IWD Data Mart Private Edition Guide

7/26/2024

Table of Contents

Contents

Find links to all the topics in this guide.

Related documentation:

-
-
-

RSS:

- For private edition

IWD Data Mart (IWDDM) is a service available with the Genesys Multicloud CX private edition offering.

Overview

Learn more about IWD Data Mart, its architecture, and how to support high availability and disaster recovery.

- About IWD Data Mart
- Architecture
- High availability and disaster recovery

Configure and deploy

Find out how to configure and deploy IWD Data Mart.

- Before you begin
- Configure IWD Data Mart
- Provision IWD Data Mart
- Deploy IWD Data Mart

Upgrade, roll back, or uninstall

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

- Upgrade, roll back, or uninstall

Observability

Learn how to monitor IWD Data Mart with metrics and logging.

- IWDDM metrics and alerts

About IWD Data Mart

Contents

Learn about iWD Data Mart and how it works in Genesys Multicloud CX private edition.

Related documentation:

-
-
-
-

RSS:

- For private edition

IWD Data Mart offers comprehensive reporting, providing management insight into business operation. It provides key indicators of performance both through current-day statistics and on an historical basis.

IWD Data Mart retrieves reporting events from Intelligent Workload Distribution (IWD) and populates task fact tables, dimensions, and aggregate tables in IWD Data Mart DB. Data from IWD Data Mart is displayed using Genesys Customer Experience Insights (GCXI) IWD reports.

Supported Kubernetes platforms

IWD DM service is supported on the following cloud platforms:

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

See the Intelligent Workload Distribution Release Notes for information about when support was introduced.

Architecture

Learn about the iWD Data Mart's architecture.

Related documentation:

-
-
-
-

RSS:

- For private edition

The architecture diagram is common for IWD, IWD Data Mart, and Email. For illustration on a sample deployment of IWD Data Mart, see IWD Architecture.

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

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

IWD Data Mart is a Cronjob that runs on a per-tenant basis, so High Availability (HA) is not applicable.

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

High Availability of IWD Data Mart is based on the CronJob's limited amount of working time and fact that no data is destroyed even if the IWD Data Mart job failed in progres.

Before you begin

Contents

Find out what to do before deploying IWD Data Mart.

Related documentation:

-
-
-
-

RSS:

- For private edition

Limitations and assumptions

The current version of IWD Data Mart:

- works as a short-living job started on schedule
- does not support scaling or HA
- requires dedicated PostgreSQL deployment per customer

IWD Data Mart is a short-living job, so Prometheus metrics cannot be pulled. Therefore, it requires a standalone Pushgateway service for monitoring.

Download the Helm charts

IWD Data Mart in Genesys Multicloud CX private edition includes the following containers:

- iwd_dm_cloud

The service also includes a Helm chart, which you must deploy to install the required containers for IWD Data Mart:

- iwddm-cronjob

See Helm Charts and Containers for IWD and IWD Data Mart for the Helm chart version you must download for your release.

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

Third-party prerequisites

Third-party services

Name	Version	Purpose	Notes
PostgreSQL	11.x	Relational database.	
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.	

Storage requirements

All data is stored in PostgreSQL, which is external to the IWD Data Mart.

Network requirements

Not applicable

Browser requirements

Not applicable

Genesys dependencies

Intelligent Workload Distribution (IWD) with a provisioned tenant.

For the order in which the Genesys services must be deployed, refer to the Order of services deployment topic in the *Setting up Genesys Multicloud CX private edition* document.

GDPR support

Content coming soon

Configure IWD Data Mart

Contents

Learn how to configure IWD Data Mart.

Related documentation:

-
-
-
-

RSS:

- For private edition

Override Helm chart values

The following table provides information on the IWD Data Mart deployment settings:

Key	Type	Default	Description
image.imagePullSecrets	list	`[]`	imagePullSecrets must have the following format: - name: pullSecret1 - name: pullSecret2
image.pullPolicy	string	`"IfNotPresent"`	Image pull policy
image.registry	string	`""`	Image registry
image.repository	string	`""`	Image repository
image.tag	string	`""`	Image tag
iwddm.cronjob.schedule	string	`"*/15 * * * *"`	IWDDM cronjob start schedule in cron format
iwddm.cronjob.suspend	bool	`false`	IWDDM suspend: if true, it won't start
iwddm.db.createConfigmap	bool	`false`	Create a config map for db access parameters
iwddm.db.dbname	string	`nil`	IWDDM database name
iwddm.db.host	string	`nil`	IWDDM database hostname
iwddm.db.port	int	`5432`	IWDDM database port
iwddm.db.secret.enabled	bool	`false`	Create IWDDM database secret
iwddm.db.secret.password	string	`nil`	IWDDM database password to put in the secret

Key	Type	Default	Description
iwddm.db.secret.secretName	string	`nil`	IWDDM database secret name
iwddm.db.user	string	`nil`	IWDDM database user
iwddm.env.batchSize	int	`10000`	Events load batch size
iwddm.env.executionChain	string	`"full"`	
iwddm.env.httpProxyUrl	string	`""`	IWDDM will try to connect to IWD using http proxy
iwddm.env.httpsProxyUrl	string	`""`	IWDDM will try to connect to IWD using https proxy
iwddm.env.monitoring.enabled	bool	`false`	Enable push metrics to Pushgateway
iwddm.env.monitoring.pushgateway_url	string	`""`	Since IWDDM is a short living cronjob, it uses Pushgateway for providing metrics
iwddm.env.restUrl	string	`""`	IWD REST endpoint url
iwddm.env.wfm	object	`{"enabled":false}`	Push data to WFM if enabled
iwddm.helmTest.args	list	`["export PGPASSWORD=${IWDDM_PASSWORD};psql -c \"select now()\""]`	Arguments passed to the helm test command
iwddm.helmTest.command	list	`["/bin/bash","-c"]`	Command for helm test
iwddm.helmTest.image	string	`"postgres:latest"`	Image for helm test
iwddm.labels	object	`{}`	Extra labels ref: https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/
iwddm.nodeSelector	object	`{}`	Node labels for assignment. ref: https://kubernetes.io/docs/user-guide/node-selection/
iwddm.priorityClassName	string	`""`	Priority Class ref: https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption/
iwddm.resources.limits.cpu	string	`"100m"`	IWDDM Kubernetes CPU limit
iwddm.resources.limits.memory	string	`"160Mi"`	IWDDM Kubernetes

Key	Type	Default	Description
			memory limit
iwddm.resources.requests.cpu	string	`"50m"`	IWDDM Kubernetes CPU request
iwddm.resources.requests.memory	string	`"128Mi"`	IWDDM Kubernetes memory request
iwddm.securityContext	object	`{}`	Security Context ref: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/#set-the-security-context-for-a-container Containers should run as genesys user and cannot use elevated permissions **Note:** These options must not be changed unless instructed by Genesys.
iwddm.serviceAccount.create	bool	`true`	Create service account for IWDDM
iwddm.serviceAccount.name	string	`nil`	The name of the ServiceAccount to use. If not set and create is true, a name is generated using the fullname template
iwddm.tenantId	int	`1`	Tenant short 4-digit ID
iwddm.volumeMounts	object	`{}`	Volumes mounted into an IWDDM container
iwddm.volumes	string	`nil`	Must be declared with starting |- since they are parsed as a template
podAnnotations	object	`{}`	Add annotations to all pods
podLabels	object	`{}`	Add labels to all pods

Configure Kubernetes

ConfigMaps

Not applicable as all required ConfigMaps are created via Helm Chart basing on the provided values.

Secrets

IWD Data Mart requires several secrets to be created.

Create the pull secret

Use the following code snippet as an example of how to create pull secret:

```
kubectl create secret docker-registry mycred --docker-server=pureengage.jfrog.io --docker-
username= --docker-password=
```

You can add *mycred* to Helm override values by setting **image.imagePullSecrets** to *[mycred]*.

IWDDM PostgreSQL password

It can be configured in Helm values as shown in the sample:

```
iwddm:
  db:
    secret:
      enabled: true
      secretName: iwddm-db-secret-0001
      password: somePassword1
  volumes: |-
    - name: iwddm-db-secrets
      secret:
        secretName: iwddm-db-secret-0001

  volumeMounts: |-
    - name: iwddm-db-secrets
      readOnly: true
      mountPath: "/mnt/env-file-secrets/db-file-secrets"
```

IWDDM iWD x-api-key

It must exist before your IWDDM deployment. It must be created with the following format:

```
IWDDM_API_KEY=value
```

Sample password creation:

```
kubectl create secret generic 0001-iwd-secrets \
--from-literal='IWDDM_API_KEY=123456-abcde-4568734'
```

IWD x-api-key can be configured as shown below:

```
iwddm:
  volumes: |-
    - name: iwd-secrets
      secret:
        secretName: 0001-iwd-secrets

volumeMounts: |-
    - name: iwd-secrets
      readOnly: true
      mountPath: "/mnt/env-file-secrets/iwd-file-secrets"
```

GIM database password

Create a multi-line secret in the following way:

Create a file, **gim-secret.txt** with the following text:

```
IWDDM_GIM_DBUSER=
IWDDM_GIM_PASSWORD=
IWDDM_GIM_URL=jdbc:postgresql://:5432/
```

Create a secret from the file:

```
kubectl create secret generic gim-secrets \
--from-file=gim-secret.txt
```

Then, mount it using Helm values:

```
iwddm:
  volumes: |-
    - name: gim-secrets
      secret:
        secretName: gim-secrets

volumeMounts: |-
    - name: gim-secrets
      readOnly: true
      mountPath: "/mnt/gim-secrets/gim-secret"
```

Configure security

IWDDM needs the username and password provided for PostgreSQL access.

Configure IWD REST endpoint

IWD REST URL provided for IWDDM in the **iwddm.env.restUrl** parameter must be:

```
http://iwd.iwd.svc.cluster.local:4024/iwd/v3
```

The above endpoint may change depending on the configuration and where IWD is installed. Refer to IWD documentation for information on this endpoint.

Configure connection to WFM

IWD DM can connect and push metrics to Workforce Management (WFM).

The following Helm values must be configured:

Helm key	Sample value
iwddm.env.wfm.enabled	true
iwddm.env.wfm.health_url	http://wfm-t-backend.wfm.svc.cluster.local.:7010/wfm/api/v3
iwddm.env.wfm.rest_url	http://wfm-t-backend.wfm.svc.cluster.local.:7010/?Handler=DISCO
iwddm.env.wfm.gauth.enabled	true
iwddm.env.wfm.gauth.rest_url	http://gauth-auth-active.gauth.svc.cluster.local.:80
iwddm.env.wfm.gauth.client_id	
iwddm.env.wfm.gauth.client_secret	valueFrom: secretKeyRef: key: name:

Provision IWD Data Mart

- Administrator

Learn how to provision IWD Data Mart.

Related documentation:

-
-
-
-

RSS:

- For private edition

Create a Postgres database for IWDDM.

Create a user for IWDDM full access to database:

```
CREATE USER "iwddm_dm" WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT NOREPLICATION
CONNECTION LIMIT -1 PASSWORD '';
```

IWDDM should provide access to its database to GCXI. For that, an additional user account can be created with the following script:

```
CREATE USER "iwddm_gcxi" WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT
NOREPLICATION CONNECTION LIMIT -1 PASSWORD '';
ALTER DEFAULT PRIVILEGES FOR USER iwddm_dm GRANT SELECT ON TABLES TO iwddm_gcxi;
```

After creating an account, the new user account can be provided to the GCXI service for database access.

Deploy IWD Data Mart

Contents

Learn how to deploy IWD Data Mart (IWDDM) into a private edition environment.

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
>
> - Make sure to review Before you begin for the full list of prerequisites required to deploy IWD Data Mart.
> - The sample code snippets and install commands in this document use an example version of IWD Data Mart. Ensure that you replace the example version with the version that is applicable for your deployment.

Kubernetes

Prepare

1. Create a new project using the following command:

   ```
   kubectl create namespace iwddm
   ```

2. Create a pull secret for accessing the JFrog registry. See Configure Kubernetes.
3. Download the IWD helm chart from the JFrog repository. See Download the Helm charts.
4. IWD Data Mart requires the Digital Channels API key. The key must be provisioned and shared via Digital Channels or IWD. See IWD x-api-key.

Deploy

1. Extract parameters from chart to see multiple (default) values used to fine tune the installation.

   ```
   $ helm show values /iwddm- > values.yaml
   ```

2. Set up essential IWDDM Helm values:
   - image.registry
   - image.imagePullSecrets (if needed)
   - image.repository
   - image.tag
   - image.repository
   - iwddm.tenantId
   - iwddm.db.*
   - iwddm.db.secret.*
   - iwddm.volumes
   - iwddm.volumeMounts
   - iwddm.env.gim.enabled: true (given that GIM DB secret is provided)
     Use the sample override file:

```
image:
  registry: "pureengage-docker-staging.jfrog.io"
  repository: "iwddm/iwd_dm_cloud"
  tag: ""
  pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: pullsecret
iwddm:
  tenantId:  #sample 100
  db:
    createConfigmap: true
    host:
    port: 5432
    dbname:
    user:
    secret:
      enabled: true
      secretName:
      password:
  cronjob:
    schedule: "*/3 * * * *"
    suspend: false
  securityContext: {}
  env:
    executionChain: "full"
    restUrl: "http://iwd.iwd.svc.cluster.local:4024/iwd/v3"
    monitoring:
      enabled: false
      pushgateway_url: ""
  volumes: |-
    - name: iwddm-db-secrets
      secret:
        secretName:
    - name: iwd-secrets
      secret:
        secretName:
  volumeMounts:
    iwddm-db-secrets:
      readOnly: true
      mountPath: "/mnt/env-secrets/db-secrets"
```

```
          iwd-secrets:
            readOnly: true
            mountPath: "/mnt/env-secrets/iwd-secrets"
```

3. Install IWD Data Mart using the following command:

```
helm upgrade --install iwddm-{short_tenant_id} /iwddm-cronjob --version={version} -f
 ./values.private.yml
```

Google Kubernetes Engine (GKE)

Prepare

1. Log in to the GKE cluster.

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

2. Create a new project:
    1. Create a *create-iwddm-namespace.json* :

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

    2. Create a namespace using the above JSON:

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

    3. Confirm the namespace creation:

```
kubectl describe namespace iwddm
```

3. Create a pull secret for accessing the JFrog registry. See Configure Kubernetes.
4. Download the IWD helm chart from the JFrog repository. See Download the Helm charts.
5. IWD Data Mart requires the Digital Channels API key. The key must be provisioned and shared via Digital Channels or IWD. See IWD x-api-key.

Deploy

1. Extract parameters from chart to see multiple (default) values used to fine tune the installation.

```
$ helm show values /iwddm- > values.yaml
```

2. Set up essential IWDDM Helm values:
   - image.registry
   - image.imagePullSecrets (if needed)
   - image.repository
   - image.tag
   - image.repository
   - iwddm.tenantId
   - iwddm.db.*
   - iwddm.db.secret.*
   - iwddm.volumes
   - iwddm.volumeMounts
   - iwddm.env.gim.enabled: true (given that GIM DB secret is provided)
     Use the sample override file:

```
image:
  registry: "pureengage-docker-staging.jfrog.io"
  repository: "iwddm/iwd_dm_cloud"
  tag: ""
  pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: pullsecret
iwddm:
  tenantId:  #sample 100
  db:
    createConfigmap: true
    host:
    port: 5432
    dbname:
    user:
    secret:
      enabled: true
      secretName:
      password:
  cronjob:
    schedule: "*/3 * * * *"
    suspend: false
  securityContext: {}
  env:
    executionChain: "full"
    restUrl: "http://iwd.iwd.svc.cluster.local:4024/iwd/v3"
    monitoring:
      enabled: false
      pushgateway_url: ""
  volumes: |-
    - name: iwddm-db-secrets
      secret:
        secretName:
    - name: iwd-secrets
      secret:
        secretName:
  volumeMounts:
    iwddm-db-secrets:
      readOnly: true
      mountPath: "/mnt/env-secrets/db-secrets"
```

```
          iwd-secrets:
            readOnly: true
            mountPath: "/mnt/env-secrets/iwd-secrets"
```

3. Install IWD Data Mart using the following command:

```
helm upgrade --install iwddm-{short_tenant_id} /iwddm-cronjob --version={version} -f
 ./values.private.yml
```

Azure Kubernetes Service (AKS)

Prepare

1. Log in to the AKS cluster.

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

2. Create a new project:
    1. Create a *create-iwddm-namespace.json* :

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

    2. Create a namespace using the above JSON:

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

    3. Confirm the namespace creation:

```
kubectl describe namespace iwddm
```

3. Create a pull secret for accessing the JFrog registry. See Configure Kubernetes.
4. Download the IWD helm chart from the JFrog repository. See Download the Helm charts.
5. IWD Data Mart requires the Digital Channels API key. The key must be provisioned and shared via Digital Channels or IWD. See IWD x-api-key.

Deploy

1. Extract parameters from chart to see multiple (default) values used to fine tune the installation.

```
$ helm show values /iwddm- > values.yaml
```

2. Set up essential IWDDM Helm values:
   - image.registry
   - image.imagePullSecrets (if needed)
   - image.repository
   - image.tag
   - image.repository
   - iwddm.tenantId
   - iwddm.db.*
   - iwddm.db.secret.*
   - iwddm.volumes
   - iwddm.volumeMounts
   - iwddm.env.gim.enabled: true (given that GIM DB secret is provided)
     Use the sample override file:

```
image:
  registry: "pureengageuse1-docker-multicloud.jfrog.io"
  repository: "iwddm/iwd_dm_cloud"
  tag: ""
  pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: pullsecret
iwddm:
  tenantId:  #sample 100
  db:
    createConfigmap: true
    host:
    port: 5432
    dbname:
    user:
    secret:
      enabled: true
      secretName:
      password:
  cronjob:
    schedule: "*/3 * * * *"
    suspend: false
  securityContext: {}
  env:
    executionChain: "full"
    restUrl: "http://iwd.${IWD_NAMESPACE}.svc.${DNS_SCOPE}:4024/iwd/v3"
    monitoring:
      enabled: false
      pushgateway_url: ""
  gim:
    enabled: true
  wfm:
    enabled: false
    rest_url:   "http://wfm-t101-backend.${WFM_NS}.svc.${DNS_SCOPE}:7010/wfm/api/v3"
    health_url: "http://wfm-
t101-backend.${WFM_NS}.svc.${DNS_SCOPE}:7010/?Handler=DISCO"
  gauth:
    enabled: false
    rest_url: http://gauth-auth.${GAUTH_NAMESPACE}.svc.${DNS_SCOPE}:80
    client_id: iwddm_client
```

```
    client_secret:
      valueFrom:
        secretKeyRef:
          name: shared-gauth-iwddm-client-secret
          key: gauth-iwddm-client-secret
  volumes: |-
    - name: iwddm-db-secrets
      secret:
        secretName:
    - name: iwd-secrets
      secret:
        secretName:
  volumeMounts:
    iwddm-db-secrets:
      readOnly: true
      mountPath: "/mnt/env-secrets/db-secrets"
    iwd-secrets:
      readOnly: true
      mountPath: "/mnt/env-secrets/iwd-secrets"
```

3. Install IWD Data Mart using the following command:

```
helm upgrade --install iwddm-{short_tenant_id} /iwddm-cronjob --version={version} -f
 ./values.private.yml
```

Validate the deployment

Watch the helm output at the end of installation. Pods must be in a Running state and they must pass all READY checks.

See the following sample output:

```
Release "iwddm" has been upgraded. Happy Helming!
NAME: iwdem
LAST DEPLOYED: Tue Jul 18 10:18:07 2021
NAMESPACE: iwddm
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Please be patient while iwddm 100.0.0741322 is being deployed
```

Note that IWDDM is a short-living job. So, pods will be created or deleted based on schedule.

Upgrade, roll back, or uninstall

Contents

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

Related documentation:

-
-
-
-

RSS:

- For private edition

> **Important**
>
> The instructions on this page assume you have deployed the services in service-specific namespaces. If you are using a single namespace for all private edition services, replace the namespace element in the commands on this page with the name of your single namespace or project.

Supported upgrade strategies

IWD Data Mart supports the following upgrade strategies:

Service	Upgrade Strategy	Notes
IWD Data Mart	Rolling Update	

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

Timing

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

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

Scheduling considerations

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

Monitoring

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

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

Preparatory steps

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

Each time you upgrade a service:

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

Rolling Update

Rolling Update: Upgrade

Execute the following command to upgrade :

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

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

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

For example,

`helm upgrade --install -f .`

Rolling Update: Verify the upgrade

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

Rolling Update: Rollback

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

```
helm rollback
```

or, to roll back to an even earlier version:

```
helm rollback
```

Alternatively, you can re-install the previous package:

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

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

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

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

For example,

`helm upgrade --install -f .`

Rolling Update: Verify the rollback

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

Uninstall

> **Warning**
> Uninstalling a service removes all Kubernetes resources associated with that service.

> Genesys recommends that you contact Genesys Customer Care before uninstalling any private edition services, particularly in a production environment, to ensure that you understand the implications and to prevent unintended consequences arising from, say, unrecognized dependencies or purged data.

Execute the following command to uninstall :

```
helm uninstall  -n
```

For example,

`helm uninstall` .

> **Important**
> Manually delete all external secrets, if needed.

Observability in IWD Data Mart

Contents

Learn about the logs, metrics, and alerts you should monitor for IWD Data Mart.

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 IWD Data Mart metrics, see:

-

See also System metrics.

Enable monitoring

IWD Data Mart works as a short living cronjob, so it uses Pushgateway for providing metrics. To enable pushing metrics, find the following Helm values as an example:

```
iwddm:
  env:
    monitoring:
      enabled: true
      pushgateway_url: "http://prometheus-pushgateway.monitoring.svc.cluster.local:9091"
```

Service	CRD or annotations?	Port	Endpoint/ Selector	Metrics update interval
	n/a	n/a	n/a	n/a

Configure metrics

The metrics that are exposed by the IWD Data Mart are available by default. No further configuration is required in order to define or expose these metrics.

Alerting

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

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

For descriptions of available IWD Data Mart alerts, see:

-

Configure alerts

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

Logging

IWDDM container logs to *stdout* in structured JSON format.

IWDDM 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
IWDDM	n/a	n/a	n/a	n/a
	Note: As a serverless component, IWDDM is run via a Kubernetes CronJob. By default, the job runs every 15 minutes and pushes information into the Prometheus Pushgateway.			

See details about:

- IWDDM metrics
- IWDDM alerts

Metrics

Labeling is used to distinguish the characteristics of the metrics that is being measured. Along with the metrics the following labels can be used:

Label	Description
tenant	Name of the tenant taken from IWDDM_CCID environment variable
job	Job name. Possible value is: • iwddm_metrics
execution_chain	Indicates execution chain. Possible values are: • intraday • historical • initialize • full
table	Allows to organize metrics by table names
dimension	Allows to organize metrics by dimension names

Metric and description	Metric details	Indicator of
iwddm_job_active Indicates that IWDDM is active Values: • 1 - job is started • 0 - job is stopped of failed	**Unit:** **Type:** gauge **Label:** tenant, job, execution_chain **Sample value:** iwddm_job_active{execution_chain="TEST", job="iwddm_metrics", tenant="TEST"} 1	Error
iwddm_job_last_start Indicates when the IWDDM job started Value: Unix timestamp when job started	**Unit:** milliseconds **Type:** gauge **Label:** tenant, job, execution_chain **Sample value:** iwddm_job_last_start{execution_chain="TEST", job="iwddm_metrics", tenant="TEST"} 1618322383	Error
iwddm_job_last_success Indicates when the IWDDM job succeeded Value: Unix timestamp when job succeeded	**Unit:** milliseconds **Type:** gauge **Label:** tenant, job, execution_chain **Sample value:** iwddm_job_last_success{execution_chain="TEST", job="iwddm_metrics", tenant="TEST"} 1618322383	Error
iwddm_job_last_fail Indicates when the IWDDM job failed Value: Unix timestamp when job failed	**Unit:** milliseconds **Type:** gauge **Label:** tenant, job, execution_chain **Sample value:** iwddm_job_last_fail{execution_chain="TEST", job="iwddm_metrics", tenant="TEST"} 1618322383	Error

Alerts

No alerts are defined for IWDDM.