections.	**Unit:** **Type:** Gauge **Label:** **Sample value:** 0	
nexus_redis_connections_ready The current number of ready Redis connections.	**Unit:** **Type:** Gauge **Label:** **Sample value:** 1	
nexus_redis_duration_until_ready The duration until Redis reaches the ready state.	**Unit:** **Type:** Histogram **Label:** 'le' **Sample value:** 0, 1, 39	
nexus_redis_errors_total The total number of Redis connection errors.	**Unit:** **Type:** Counter **Label:** **Sample value:** 0	
nexus_db_connect_total The total number of all database connection requests.	**Unit:** **Type:** Counter **Label:** 'db' **Sample value:** 1252424, 1457770	
nexus_db_disconnect_total The total number of all database disconnection requests.	**Unit:** **Type:** Counter **Label:** 'db' **Sample value:** 1252424, 1457770	
nexus_db_request_total The total number of all database requests sent.	**Unit:** **Type:** Counter **Label:** 'db' **Sample value:** 4850730, 5056452	
nexus_db_success_total The total number of all database requests executed successfully.	**Unit:** **Type:** Counter **Label:** 'db', 'command' **Sample value:** 2307896, 2126805, 1221394, 1450355	
nexus_db_errors_total The total number of all database errors.	**Unit:** **Type:** Counter **Label:** 'db', 'code' **Sample value:** 131, 5, 4	
nexus_db_request_duration_milliseconds	**Unit:**	

Metric and description	Metric details	Indicator of
The database transaction duration.	**Type:** histogram **Label:** 'le', 'db', 'method' **Sample value:** 2290844, 2306385, 2307241, 2307894	
iwd_process_cpu_user_seconds_total The total user CPU time spent, in seconds.	**Unit:** **Type:** counter **Label:** **Sample value:** 1634045655571	
nexus_process_cpu_system_seconds_total The total system CPU time spent, in seconds.	**Unit:** **Type:** counter **Label:** **Sample value:** 1634045655571	
nexus_process_cpu_seconds_total The total user and system CPU time spent, in seconds.	**Unit:** **Type:** counter **Label:** **Sample value:** 1634045655571	
nexus_process_start_time_seconds The start time of the process since the Unix epoch, in seconds.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1633992102	
nexus_process_resident_memory_bytes The resident memory size, in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_process_virtual_memory_bytes The virtual memory size, in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_process_heap_bytes The process heap size, in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_process_open_fds The number of open file descriptors.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_process_max_fds The maximum number of open file descriptors.	**Unit:** **Type:** gauge **Label:** **Sample value:** 197176	
iwd_nodejs_eventloop_lag_seconds	**Unit:**	

Metric and description	Metric details	Indicator of
The Node.js event loop lag, in seconds.	**Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_nodejs_active_handles The number of active libuv handles, grouped by handle type. Every handle type is a C++ class name.	**Unit:** **Type:** gauge **Label:** 'type' **Sample value:** 17, 1, 69	
nexus_nodejs_active_handles_total The total number of active libuv handles.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_nodejs_active_requests The number of active libuv requests, grouped by request type. Every request type is a C++ class name.	**Unit:** **Type:** gauge **Label:** 'type' **Sample value:** 2	
nexus_nodejs_active_requests_total The total number of active libuv requests.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_nodejs_heap_size_total_bytes The process heap size from Node.js, in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_nodejs_heap_size_used_bytes The process heap size used from Node.js, in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_nodejs_external_memory_bytes The Node.js external memory size, in bytes.	**Unit:** **Type:** gauge **Label:** **Sample value:** 1634045655572	
nexus_nodejs_heap_space_size_total_bytes The process heap space size total from Node.js, in bytes.	**Unit:** **Type:** gauge **Label:** 'space' **Sample value:** 262144, 16777216, 130428928, 6721536	
nexus_nodejs_heap_space_size_used_bytes The process heap space size used from Node.js in bytes.	**Unit:** **Type:** gauge **Label:** 'space' **Sample value:** 32808, 1479672, 92634792, 4852384	
nexus_nodejs_heap_space_size_available_bytes	**Unit:**	

Metric and description	Metric details	Indicator of
The process heap space size available from Node.js, in bytes.	**Type:** gauge **Label:** 'space' **Sample value:** 0, 6899976, 37040456, 1542496	
nexus_nodejs_version_info Node.js version information.	**Unit:** **Type:** gauge **Label:** 'version', 'major', 'minor', 'patch' **Sample value:** 1	

Alerts

The following alerts are defined for Digital Channels.

Alert	Severity	Description	Based on	Threshold
Nexus error rate	Critical	Triggered when the error rate on this pod is greater than 20% for 15 minutes.	nexus_errors_total, nexus_request_total	For 15 minutes
Memory usage is above 3000 Mb	Critical	Triggered when the memory usage on this pod is above 3000 Mb for 15 minutes.	nexus_process_resident_memory_bytes	For 15 minutes