



Installation and Administration Guide

Privitar Data Security Platform, version 2.1.0

Publication date December 13, 2023

Privitar Data Security Platform, version 2.1.0

© Copyright Informatica LLC 2016, 2023

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, Informatica Cloud, Informatica Intelligent Cloud Services, PowerCenter, PowerExchange, and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMatica PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Table of Contents

1. Welcome	5
1.1. Who Installs and Administers the Privitar Data Security Platform?	5
1.1.1. System Administrators	5
1.1.2. Enterprise Administrators	5
1.1.3. Exchange Administrators	6
1.2. High-Level Platform Installation Workflow	6
1.3. High-Level Platform Administration Workflow	6
1.3.1. How to Access the Platform	6
1.3.2. Responsibilities of an Administrator	7
2. Installation and Upgrade	9
2.1. Installation Prerequisites	9
2.1.1. Installation Host Requirements	9
2.1.2. Set Up Your Deployment Environment	9
2.1.3. Control Plane Installation Requirements	9
2.1.4. Data Plane Installation Requirements	10
2.2. Install the Platform	11
2.2.1. Download the Software Installation Bundle	11
2.2.2. Install the Control Plane	11
2.2.3. Install the Data Plane	15
2.2.4. Deploy HashiCorp Vault as KMS or Secrets Manager	20
2.2.5. Enable Auditing in the Installation Configuration	26
2.3. Upgrade the Platform	33
2.3.1. Back Up Fluent Bit Configurations	33
2.3.2. Upgrade the Control Plane	34
2.3.3. Upgrade Data Planes	36
2.3.4. Restore the Configuration Map for the SIEM Tool	38
2.3.5. Restart the Fluent Bit Pods	38
3. Additional Deployment Considerations	39
3.1. Deploy a Persistent Volume for JDBC Drivers	39
3.2. Configure Single Sign-On	40
3.2.1. Set Up OIDC SSO	40
3.2.2. Set Up SAML SSO	53
3.3. Customize the User Interface	65
3.3.1. User Interface Theme Schema	65
4. Architectural Overview	74
4.1. The Enterprise and the Data Exchange	74
4.2. What Is the Control Plane?	75
4.3. What Is a Data Plane?	76
4.4. Cross-Plane Authentication	77
5. Enterprise Administration Tasks	79
5.1. LDAP Configuration	79
5.1.1. LDAP Authentication	80
5.1.2. LDAP User Setting	80
5.1.3. LDAP Group Setting	82
5.1.4. User Attribute Mappings	83
5.1.5. Test LDAP Connection	83
5.1.6. LDAP Save and Enable	84
5.2. Create Your Initial Data Exchange	84

6. Exchange Administration Tasks	86
6.1. Create a PKI Infrastructure	87
6.2. Create and Edit a Data Plane	88
6.3. Deploy the Data Plane Artifacts	92
6.4. Managing Users and Groups	93
6.4.1. Add Users and Assign Roles	93
6.4.2. Add Groups and Assign Roles	95
6.5. Manage Policy Settings	97
6.5.1. Change Default Access Control Policy Behavior	97
6.6. Allow External Configuration of Correlation ID Queries	97
6.6.1. Configure the Data Proxy	98
6.6.2. Configure the Data Proxy Driver	99
7. System Administration Tasks	101
7.1. Backup, Restore, and Business Continuity	101
7.1.1. Stateless and Stateful Components	101
7.1.2. Data Proxy – Control Plane Communication Behavior	104
7.1.3. Backup and Restore	104
7.1.4. Disaster Recovery	106
7.2. System Logs and Auditing	106
7.2.1. Log Queries and Examples	107
7.2.2. Audit Events	119
7.2.3. Privitar Query Engine and Data Proxy Metrics	127
7.3. Monitoring Metrics	131
7.3.1. Monitoring the Privitar Data Security Platform	132
7.3.2. Viewing Data Plane Metrics	140
7.4. Watermarking	145
7.4.1. About Watermark Investigation	145
7.4.2. Configure Watermark Investigation	146
7.4.3. Investigate a Watermark	153
8. Integrations	154
8.1. Integration of Denodo Data Virtualization	154
8.1.1. Configure the Proxy Driver Connection in Denodo	154
8.1.2. Set Up a Denodo Base View and Run a Query	155
8.1.3. Query Denodo from a Business Intelligence Tool	157
8.1.4. Data Sources Supported with Denodo	158
8.2. Privitar Query Engine Properties	158
8.2.1. Privitar Query Engine Example Implementation	158
8.2.2. Privitar Query Engine JDBC Configuration Information	159
8.2.3. Privitar Query Engine JDBC Connection Properties	159
9. Third-Party Licensing Information	171
10. Administration Troubleshooting	172
10.1. Privitar Support	172
10.2. Reconnect a Node	172
11. Glossary of Data Security Terminology	175

1. Welcome

Welcome to the installation and administration guide for the Privitar Data Security Platform. This document is for enterprise administrators and exchange administrators, though other readers may use it to get an overall feel for the platform architecture.

Privitar's best practice is to have our Professional Services team assist in all installations to ensure the best fit for your local environment.

This guide includes:

- an overview of essential installation tools and components
- a high-level description of the installation process for system administrators and enterprise administrators
- a general overview of how to build the platform in an organization
- an explanation of how the platform authenticates access to its different components
- enterprise administration tasks, including:
 - user/group settings and LDAP
 - creating a data exchange
- exchange administration tasks, including:
 - creation and deployment of a data plane
 - assigning roles to existing users and groups

1.1. Who Installs and Administers the Privitar Data Security Platform?

The following user types are involved in the installation and administration of the Privitar Data Security Platform:

- [System administrators](#)
- [Enterprise administrators](#)
- [Exchange administrators](#)

1.1.1. System Administrators

System administrators are users who perform activities to install and set up the Privitar Data Security Platform. Most of these activities are external to the platform, such as deploying the platform, managing secrets required for installation, performing backup and restore activities, and performing updates to the platform.

1.1.2. Enterprise Administrators

Enterprise administrators are users who perform operations within the Privitar Data Security Platform, such as creating a data exchange, creating a data plane, and configuring a data plane.

1.1.3. Exchange Administrators

Exchange administrators are users who perform tasks within a data exchange, such as creating and editing a data plane, managing users and groups, and performing everyday administration tasks.

1.2. High-Level Platform Installation Workflow

With the assistance of our Professional Services team, administrators in your organization will perform the following tasks to install the platform:

1. A system administrator confirms that all [prerequisites](#) are in place.
2. A system administrator installs and configures the [control plane](#).
3. A system administrator installs and configures a key management system (*Hashicorp Vault KMS* is supported out-of-the-box).
4. A system administrator installs and configures the [data plane](#).
5. Administrators at your organization can now [administer the platform](#).

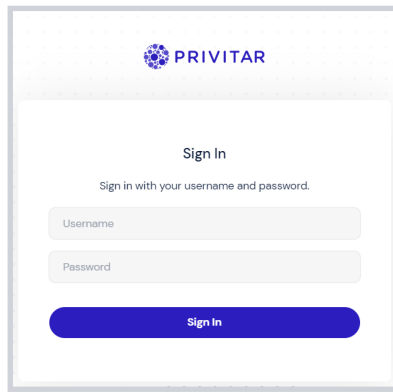
1.3. High-Level Platform Administration Workflow

Administrators in your organization perform the following tasks to enable others to provision data through the Privitar Data Security Platform:

1. System administrators install the Privitar Data Security Platform. This includes deployment of the data agent and data proxy.
2. An enterprise administrator:
 - a. [Configures user management through LDAP](#) or an internal registry
 - b. [Creates one or more data exchanges](#)
3. Exchange administrators:
 - a. [Create a data plane](#)
 - b. [Configure the data plane \(data agent and data proxy\)](#)
 - c. Deploy the data agent and data proxy with a system administrator
 - d. [Add users and user groups](#)
 - e. Assign roles to [users](#) and user [groups](#)
4. Exchange administrators perform ongoing administration:
 - a. Edit/manage data planes
 - b. Assign roles to existing users and groups

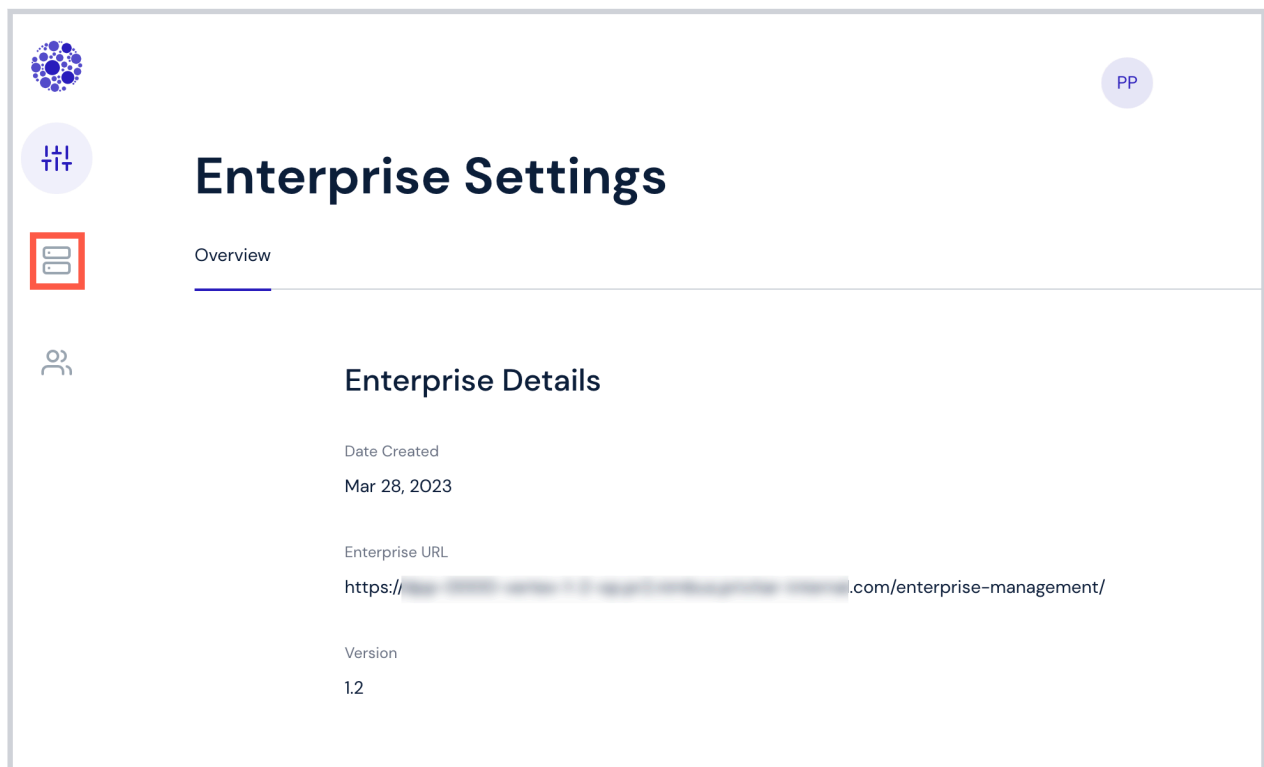
1.3.1. How to Access the Platform

During installation, the installer script sets a username and password. These are the login credentials of the initial bootstrap user, effectively the first enterprise administrator.



After installation, use the login credentials and the supplied URL to log in as an enterprise administrator on the Privitar Data Security Platform (DSP) login page.

Once you sign in, you can create other users and invite them to log in to the platform. For a SaaS installation, the platform sends an email to the new user.



The **Enterprise Settings** page is an overview of your enterprise. In the left navigation, there are **Data Exchanges** (highlighted) and **User Management** menu options.

1.3.2. Responsibilities of an Administrator

Enterprise administrators create the exchanges that are subsequently administered by the exchange administrators. Most enterprise administration takes place during initial start-up and configuration.

Enterprise administrators will manage, or have access to, a Lightweight Directory Access Protocol (LDAP) registry. You can set the platform to use an internal registry but, in most cases, you will want to make use of your existing user and group accounts in LDAP. The

enterprise administrator creates the links to LDAP and configures the underlying user and group management.

Exchange administrators manage exchanges within the enterprise. They are responsible for the bulk of ongoing maintenance administration. After initial setup and enterprise-level configuration, the exchange administrator:

- creates [data planes](#) (data agent and data proxy)
- manages data planes
- creates users and assigns user roles
- creates groups and assigns group roles

System administrators deploy the data agent and data proxy.

2. Installation and Upgrade

2.1. Installation Prerequisites

2.1.1. Installation Host Requirements

The host machine used to run the Privitar Data Security Platform installation with both the data plane and control plane clusters must have:

- A [Linux](#) host
- Internet access (optionally required for pulling the latest version of the Istio service mesh as part of install process); please see [Control Plane Installation Requirements](#)
- [Docker](#)
- [Helm](#) 3.8+
- [kubectl](#) 1.20+ ([Kubernetes](#) command line tool)
- [jq](#) (JSON query tool)
- [OpenSSL](#)
- [Python](#)
- [kubeconfig configuration files](#) for both control plane and data plane cluster
- Connectivity to both control plane and data plane cluster using the [kubectl](#) [Kubernetes](#) command line tool

2.1.2. Set Up Your Deployment Environment

Ensure you have installed the following on the machine you are using for the deployment:

- Helm
- Docker Desktop

To set up your deployment environment:

- Load the [Installation Bundle](#) tar files on a machine with the pre-requisites.
- Unzip both the data plane and control plane TGZ files into a workspace directory:

```
1 tar xvzf dpp-data-plane-[version].tar.gz
2 tar xvzf dpp-control-plane-[version].tar.gz
```

2.1.3. Control Plane Installation Requirements

To deploy a working control plane you need:

- A [Kubernetes](#) cluster on [Kubernetes](#) version 1.20+ with a minimum of three nodes with eight cores and 30 GB of memory, based on an equivalent public cloud store-keeping unit (SKU)
- Internet access from the [Kubernetes](#) cluster (optional; the platform can be installed in "air-gapped" [Kubernetes](#) clusters)

- A container registry to store images (reachable from the Kubernetes cluster)
- A [StorageClass resource](#) to support the [PersistentVolumes subsystem](#)
- Service mesh configured in the Kubernetes cluster; Privitar recommends [Istio service mesh](#) version 1.10+; the `dppctl` application can be configured to install Istio automatically (if your Kubernetes cluster has Internet access); can also be used with a third-party service mesh of your choice
- A cluster load balancer.

**Note**

When deploying in the three public cloud providers (AWS, GCP, and Azure), the cluster load balancer is typically created automatically as part of the configuration of the cluster service mesh.

- A DNS record to route traffic from the chosen Control Plane fully qualified domain name (FQDN) to the load balancer public IP address
- Network access to external services, such as LDAP, if required

The platform supports optional audit exports and metrics (a cluster with [Prometheus](#) pre-installed). Refer to the [Configure the control-plane-values File](#) for details on how to configure these options.

2.1.4. Data Plane Installation Requirements

To deploy a working data plane using a self-generated certificate based on Istio service mesh, you require:

- A Kubernetes cluster on [Kubernetes](#) version 1.20+ with a minimum of three nodes with eight cores and 30 GB of memory, based on an equivalent public cloud store-keeping unit (SKU)
- Internet access from the Kubernetes cluster (optional; the platform can be installed in "air-gapped" Kubernetes clusters)
- A container registry to store images (reachable from the Kubernetes cluster)
- A [StorageClass resource](#) to support the [PersistentVolumes subsystem](#)
- Persistent Volume containing the JDBC drivers the Data Proxy will be used to connect to source data systems (refer to [Deploy a Persistent Volume for JDBC Drivers](#) for more detail)
- Service mesh; Privitar recommends [Istio service mesh](#) version 1.10+; the `dppctl` application can be configured to install Istio automatically (if your Kubernetes cluster has Internet access); the platform can also be used with a third-party service mesh of your choice
- A cluster load balancer.

**Note**

When deploying in the three public cloud providers (AWS, GCP, and Azure), the cluster load balancer is typically created automatically as part of the configuration of the cluster service mesh.

- A DNS record to route traffic from the chosen Control Plane fully qualified domain name (FQDN) to the load balancer public IP address
- Network access to the source database(s) from the Kubernetes cluster
- A deployed secrets management system. See [Deploy HashiCorp Vault as KMS or Secrets Manager](#).
- A third-party event monitoring tool to collect metrics.

If you have installed [Prometheus](#) within the same namespace in the data plane, it can autoscan for the available metrics from the data plane.

**Note**

Ensure that your environment has the correct permissions to allow autoscan to occur.

If you have installed Prometheus outside of the namespace where you deployed the data plane, you need to manually expose the metrics endpoint 9095 for the pods from which you are intending to collect metrics.

2.2. Install the Platform

These instructions are the steps required to deploy a Privitar Data Security Platform environment from an installation host machine.

2.2.1. Download the Software Installation Bundle

The software installation bundle consists of the following files:

- `dpp-data-plane-[nnnn].tar.gz`; the release bundle for the DSP Platform data plane
- `dpp-control-plane-[nnnn].tar.gz`; the release bundle for the DSP Platform control plane
- `licences-[nnnn]/privitar-licences.yaml`; the release licenses file

**Note**

The placeholder `nnnn` represents a series of digits.

To download the software installation bundle:

1. Ensure you have met all the [Installation Host Requirements](#).
2. Download the software installation bundle to your host machine.
3. Log in to your preferred container registry.
4. Update your `kubectl` config file to manage the destination cluster.

2.2.2. Install the Control Plane

To install the control plane, follow the steps in these sections:

1. [Run the Control Plane Installer](#)
2. [Configure the control-plane-values File](#)

**Important**

You must [install Bitnami Prometheus](#) before you [install the control plane](#) in order to enable the built-in metrics.

Run the Control Plane Installer

To install the control plane:

1. Ensure you have completed all the [Control Plane Installation Requirements](#).
2. From within the `dpp-control-plane-[nnnn]` folder, [configure the control-plane-values.yaml file](#).

**Note**

The placeholder `nnnn` represents a series of digits.

3. Deploy the control plane using the following command:

```
./dppctl install -f ./control-plane-values.yaml -u privitar -p password
```

**Note**

If your LDAP server requires an SSL certificate, add a `-l [myldapcert].cert` parameter to the command. You obtain the SSL certificate from the LDAP server.

4. Create a type "A" DNS address record ("A" record) within your DNS for the hostname you configured in the `control-plane-values.yaml` file.
5. Point the "A" record target to your load balancer IP address.

Configure the control-plane-values File

The following parameters are available in the `control-plane-values` file to configure for your environment.

**Note**

You can optionally configure auditing. Ensure you have met the [Control Plane Installation Requirements](#) if you choose to do so.

Table 1. Configure the Control Plane Namespace and Hostname

Parameter	Description	Options / Example
namespace (required)	The Kubernetes namespace into which the platform will be installed.	"control-plane"
clean	Specifies whether or not the existing namespace and its resources should be cleaned out.	true / false
externalHostname (required)	The hostname at which the platform will be accessible, external to the cluster.	"DSPprod.com"

Table 2. Configure Istio Service Mesh

Parameter	Description	Options / Example
istio.install	Specifies whether or not istio should be installed.	true / false
istio.enabled	Specifies whether or not istio is enabled.	true / false
istio.namespace Label	Specifies the namespace label used to inject istio sidecars.	"istio-injection=enabled"

Table 3. Configure Transport Layer Security (TLS)

Parameter	Description	Options / Example
tls.autoGenerate	Specifies whether or not to generate self-signed certificate for the externalHostname.	true / false
tls.autoGeneratedSecret	The name of the secret where the auto generated TLS certificate is stored.	"dpp-control-plane-tls-ingress-cert"
tls.existingCertificateSecret	The name of the secret where the TLS certificate is stored.	
tls.keyFile	The private key file to use for external TLS.	
tls.certificateFile	The public key file to use for external TLS.	
tls.certificateNamespace	The namespace into which to install the public certificate.	"istio-system"

Table 4. Configure Container Images

Parameter	Description	Options / Example
images.load	Specifies whether or not images should be loaded into the local registry.	true / false
images.push	Specifies whether or not images should be pushed to the target registry.	true / false
images.targetRegistry	URL of the registry to which images should be pushed and from which cluster should pull.	
images.imagepullSecret	The name of a kubernetes secret which will keep the secret for the registry.	

Table 5. Configure Admin User, Global Resources, Persistent Storage Class Name, and the Scale

Parameter	Description	Options / Example
adminUsername	The name of the first user who will be used to bootstrap the installation: if omitted (recommended), the installer prompts for it.	"testuser@acme.com" Ensure that you use a valid email address. Keycloak requires that the administrator username is an email address.
adminPassword	The password of the first user who will be used to bootstrap the installation: if omitted (recommended), the installer prompts for it.	"somepassword"
installGlobal	Specifies whether or not global resources (priority classes) should be created.	true / false
storageClass	Specifies the storage class name to use for all persistent storage.	"standard-rwo"
scaleConfig (required)	Changes the capacity of services by adjusting the memory and CPU resource settings and the number of pods.	small / medium / large

Table 6. Configure Audit

Parameter	Description	Options / Example
audit.installFluentbit	Specifies whether or not Fluent Bit should be installed.	true / false
audit.enabled	Specifies whether or not audit is enabled.	true / false
audit.auditServiceUrl	The hostname at which the audit service is accessible.	"http://fluent-bit:8888"

Table 7. Configure Whatfix

Parameter	Description	Options / Example
whatfix.url	The URL that points to the location of the WhatFix content and script: by default, it points to the WhatFix CDN script but may be adjusted to point to a locally installed server.	Default: https://cdn.whatfix.com/prod/8324b840-482e-11ec-83be-3e2a4292da47/initiator/initiator.nocache.js

Table 8. Configure the User Interface Theme

Parameter	Description	Options / Example
uiTheme	<p>A JSON object that controls the look of the platform's user interface (UI), including colors and logos.</p> <p>Must comply with the JSON schema (<code>theme-schema.json</code>).</p>	<p>Default: <code>{ }</code></p> <p>For details and examples, see Customize the User Interface.</p>

2.2.3. Install the Data Plane

To install the data plane:

1. [Run the Data Plane Installer](#)
2. [Configure the data-plane-values.yaml File](#)

Run the Data Plane Installer

To install the data plane:

1. From within the `dpp-data-plane-[nnnn]` folder, create the public keys required for the data agent and data proxy agent using the following command:



Note

The placeholder `nnnn` represents a series of digits.

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out ./rsa-private.pem
openssl rsa -pubout -in ./rsa-private.pem -out ./rsa-public.pem
```

These keys are required for `data-plane-values.yaml` file.

2. Access the platform's user interface (UI), and create a new data plane. Do this by logging into the relevant data exchange.
 - a. Click your profile icon in the upper right-hand corner of the screen, and select **View Exchange**.
 - b. Select the **Data Planes** tab, and click the **+** button to create a new data plane.
3. Follow the wizard to create your new data plane, providing the two public keys created in step [???TITLE???](#).
4. When prompted, enter the domain name you intend to use for your new data plane. Set the port to the default deployment port, 443.
5. On completion of the wizard, you are prompted to download two configuration files. Open them in a text editor and note the values for `dataPlaneId` and `exchangeId`.

These IDs are required for the `data-plane-values.yaml` file.

6. From within the `dpp-data-plane-[nnnn]` folder, retrieve the server certificate used by the platform and store it as a file. Retrieve the server certificate using the following command. Please note; if you are using the default service mesh (istio) that

is optionally deployed by Privitar, the namespace of your service mesh is typically `istio-system`.



Note

The placeholder `nnnn` represents a series of digits.

```
kubectl get secret dpp-control-plane-tls-ingress-cert -n [namespace of
your service mesh] -o jsonpath='{.data.tls\.cert}' | base64 --decode > ./
controlplane.crt
```

This path to this certificate file is required for the `data-plane-values.yaml` file.

7. Configure the `data-plane-values.yaml` file. Refer to [Configure the data-plane-values.yaml File](#) for more detail.
 - a. Provide the hostnames, data agent key, data proxy agent key, and certificate file path, created in the previous steps as well as the Docker registry URL.
 - b. Set both `load` and `push` in the `images` section to **true**.
 - c. Make sure the `dataPlaneConfigClaimName` parameter is set to the name of the Persistent Volume Claim (PVC) for the Persistent Volume holding your JDBC drivers.
 - d. Ensure your `clean` parameter is set to **false** to allow you to create the PersistentVolume (pv) and PersistentVolumeClaim (pvc) manually before running the script.
8. Create a type "A" DNS address record ("A" record) within your DNS for the hostname you configured in the `data-plane-values.yaml` file.
9. Point the "A" record target to your load balancer IP address.
10. Run the following command to deploy the data plane components:

```
./dppctl install -f ./data-plane-values.yaml -u privitar -p password
```

Configure the data-plane-values.yaml File

The following parameters are available in the `data-plane-values.yaml` file to configure for your environment.

Table 9. Configure the Data Plane Namespace and Hostname

Parameter	Description	Options / Example
<code>namespace</code>	The Kubernetes namespace into which the data plane will be installed.	"data-plane"
<code>clean</code>	Specifies whether or not the existing namespace and its resources should be cleaned out.	true / false
<code>externalHostnam e (required)</code>	The hostname at which the data plane will be accessible, external to the cluster.	"dataplane.privitar.in ternal"

Table 10. Configure the Control Plane

Parameter	Description	Options / Example
<code>controlPlane.hostname</code> (required)	The hostname at which the control plane is deployed.	"controlplane.privatar.internal"
<code>controlPlane.hostIp</code>	The externally accessible IP of the control plane; if left blank, it will be looked up normally, using DNS.	"20.90.124.31"
<code>controlPlane.dataPlaneId</code> (required)	The ID of a data agent that has been created on the control plane.	"55f8de39-6440-47f1-8a33-8b066512f571"
<code>controlPlane.exchangeId</code> (required)	The ID of an exchange that has been created on the control plane.	"e5252cce-b41e-4781-b856-d508183f9686"
<code>controlPlane.certificateFile</code> (required)	The path to a file containing the certificate used by the control plane.	"/path/to/controlplane.crt"

Table 11. Configure the Data Proxy

Parameter	Description	Options / Example
<code>dataProxy.enableContextHeaderAuth</code>	Specifies whether or not to enable context header auth on the data proxy to allow the driver to log in using a JWT. This will bypass some security in the data proxy and <i>must</i> be set to false, unless there is an authoriser protecting the data proxy that will deny requests that don't have valid authentication.	true / false
<code>dataProxy.bypassPolicyAllowList</code>	A comma-separated list of numeric user IDs generated by the platform that are allowed to bypass a policy. Service account (non-human) users often have this permission to allow them to query the schema and metadata of the database.	
<code>dataProxy.bypassPolicyMode</code>	The bypass policy mode to be used.	"RESTRICTIVE"

Table 12. Configure Istio Service Mesh

Parameter	Description	Options / Example
<code>istio.install</code>	Specifies whether or not Istio should be installed.	true / false
<code>istio.enabled</code>	Specifies whether or not Istio is enabled.	true / false
<code>istio.namespaceLabel</code>	Specifies the namespace label used to inject Istio sidecars.	"istio-injection=enabled"

Table 13. Configure Transport Layer Security (TLS)

Parameter	Description	Options / Example
<code>tls.autoGenerate</code>	Specifies whether or not to generate a self-signed certificate for the <code>externalHostname</code> .	<code>true</code> / <code>false</code>
<code>tls.autoGeneratedSecret</code>	The name of the Kubernetes secret where the auto-generated TLS certificate is stored.	"dpp-data-plane-tls-ingress-cert"
<code>tls.existingCertificateSecret</code>	The name of the Kubernetes secret where the TLS certificate is stored.	
<code>tls.keyFile</code>	The private key file to use for external TLS.	
<code>tls.certificateFile</code>	The public key file to use for external TLS.	
<code>tls.certificateNamespace</code>	The namespace into which to install the public certificate.	

Table 14. Configure Container Images

Parameter	Description	Options / Example
<code>images.load</code>	Specifies whether or not images should be loaded into the local registry.	<code>true</code> / <code>false</code>
<code>images.push</code>	Specifies whether or not images should be pushed to the target registry.	<code>true</code> / <code>false</code>
<code>images.targetRegistry</code>	URL of the registry to which images should be pushed and from which cluster should pull.	
<code>images.imagePullSecret</code>	The name of a Kubernetes secret which will keep the secret for the registry.	

Table 15. Configure Global Resources, Persistent Storage Class Name, and the Scale

Parameter	Description	Options / Example
<code>installGlobal</code>	Specifies whether or not global resources (priority classes) should be created.	<code>true</code> / <code>false</code>
<code>dataPlaneConfigClaimName</code>	Specifies the name of the <code>PersistentVolumeClaim</code> (PVC) containing the config files used by the data plane pods.	"data-plane-pvc"

Table 16. Configure Client Authorization

Parameter	Description	Options / Example
<code>clientauth.dataAgentPrivateKeyFile</code> (required)	A path to a file containing a private key for the data agent to use to authenticate with the control plane.	<code>../key/rsa-private.pem</code>
<code>clientauth.dynamicProxyPrivateKeyFile</code> (required)	A path to a file containing a private key for the dynamic proxy to use to authenticate with the control plane.	<code>../key/rsa-private2.pem</code>

Parameter	Description	Options / Example
<code>clientauth.scaleConfig</code> (required)	Changes the capacity of services by adjusting the memory and CPU resource settings, and the number of pods.	small / medium / large

Table 17. Configure Audit

Parameter	Description	Options / Example
<code>audit.installFluentbit</code>	Specifies whether or not Fluent Bit should be installed.	true / false
<code>audit.enabled</code>	Specifies whether or not audit is enabled.	true / false
<code>audit.auditServiceUrl</code>	The hostname at which the audit service is accessible.	"http://fluent-bit:8888"

Table 18. Configure the Key Management System (HashiCorp Vault for Consistent Tokenization)

Parameter	Description	Options / Example
<code>kms.vaultlessKeyIdentifier</code>	Identifier for the key in the Key Management Service (KMS) to use for vaultless tokenization.	0 privitar-vaultless-key
<code>kms.watermarkingKeyIdentifier</code>	An identifier for the key in the KMS to use for watermarking.	0 privitar-watermarking-key
<code>kms.extraProperties</code>	Containing authentication properties provided as Base64-encoded key-value pairs. This can be used as an alternative to creating the secret and setting <code>extraPropertiesSecretName</code> .	authenticationToken=root
<code>kms.extraPropertiesSecretName</code>	The name of the secret that containing authentication properties provided as encoded key-value pairs.	
<code>kms.properties</code>	Identifies the specific properties for connecting to KMS: URL and path.	<pre> properties: url: "http://10.20.30.40:8200" path: "secret/" authenticationMethod: "KUBERNETES" kubernetesAuthPath: "kubernetes" kubernetesRole : "KuthAuthRoleForDSP" </pre>

**Important**

There are strict length requirements for these keys. You can use a command like the following example to generate keys of the appropriate length:

```
openssl rand -base64 32 | tr -d '\n'
```

This example generates a 32-bit key.

Table 19. Configure the Secrets Manager

Parameter	Description	Options / Example
<code>secretsManager.enabled</code>	Specifies whether or not to use the secrets manager for resolving connection username and password.	true / false
<code>secretsManager.properties</code>	Identifies the specific properties for connecting to the secrets manager: URL and path.	
<code>secretsManager.adapterJarPath</code>	The JAR path for the secrets manager: a default should exist in the image. Other images should be stored on the config volume (mounted at /config).	
<code>secretsManager.extraProperties</code>	Extension of secrets manager properties, stored as a <code>.properties</code> file in a secret: key and value.	
<code>secretsManager.extraPropertiesSecretName</code>	An alternative way to provide extra properties (<code>extraProperties</code> should be set to [], if you use this parameter). The secret should contain a <code>secret-manager-extra.properties</code> file.	

Table 20. Configure Transparent Mode

Parameter	Description	Options / Example
<code>transparentMode</code>	Determines whether to enable transparent mode in the Privitar Query Engine.	true / false
:	See Integration of Denodo Data Virtualization for more information.	Only set to true when using Denodo.

2.2.4. Deploy HashiCorp Vault as KMS or Secrets Manager

To enable consistent tokenization on the DSP Platform, install and configure a key management system. The platform has "out-of-the-box" support for [HashiCorp® Vault Key Management System](#) (HashiCorp® Vault KMS). You can also use Kubernetes Auth in conjunction with HashiCorp Vault to manage keys for consistent tokenization.

**Note**

Despite the name, HashiCorp Vault is a KMS, not a token vault.

Optionally, you can also use the same instance or a different instance of HashiCorp Vault as a secrets manager to obscure database login credentials. You can also use Kubernetes Auth in conjunction with HashiCorp Vault to manage these secrets.

**Note**

If you choose not to use a secrets manager, you will need to type the database login credentials (username and password) in the the platform's user interface (UI) in cleartext.

Deploy HashiCorp Vault on a machine instance in a network location accessible from your Kubernetes cluster. For installation instructions, refer to <https://developer.hashicorp.com/vault/tutorials/getting-started/getting-started-install>.

After installation, you must configure an account for Privitar to access HashiCorp Vault using token authentication. Take note of this account's access token. For more detail on HashiCorp Vault token authentication, refer to the HashiCorp documentation: <https://developer.hashicorp.com/vault/docs/internals/token>.

Configure HashiCorp Vault KMS and Create Secret Material

To configure HashiCorp Vault KMS and install secret material for use by the platform, make sure your HashiCorp Vault server is running and unsealed.

The platform requires two primary secrets:

- a "vaultless" secret used in NOVLT tokenization
- a "watermarking" secret used to insert watermarks in protected data

Use the following instructions to enable HashiCorp Vault's KMS features and install secret material:

Repeat step 2 to create a vaultless secret and a watermarking secret.

1. Enable your versioned secrets manager as follows:

In the following example, the `[secret path]` is an internal HashiCorp vault "folder" to hold relevant platform secrets.

```
vault secrets enable -path=[secret path] kv-v2
vault kv enable-versioning [secret path]/
```

2. Install a secret, similar to the following example:

In the example below, `[secret name]` is a label used to identify the secret to the platform.

```
vault kv put [secret path]/[secret name] value=[secret material]
```

**Note**

When using HashiCorp Vault Enterprise, include the vault namespace in the properties used by the data plane to establish connectivity with the HashiCorp vault. So, if the namespace is `ns1`, the path will be `ns1/secret path`.

Configure the Data Plane Deployment to Use HashiCorp Vault as a KMS Using Token Auth

In your `data-plane-values.yaml` file you must specify KMS attributes to ensure that the platform can connect to, and use, HashiCorp Vault.

To edit the properties in the `kms` block:

1. Enter your HashiCorp Vault connection details:
 - `kms.properties.url`—Enter the hostname URL and port.
 - `kms.properties.path`—Enter the secret path you specified when deploying HashiCorp.
 - `authenticationMethod`—Enter "TOKEN".

**Important**

System tokens expire after 32 days. Be sure to set a token renewal process within 32 days. Privitar does not advise using root tokens in production environments. To learn more, see <https://developer.hashicorp.com/vault/docs/concepts/tokens#root-tokens>.

2. `extraProperties`:

The platform accepts the access token as a base64-encoded property in this parameter.

Base64 encode your authentication token using the following command:

```
echo -n "authenticationToken=[HashiCorp Vault Authentication Token]" |  
base64 -
```

Copy the resultant string into the `extraProperties` parameter.

3. Identify the secret that you'd like to use for platform tokenization operations by setting the parameters `vaultlessKeyIdentifier` and `watermarkingKeyIdentifier`.
4. Set these parameters to the versioned name of the vaultless secret and watermarking secret respectively, configured in previous steps. The versioned name should take the form of `[version number]|[secret name]`. Newly-created secrets will have a version of 0; a valid value for `vaultlessKeyIdentifier` could therefore be `0|privitar-vaultless-key`.

5. To ensure that the values you are specifying are correct, query HashiCorp Vault to confirm key material information. From the machine running HashiCorp Vault server, run the following example command:

```
$ vault kv get secret/privitar-vaultless-key
===== Secret Path =====
secret/data/privitar-vaultless-key

===== Metadata =====
Key          Value
---          -
created_time  2022-10-04T13:42:12.527717585Z
custom_metadata  <nil>
deletion_time  n/a
destroyed     false
version       0

==== Data ====
Key          Value
---          -
value        xxxx
```

To learn more, see <https://developer.hashicorp.com/vault/docs/auth/token>.

Configure the Data Plane Deployment to Use HashiCorp Vault as a KMS Using Kubernetes Auth

In your `data-plane-values.yaml` file you must specify KMS attributes to ensure that the platform can connect to, and use, HashiCorp Vault.



Note

Be sure to follow all instructions for configuring HashiCorp Vault to use Kubernetes Auth, as described here: <https://developer.hashicorp.com/vault/docs/auth/kubernetes>.

To edit the properties in the `kms` block:

1. Enter your HashiCorp Vault connection details:
 - `kms.properties.url`—Enter the hostname URL and port.
 - `kms.properties.path`—Enter the secret path you specified when deploying HashiCorp.
 - `authenticationMethod`—Enter "KUBERNETES".
 - `kubernetesRole`—Enter the role used to authenticate with HashiCorp Vault.
2. Identify the secret that you'd like to use for platform tokenization operations by setting the parameters `vaultlessKeyIdentifier` and `watermarkingKeyIdentifier`.
3. Set these parameters to the versioned name of the vaultless secret and watermarking secret respectively, configured in previous steps. The versioned name should take

the form of `[version number]|[secret name]`. Newly-created secrets will have a version of 0; a valid value for `vaultlessKeyIdIdentifier` could therefore be `0|privitar-vaultless-key`.

4. To ensure that the values you are specifying are correct, query HashiCorp Vault to confirm key material information. From the machine running HashiCorp Vault server, run the following example command:

```
$ vault kv get secret/privitar-vaultless-key
===== Secret Path =====
secret/data/privitar-vaultless-key

===== Metadata =====
Key                           Value
---                           -
created_time                  2022-10-04T13:42:12.527717585Z
custom_metadata               <nil>
deletion_time                 n/a
destroyed                     false
version                       0

==== Data ====
Key      Value
---      -
value    xxxx
```

Configure the Data Plane Deployment to Use HashiCorp Vault as a Secrets Manager Using Token Auth

In your `data-plane-values.yaml` file you must specify secrets management attributes to ensure that the platform can connect to, and use, HashiCorp Vault.

To edit the properties in the `secretManager` block:

1. Enter your HashiCorp Vault connection details:
 - `kms.properties.url`—Enter the hostname URL and port.
 - `kms.properties.path`—Enter the secret path you specified when deploying HashiCorp.
 - `authenticationMethod`—Enter "TOKEN".



Important

System tokens expire after 32 days. Be sure to set a token renewal process within 32 days. Privitar does not advise using root tokens in production environments. To learn more, see <https://developer.hashicorp.com/vault/docs/concepts/tokens#root-tokens>.

2. `extraProperties`:

The platform accepts the access token as a base64-encoded property in this parameter.

Base64 encode your authentication token using the following command:


```
echo -n "authenticationToken=[HashiCorp Vault Authentication Token]" |
base64 -
```

Copy the resultant string into the `extraProperties` parameter.

3. To ensure that the values you are specifying are correct, query HashiCorp Vault to confirm key material information. From the machine running HashiCorp Vault server, run the following example command:

```
$ vault kv get secret/postgres-password-username
===== Secret Path =====
secret/data/postgres-password-username

===== Metadata =====
Key                               Value
---                               -
created_time                      2022-10-04T13:42:12.527717585Z
custom_metadata                   <nil>
deletion_time                     n/a
destroyed                        false
version                          0

==== Data ====
Key      Value
---      -
value    xxxx
```

To learn more, see <https://developer.hashicorp.com/vault/docs/secrets/databases>.

Configure the Data Plane Deployment to Use HashiCorp Vault as a Secrets Manager Using Kubernetes Auth

In your `data-plane-values.yaml` file you must specify secrets management attributes to ensure that the platform can connect to, and use, HashiCorp Vault.

To edit the properties in the `secretManager` block:

1. Enter your HashiCorp Vault connection details:
 - `kms.properties.url`—Enter the hostname URL and port.
 - `kms.properties.path`—Enter the secret path you specified when deploying HashiCorp.
 - `authenticationMethod`—Enter "KUBERNETES".
2. To ensure that the values you are specifying are correct, query HashiCorp Vault to confirm key material information. From the machine running HashiCorp Vault server, run the following example command:

```
$ vault kv get secret/postgres-password-username
===== Secret Path =====
secret/data/postgres-password-username

===== Metadata =====
Key                               Value
---                               -
```

```

created_time      2022-10-04T13:42:12.527717585Z
custom_metadata   <nil>
deletion_time     n/a
destroyed        false
version          0

==== Data ====
Key              Value
---             -
value           xxxx

```

To learn more, see <https://developer.hashicorp.com/vault/docs/platform/k8s>.

2.2.5. Enable Auditing in the Installation Configuration

Data guardians use logs of all Privitar Data Security Platform events to ensure that all data is provisioned in compliance with company policies and to audit changes to demonstrate data compliance. System administrators also use them to query logs and check for errors.

The Privitar Data Security Platform uses Fluent Bit to collect audit events for display in the user interface. Additionally, you can forward audit records to your preferred security information and event management (SIEM) solution, for example, Splunk or QRadar.

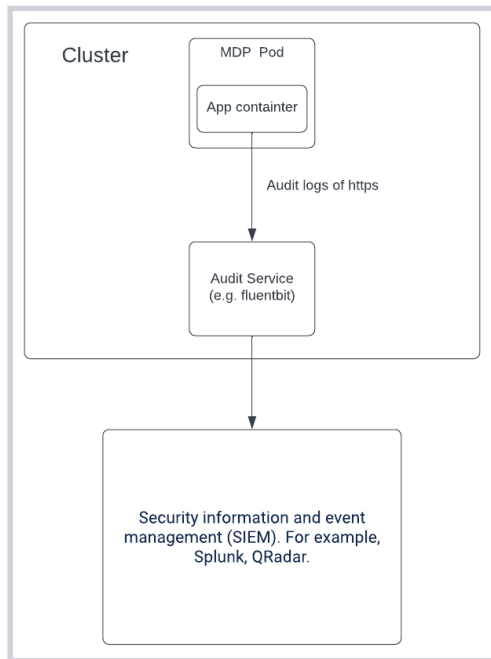


Warning

If you change the `auditService` away from Fluent Bit, you are also explicitly disabling the built-in audit view in DSP.

Audit Records Collector

Audit records can contain sensitive information, so the platform does not stream them to standard output and only stores them within internal, secure repositories for display in the Audit View user interface. In all cases, they are sent over HTTPS to a [Fluent Bit](#) audit service. [Mutual TLS](#) (mTLS) is handled by [Istio](#) if the audit service is deployed in the same namespace or any namespace where Istio is available.



Fluent Bit

Fluent Bit is an open-source log collector that uses input and output plug-ins to:

- collect data from multiple sources
- distribute or send data to various destinations

The on-premises DSP Platform is shipped with Fluent Bit. It is configured by default to use HTTP source and the NULL output plug-in that throws away events. This means you can get up and running easily and forward audit records to your chosen SIEM solution. To do that, you need to [update a configmap](#), created by the installer, with any supported Fluent Bit output plug-in.



Note

You can install any other audit service by disabling the installation of Fluent Bit and providing the installer with any audit service URL.

File System Buffering and Retries

Fluent Bit supports file system buffering to handle slow network or unresponsive target services. With file system buffering, Fluent Bit stores the data in memory but also maps a copy on the disk. Since audit data is sensitive, Fluent Bit uses in-memory buffering by default. You can enable file system buffering, but consider the risk of storing sensitive information on disk.

Fluent Bit has a retry mechanism and provides a simple configuration option called `Retry_Limit`. You can set it on each output to impose a limit to try `n` times and then discard the data after reaching the limit. When you set `Retry_Limit` to `no_limits`, there is no limit to the number of retries.

Default Configuration Settings for Fluent Bit

There are two Fluent Bit services: one for the control plane, and one for each data plane.

These Fluent Bit services are preconfigured upon installation. The preset defaults for both the control plane and data planes are preconfigured with an output plugin that forwards and routes events into the internal Privitar Data Security Platform audit collection. In addition, the control plane Fluent Bit has a default filter configured.



Note

It is expected that the preconfigured settings remain as-is. Comments have been added to the configurations that are visible when the Fluent Bit configuration map is edited to ensure that the sections are clearly distinguished when modifications are made.

Enable Auditing in the Installer Configuration

Both the control plane values file and the data plane values file have the following section:

```
audit:
  ## @param audit.installFluentbit specifies whether or not Fluent Bit
  should be installed
  ##
  installFluentbit: true
  ## @param audit.enabled specifies whether or not audit is enabled
  ##
  enabled: true
  ## @param audit.auditServiceUrl is the hostname at which the audit service
  is accessible
  ##
  auditServiceUrl: http://fluent-bit:8888
```

The parameters to configure are:

- `installFluentbit`—whether to install Fluent Bit as part of the deployment in the target namespace.



Note

If Fluent Bit is not installed, audit logs cannot be routed and displayed within the platform's Audit Log view.

- `enabled`—whether pods should emit audit records.



Note

If the pod is not enabled to emit audit records, no audit events will be routed and available for display in the platform's Audit Log view.

- `auditServiceUrl`—the audit service host. It is configured to the Fluent Bit service by default, but you can be change it to any other audit service with an HTTP endpoint.

**Note**

Only Fluent Bit can be used to route and display audit events within the platform's Audit Log view.

Configure the Control Plane Fluent Bit Instance

This Fluent Bit instance receives logs relating to audit events originating from control plane configuration changes (such as policy creation, project approval, and so on).

It will also receive log events forwarded on from the data plane Fluent Bit instances. This is how the platform collects events internally for display in the Audit Log view.

If you have made changes to the port numbers in use, modify the configuration map accordingly. If you are planning to route to your SIEM tool as well, see [Configure Log Routing to a SIEM Tool](#).

Control Plane Fluent Bit Default Configuration Map

```
[SERVICE]
  Daemon Off
  Flush 1
  Log_Level info
  Parsers_File parsers.conf
  Parsers_File custom_parsers.conf
  HTTP_Server On
  HTTP_Listen 0.0.0.0
  HTTP_Port 2020
  Health_Check On

[INPUT]
  name http
  host 0.0.0.0
  port 8888
  tag incoming

# Typically, if using an external audit collector, both the control plane
# Fluent Bit and data plane Fluent Bit instances would be configured to output
# to it.
#
# This filter defines a new tag `incoming.excludingDataPlaneEvents` which
# will only be present for events that are not
# coming from the data plane. This is needed if you wish to prevent
# duplicate data plane events being emitted both from the data
# plane Fluent Bit and the control plane Fluent Bit to an output (data plane
# events are also routed through the control plane instance
# in order to support internal audit functionality).
#
# This tag helps correctly configure the scenario described above. The
# customer output rules for the external audit collector in this control plane
# Fluent Bit config file, should look at only events with the
```

```

`incoming.excludingDataPlaneEvents` tag. See the example under `outputs`.
The data
# plane Fluent Bit config file would send *all* it's events to the external
audit collector, as it will then fill the gap for the data plane events.
#
# The `incoming` tag will still include all events, including data plane
ones as well. This is used when sending to the `/audit-record` URL, which
# receives all internal events, *including* those from the data plane, from
the control plane Fluent Bit. This `[OUTPUT]` should not be changed unless
# the customer intends to change what audit events the software itself
collects.
[FILTER]
    name rewrite_tag
    match incoming
    rule $header['event']['action'] ^(?!(bypass\.policy|policy\.resolution|
authentication\.user)$).* incoming.excludingDataPlaneEvents true

[OUTPUT]
    name null
    match incoming
[OUTPUT]
    name http
    match incoming
    host control-plane-api
    port 8079
    uri /audit-records
    format json
    json_date_key false

# The following shows an example for how to configure an output to a remote
source for the control plane,
# whilst excluding events routed through this Fluent Bit from the data
plane, in order
# to prevent duplicate events. See comment under `filters` for full details.
#
# [OUTPUT]
#     name http
#     match incoming.excludingDataPlaneEvents # Avoids sending events from
the DP
#     host myhost.example
#     port 8080
#     format json

```

Configure the Data Plane Fluent Bit

The data plane Fluent Bit instance receives logs relating to audit events originating from the data plane, primarily policy resolution events.

The data plane Fluent Bit instance forwards these events into internal services to the control plane Fluent Bit.

Both the control plane and the data plane Fluent Bit components have their own Fluent Bit configuration map within their respective namespaces. Each data plane will have its own Fluent Bit component and configuration map.

If you have made changes to the port numbers in use, modify the configuration map accordingly. If you are planning to route to your SIEM tool as well, see [Configure Log Routing to a SIEM Tool](#).

Data Plane Fluent Bit Default Configuration Map

```
[SERVICE]
    Daemon Off
    Flush 1
    Log_Level info
    Parsers_File parsers.conf
    Parsers_File custom_parsers.conf
    HTTP_Server On
    HTTP_Listen 0.0.0.0
    HTTP_Port 2020
    Health_Check On

[INPUT]
    name http
    host 0.0.0.0
    port 8888

[FILTER]
    Name kubernetes
    Match kube.*
    Merge_Log On
    Keep_Log Off
    K8S-Logging.Parser On
    K8S-Logging.Exclude On

# This output forwards data plane events to the deeper into the system,
# eventually landing in the control plane.
# It's purpose is to support our own internal audit functionality. Usually,
# it should not be changed.
[OUTPUT]
    name http
    match *
    host data-agent
    port 8540
    uri /audit-records
    format json
    json_date_key false

# The following shows an example for how to configure an output to a remote
# source for the data plane. It is generally
# recommended to setup an external audit provider on both the data plane
# configuration (this config) and the control
# plane configuration in order to reduce the critical path for events to
# reach said system. Please see the control plane
# configuration comments for further details on this setup.
#
# [OUTPUT]
#     name http
#     match *
#     host myhost.example
```

```
# port 8080
# format json
```

Configure Log Routing to a SIEM Tool

To enable log events to flow to your security information and event management (SIEM) tool, you will need to:

- Individually route the output from each data plane Fluent Bit instance to the SIEM tool.
- Map the control plane Fluent Bit instance to output to the SIEM tool.

As data plane events are routed to the control plane Fluent Bit instance for collection and display, you need to include an explicit filter to ensure that these data plane events are not routed further by the control plane Fluent Bit instance and duplicated in your external SIEM tool.

To mitigate this possibility, a new filter is included in the default Fluent Bit configuration. This includes comments that explain the requirements as a reminder. To use this filter, you need to include in the new output rule for the SIEM tool the following parameter:

```
match incoming.excludingDataPlaneEvents
```

This ensures that only events from the control plane that are specifically not data plane events are forwarded to your external SIEM tool.

Fluent Bit–Splunk Integration

The following is an example of using Fluent Bit to forward audit records to Splunk.

1. Set up a Splunk Cloud instance:
 - a. Go to [Splunk](#).
 - b. Create a cloud trial account.
 - c. Log in to the Splunk Cloud Platform using the URL in the Splunk welcome email.
 - d. Click **Settings** → **Data input** → **Select HTTP Event Collector**.
 - e. Click **New Token**.
 - f. Enter a name, such as *dsp-token*.
 - g. Click **Next**.
 - h. Select the main index.
 - i. Click **Review**.
 - j. Click **Submit**.
 - k. Copy the token value.
2. Update the Fluent Bit config map for the control plane instance:
 - a. `kubectl edit configmap fluent-bit -n NAMESPACE`
 - b. Add a new output section with the following properties (replace `host` and `splunk_token`):

```
[OUTPUT]
  name splunk
  match incoming.excludingDataPlaneEvents
```



```
host prd-p-wma5y.splunkcloud.com
splunk_token aad8350d-24a9-4b8d-a6eb-5d6a729927cf
port 8088
tls on
tls.verify off
```

3. Update the Fluent Bit config map for each data plane instance:
 - a. `kubectl edit configmap fluent-bit -n NAMESPACE`
 - b. Add a new output section with the following properties (replace `host` and `splunk_token`):

```
[OUTPUT]
  name splunk
  host prd-p-wma5y.splunkcloud.com
  splunk_token aad8350d-24a9-4b8d-a6eb-5d6a729927cf
  port 8088
  tls on
  tls.verify off
  match *
```

2.3. Upgrade the Platform

Upgrading the platform involves upgrading the control plane and upgrading each data plane.



Important

When upgrading the Privitar Data Security Platform (DSP), you must upgrade both the control plane and the data plane to the same version.



Note

You cannot migrate the platform to a different Kubernetes cluster during an upgrade.



Note

The upgrade process only upgrades changed components.

2.3.1. Back Up Fluent Bit Configurations

The upgrade will overlay the existing Fluent Bit configurations. If you have changed any configurations during the original installation or subsequently, you must restore these configurations need after completing the upgrade.

To open and back up the existing Fluent Bit configuration maps, complete the following steps for the control plane and each data plane:

1. Open the Fluent Bit configuration map:Open the Fluent Bit configuration map:

```
kubectl edit configmap fluent-bit -n <namespace>
```

2. Copy and save the complete configuration map.

2.3.2. Upgrade the Control Plane

When you upgrade the control plane, you update the versions of Privitar Data Security Platform and all third-party services in the control plane. The automated upgrade process performs a schema migration for the platform's services.



Note

The upgrade process does not regenerate certificates.

Upgrade the Control Plane: Prerequisites

Perform the following prerequisites before you upgrade the control plane:

1. Ensure that you have an existing installation of the control plane from the previous version of the platform.
2. Scale down the services using the following commands:

```
kubectl scale deployment dpp-emp-api --replicas=0 -n <namespace>
kubectl scale deployment control-plane-api --replicas=0 -n <namespace>
```

3. [Create a backup of the current installation.](#)
4. From within the installation folder of the upgrade version, configure the `control-plane-values.yaml` file, as described in [Configure the control-plane-values File](#) with the following exceptions:

Table 21. Upgrade Changes to control-plane-values.yaml File

Value	Notes
<code>clean</code>	Set to <code>false</code> to preserve the existing pods.
<code>namespace</code>	Use the same as the existing.
<code>externalHostname</code>	Use the same as the existing.
<code>istio, install</code>	Set to <code>false</code> to prevent overwriting existing certs.
<code>tls, autoGenerate</code>	Set to <code>false</code> to prevent overwriting existing certs.
<code>tls, autoGenerateSecret</code>	Comment out this value.
<code>tls, existingCertificateSecret</code>	Edit to match the current generated cert name.
<code>tls, keyFile</code>	Comment out this value.
<code>tls, certificateFile</code>	Comment out this value.

Value	Notes
images, load	Set to <code>true</code> to load the upgrade images. This is important if this is the first time that the upgrade is run in the environment.
images, push	Set to <code>true</code> to push the upgrade images. This is important if this is the first time that the upgrade is run in the environment.

Upgrade the Control Plane: Installation

To install the new version of the control plane:

1. From within the installation folder, run the following installer upgrade command:

```
./dppctl upgrade -f ./control-plane-values.yaml -u privitar -p password
```



Note

Username and password are not required if LDAP is currently enabled.

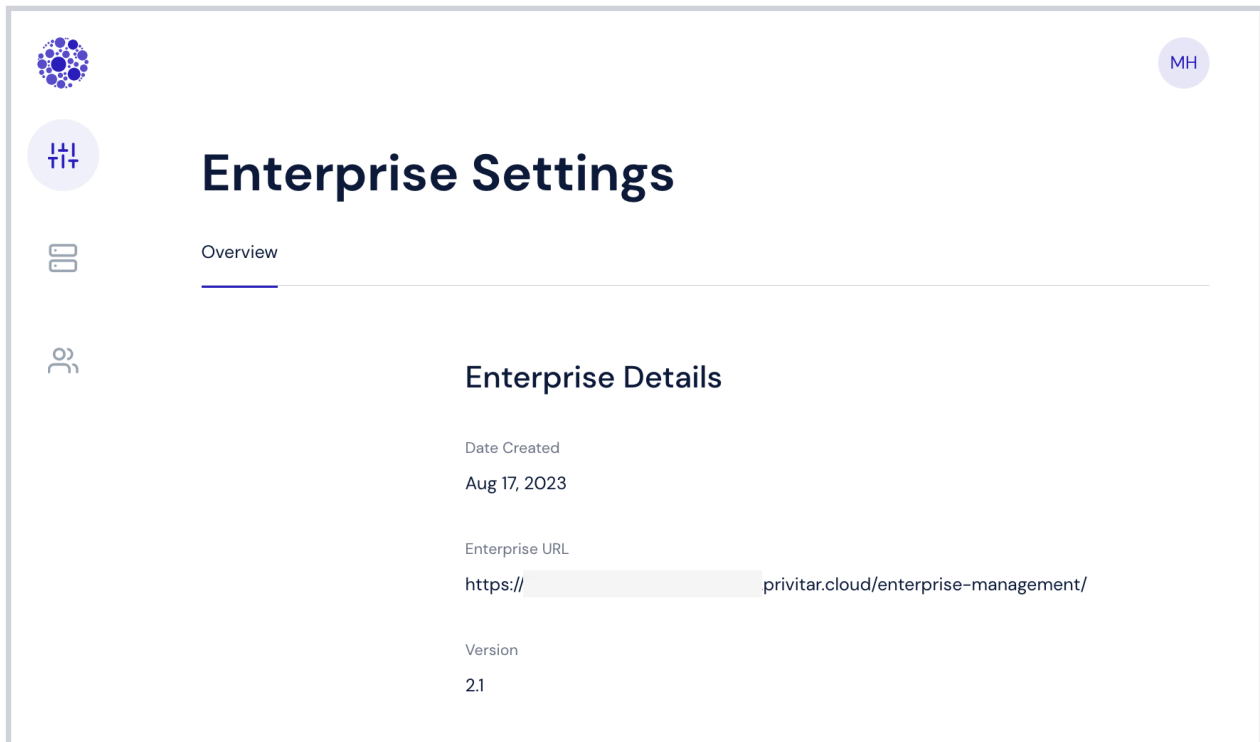
2. Bring the services back up, using the following commands:

```
kubectl scale deployment dpp-emp-api --replicas=3 -n <namespace>
kubectl scale deployment control-plane-api --replicas=3 -n <namespace>
```

Upgrade the Control Plane: Validation

To ensure that the new version of the control plane installed correctly, perform the following steps:

1. Restore administrator access to the platform.
2. Ensure that the new version number appears in the platform's user interface (UI):
 - a. Log in to the platform's UI.
 - b. Click Enterprise Settings in the left navigation.
Or click the avatar symbol in the top right corner of any page, and click **About**.
 - c. Note the number that appears under **Version**.



3. Ensure that user management functions restored correctly:
 - a. Click **User Management** in the left navigation.
 - b. Check that users and user groups appear on the User Management tab.
 - c. If you had enabled LDAP, ensure that LDAP is still enabled.
4. Ensure that projects restored correctly in each data exchange:
 - a. Click **Data Exchanges** in the left navigation, and select **Open Data Exchange**.
 - b. Click the avatar symbol in the top right corner of the page and select **Manage Exchange**.
 - c. Ensure that business information (tags, terms, data classes), projects, datasets, assets, and so on display correctly.
 - d. Click the avatar symbol in the top right corner of the page, and select **View Exchange**.
 - e. Verify that the right groups and users are assigned to the data exchange.
 - f. Repeat these steps for each data exchange.

If all validation steps pass, [upgrade the data planes](#).

If you experience upgrade issues, please [contact Privitar Support](#).

2.3.3. Upgrade Data Planes

When you upgrade data planes, you update the versions of Privitar Data Security Platform on the data planes.



Important

You must repeat the steps in this section for each of your existing data planes.

**Note**

The upgrade process does not regenerate certificates.

Upgrade Data Planes: Prerequisites

Perform the following prerequisites before you upgrade data planes:

1. Ensure that you have existing installations of one or more data planes: from the *previous* version of the platform.
2. Block all user access to the current installation of the platform.
3. Ensure that you have an existing installation of the control plane from the *current* version of the platform.
4. From within the installation folder of the upgrade version, configure the `data-plane-values.yaml` file, as described in [Configure the data-plane-values.yaml File](#) with the following exceptions:

Table 22. Upgrade Changes to data-plane-values.yaml File

Value	Notes
<code>clean</code>	Set to <code>false</code> to preserve the existing pods.
<code>namespace</code>	Use the same as the existing.
<code>externalHostname</code>	Use the same as the existing.
<code>istio, install</code>	Set to <code>false</code> to prevent overwriting existing certs.
<code>tls, autoGenerate</code>	Set to <code>false</code> to prevent overwriting existing certs.
<code>tls, autoGenerateSecret</code>	Comment out this value.
<code>tls, existingCertificateSecret</code>	Edit to match the current generated cert name.
<code>tls, keyFile</code>	Comment out this value.
<code>tls, certificateFile</code>	Comment out this value.
<code>images, load</code>	Set to <code>true</code> to load the upgrade images. This is important if this is the first time that the upgrade is run in the environment.
<code>images, push</code>	Set to <code>true</code> to load the upgrade images. This is important if this is the first time that the upgrade is run in the environment.
<code>dataplaneID</code>	Use the same as the existing.
<code>exchangeId</code>	Use the same as the existing.

Upgrade Data Planes: Installation

To install the new versions of the data planes, run the following installer upgrade command from within the installation folder:

```
./dppctl upgrade -f ./data-plane-values.yaml -u privitar -p password
```

**Note**

Username and password are not required if LDAP is currently enabled.

Upgrade Data Planes: Validation

To ensure that the new version of the data planes installed correctly, perform the following steps:

1. Ensure that all data planes restored correctly:
 - a. Perform a connection test for each connection in the data exchange to ensure that the connection functions properly.
 - b. Run a simple query for each data plane to ensure that you can connect to each one. (Each data exchange has its own data plane.)
2. Verify that the correct data agent and data plane version installed.
3. Confirm that the Secure Hash Algorithm (SHA) values for the data plane images match those in Docker Hub.

The upgrade process is complete.

4. Restore all user access to the platform.

If you experience upgrade issues, please [contact Privitar Support](#).

2.3.4. Restore the Configuration Map for the SIEM Tool

To reconfigure the Fluent Bit configuration map following the upgrade, complete the following steps for the control plane and each data plane:

1. Open the Fluent Bit configuration map:Open the Fluent Bit configuration map:

```
kubectl edit configmap fluent-bit -n <namespace>
```

2. Copy and paste the prior configuration map for the respective control plane or data planes.
3. Save the configuration map.

2.3.5. Restart the Fluent Bit Pods

After changing the Fluent Bit configuration, restart the Fluent Bit pod in the respective control plane or data plane that has been restored.

3. Additional Deployment Considerations

The following are additional and optional deployment considerations.

3.1. Deploy a Persistent Volume for JDBC Drivers

The platform data plane requires a persistent volume (PV), deployed in the cluster, to hold JDBC drivers for use by the data plane. The persistent volume must have the following directory structure:

```
shared/
  jdbc-drivers/
    1..n JDBC Driver JAR files
data-agent
  EMPTY
dynamic-proxy
  EMPTY
```

If you are using Google BigQuery, unzip its JDBC driver .zip file, and copy the contents (.jar file and library files) into a directory within the `/jdbc-drivers/` directory.

This persistent volume can be delivered using a method most convenient for your specific Kubernetes environment, such as using EFS volumes in AWS.

A persistent volume for the JDBC drivers should be delivered by creating a mountable storage instance, populating it with the above directory structure, and then registering a persistent volume in the cluster. This is typically done using Kubernetes manifest files, deployed using `kubectl`. You must also register a persistent volume claim (PVC). Put the PVC in your data plane configuration YAML file during deployment.

This example persistent volume and PVC mounts an AWS EFS volume containing the directory structure above, as well as the relevant JDBC drivers:

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: data-plane-config-storage-class
provisioner: efs.csi.aws.com
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: data-plane-config-pvc
spec:
  storageClassName: data-plane-config-storage-class
  accessModes:
    - ReadWriteMany
  volumeName: data-plane-config-pv
  resources:
    requests:
      storage: 1Gi
---
```

```

apiVersion: v1
kind: PersistentVolume
metadata:
  name: data-plane-config-pv
spec:
  accessModes:
    - ReadWriteMany
  capacity:
    storage: 5Gi
  csi:
    driver: efs.csi.aws.com
    volumeHandle: [EFS ID]
persistentVolumeReclaimPolicy: Retain
storageClassName: data-plane-config-storage-class
volumeMode: Filesystem

```

3.2. Configure Single Sign-On

You can configure the Privitar Data Security Platform to work with your organization's single sign-on (SSO) identity provider so that users can access the platform without the need to re-enter login information.



Note

All users logging in to the platform with SSO must also have their identity registered in LDAP.

3.2.1. Set Up OIDC SSO

OpenID Connect (OIDC) single sign-on (SSO) requires the following services to communicate with each other:

- **Service provider**—The Privitar Data Security Platform (the platform) is the service provider.
- **Identity provider**—The identity provider (IdP) is the service that manages identity information for your organization. Examples include Okta, Microsoft Entra ID (formerly Azure Active Directory), auth0, Ping, and VMware. The examples in this document use Okta as the IdP, but the instructions apply to all IdPs.
- **Broker**—The platform supports Keycloak as the SSO broker.

Start the Broker Setup

After you [configure LDAP](#) on the platform, set up a broker to work with your identity provider. The platform supports Keycloak as the SSO broker.

1. Obtain the Keycloak admin password by entering the following Kubernetes command (replace <control-plane-namespace> with the name of your control plane):

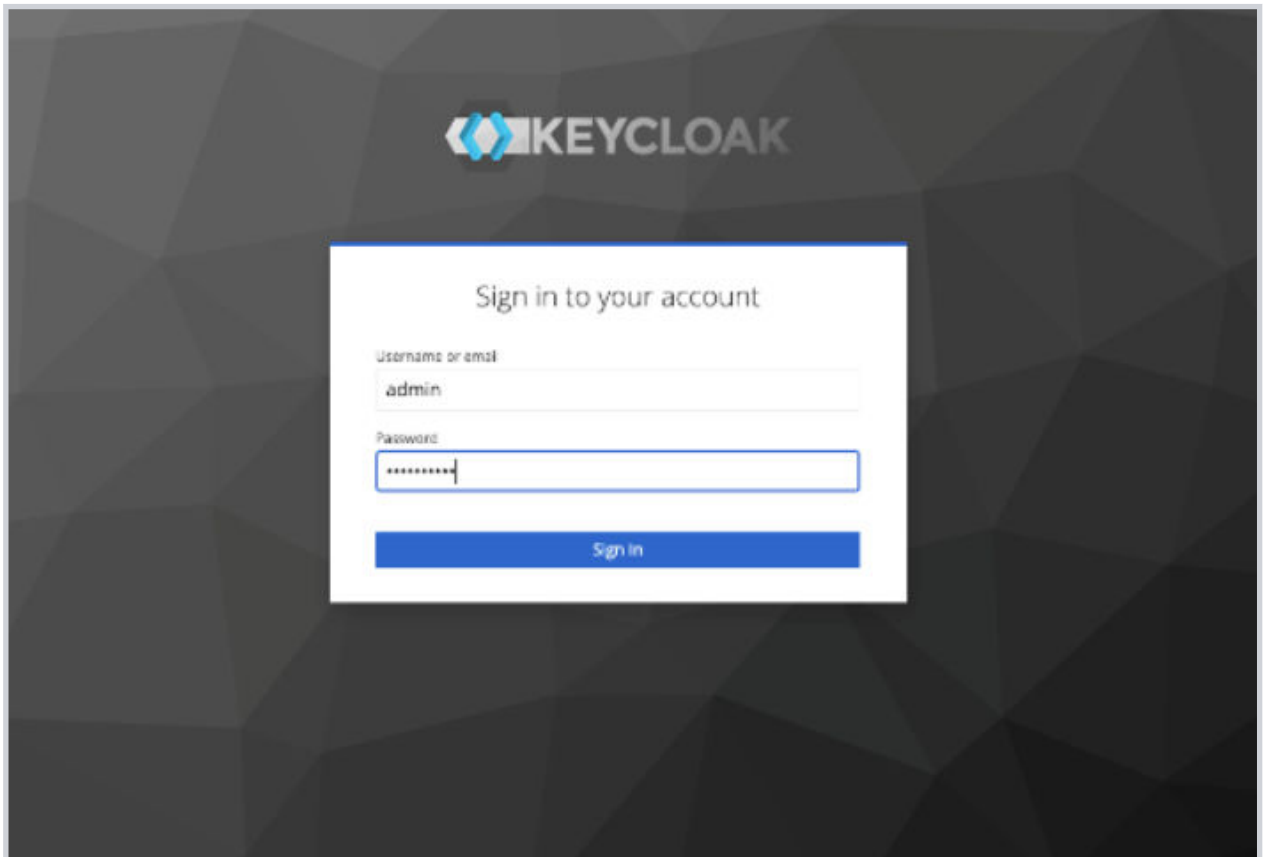
```

kubectl get secret dpp-keycloak-admin-secret -n <control-plane-namespace> -o jsonpath='{.data.admin-password}' | base64 --decode

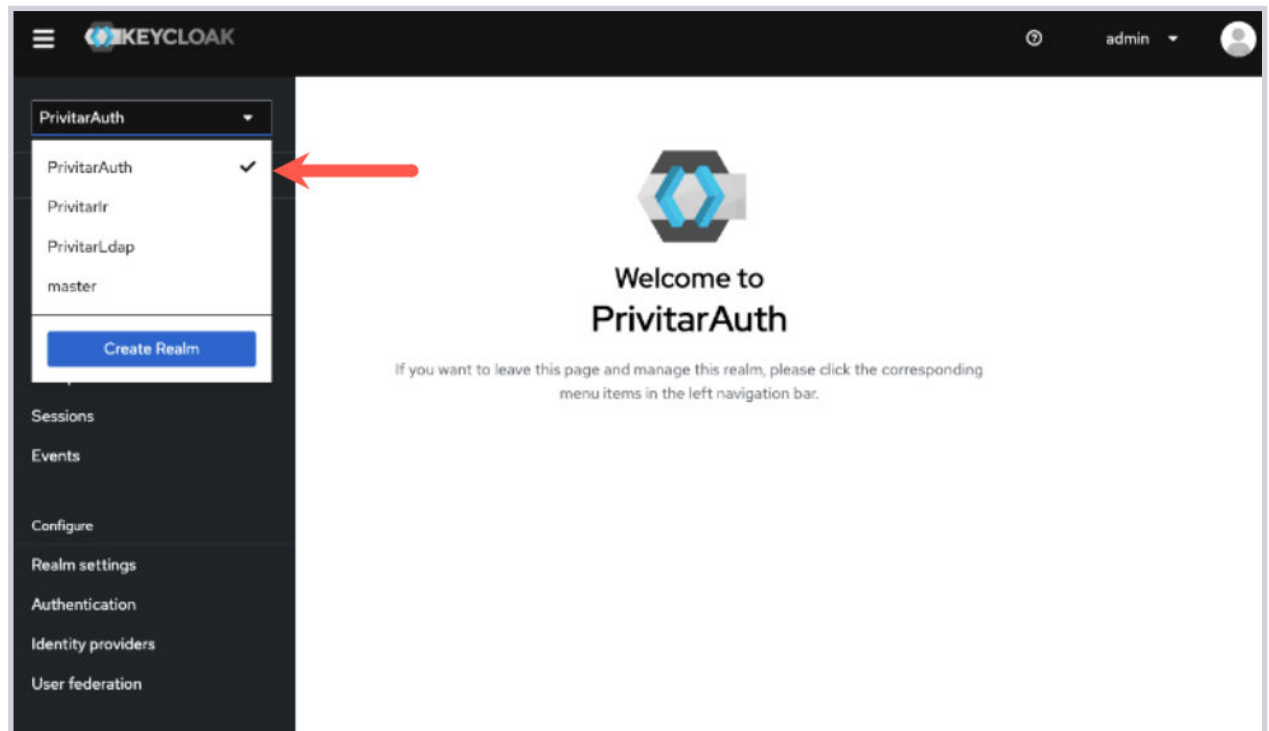
```


2. Copy the value for `dpp-keycloak-admin-secret` (generated by the DSP installation package).
3. Go to the Keycloak admin console in your browser at `https://<domain>/keycloak/admin`.

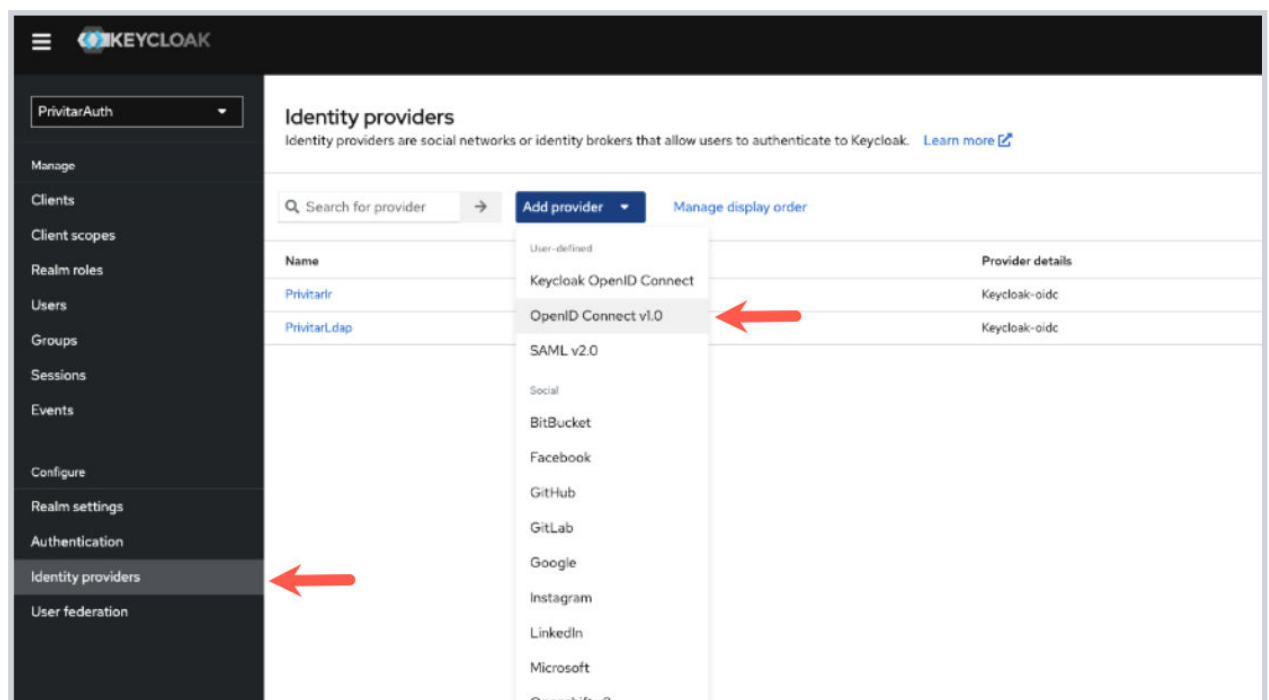
A login page appears.



4. Log in to the Keycloak admin console:
 - **Username or email**—admin
 - **Password**—[This is the value for `dpp-keycloak-admin-secret` that you copied in a previous step.]
5. Select **PrivitarAuth** from the Realm menu.



6. Click **Identity providers** in the left navigation.
7. Select **OpenID Connect v1.0** from the **Add provider** menu.



The **Add OpenID Connect provider** page appears.

The screenshot shows the Keycloak Admin Console interface. On the left, a sidebar menu is visible with the 'PrivitarAuth' realm selected. The main content area is titled 'Add OpenID Connect provider'. It contains four input fields: 'Redirect URI' (with a value starting with 'https://i/.../keycloak/realms/PrivitarAuth/broker/oidc/endpoint'), 'Alias' (with the value 'okta-acme'), 'Display name' (with the value 'Acme's Okta'), and 'Display order' (which is empty).

8. **Redirect URI**—Copy the redirect URI, which is in the following format:

```
https://<domain>/<service_provider>/realms/<realm_name>/broker/<alias>/
endpoint
```

For example:

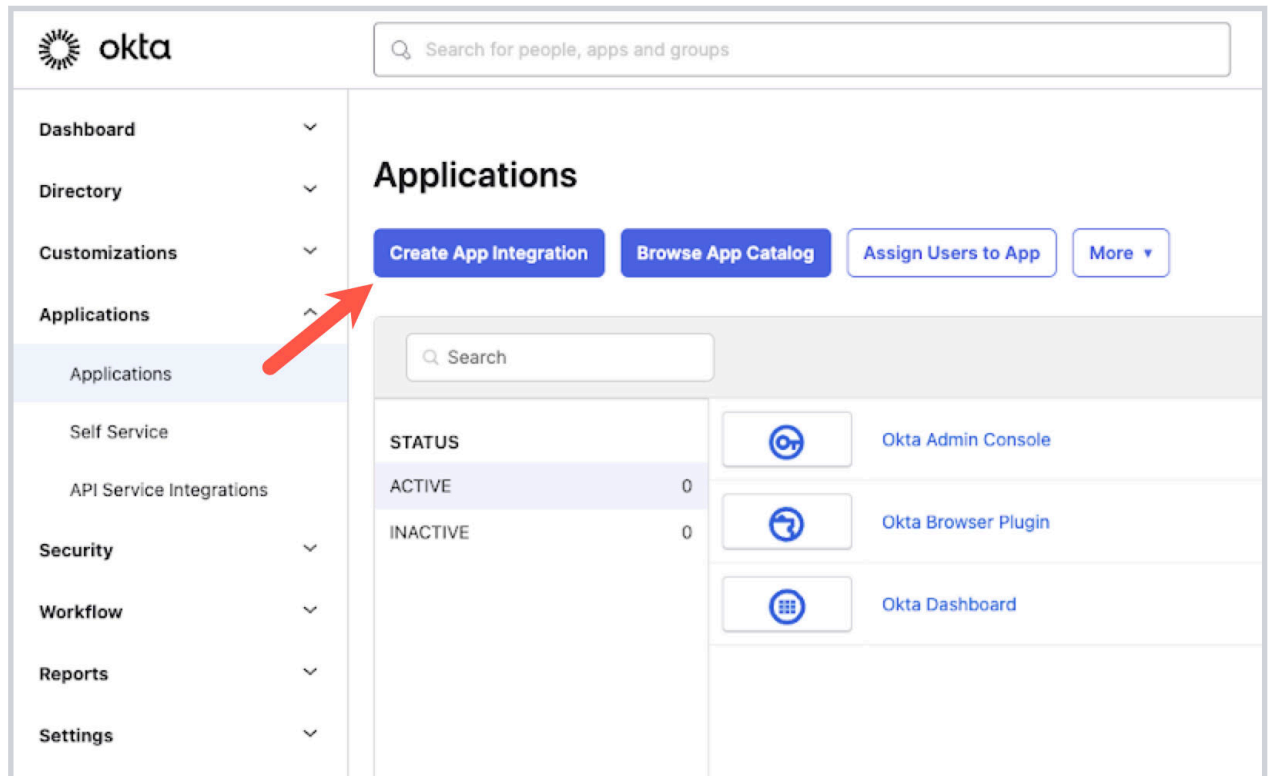
```
https://acme.privitar.cloud/keycloak/realms/PrivitarAuth/broker/acme-
okta/endpoint
```

9. **Alias**—Specify a unique alias for the identity provider.
This alias will be a component of the redirect uniform resource identifier (URI).
10. **Display name**—Specify a display name for the identity provider that will show up in the user interface (UI) of the Keycloak admin console.
11. Follow the steps in [Set Up an Identity Provider](#).

Set Up an Identity Provider

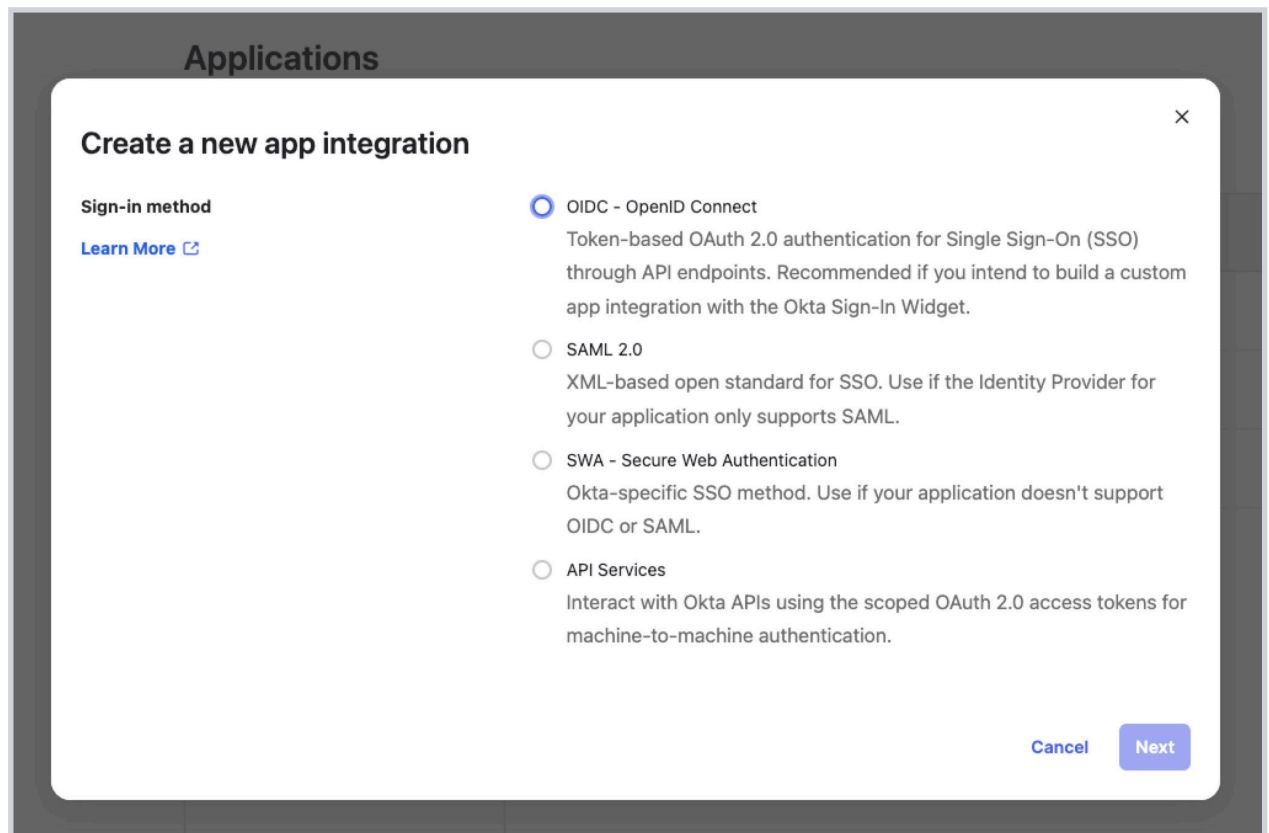
After you [configure LDAP](#) on the platform and set up a broker, you can set up an identity provider.

1. Log in to your identity provider.
These instructions use Okta as an example.
2. Go to **Applications**.
3. Click **Create App Integration**.



The **Create a new app integration window** opens.

4. Select **OIDC**.



The **Application type** section appears.

5. Select **Web Application**.

30 days left in your trial. [Discuss your needs](#)

Create a new app integration

Sign-in method

[Learn More](#)

☒ **OIDC - OpenID Connect**
Token-based OAuth 2.0 authentication for Single Sign-On (SSO) through API endpoints. Recommended if you intend to build a custom app integration with the Okta Sign-In Widget.

☐ **SAML 2.0**
XML-based open standard for SSO. Use if the Identity Provider for your application only supports SAML.

☐ **SWA - Secure Web Authentication**
Okta-specific SSO method. Use if your application doesn't support OIDC or SAML.

☐ **API Services**
Interact with Okta APIs using the scoped OAuth 2.0 access tokens for machine-to-machine authentication.

Application type

What kind of application are you trying to integrate with Okta?

Specifying an application type customizes your experience and provides the best configuration, SDK, and sample recommendations.

☒ **Web Application**
Server-side applications where authentication and tokens are handled on the server (for example, Go, Java, ASP.Net, Node.js, PHP)

☐ **Single-Page Application**
Single-page web applications that run in the browser where the client receives tokens (for example, Javascript, Angular, React, Vue)

☐ **Native Application**
Desktop or mobile applications that run natively on a device and redirect users to a non-HTTP callback (for example, iOS, Android, React Native)

Cancel

Next

6. Click **Next**.

The **New Web App Integration** page appears.

New Web App Integration

General Settings

App integration name

Privitar Data Security Platform

Logo (Optional)



Grant type

[Learn More](#)

Client acting on behalf of itself

☐ Client Credentials

Client acting on behalf of a user

☒ Authorization Code

☐ Refresh Token

☐ Implicit (hybrid)

Sign-in redirect URIs

Okta sends the authentication response and ID token for the user's sign-in request to these URIs

[Learn More](#)

☐ Allow wildcard * in sign-in URI redirect.

https://

+ Add URI

Sign-out redirect URIs (Optional)

After your application contacts Okta to close the user session, Okta redirects the user to one of these URIs.

[Learn More](#)

https://

+ Add URI

Trusted Origins

Base URIs (Optional)

Required if you plan to self-host the Okta Sign-in Widget. With a Trusted Origin set, the Sign-in Widget can make calls to the authentication API from this domain.

[Learn More](#)

+ Add URI

Assignments

Controlled access

Select whether to assign the app integration to everyone in your org, only selected group(s), or skip assignment until after app creation.

☒ Allow everyone in your organization to access

☐ Limit access to selected groups

☐ Skip group assignment for now

Enable immediate access (Recommended)

Recommended if you want to grant access to everyone without pre-assigning your app to users and use Okta only for authentication.

☒ Enable immediate access with **Federation Broker Mode**



To ensure optimal app performance at scale, Okta End User Dashboard and provisioning features are disabled. [Learn more about Federation Broker Mode](#).

7. **App integration name**—Enter a name for the app.
8. **Sign-in redirect URIs**—Specify the sign-in URI.

This is the URI that you noted down while following the steps in [Start the Broker Setup](#).

9. **Sign-out redirect URIs**—Specify the sign-out URI.

You form this by adding `/logout_response` to the end of the sign-in URI. For example:

```
https://acme.privitar.cloud/keycloak/realms/PrivitarAuth/broker/acme-okta/endpoint/logout_response
```

10. **Controlled access**—Specify who should have access to the platform.
11. Click **Save**.

A confirmation page appears.

← Back to Applications

Privitar Data Security Platform

Active View Logs

General Sign On Assignments Okta API Scopes Application Rate Limits

Client Credentials Edit

Client ID 01...97 🔑

Public identifier for the client that is required for all OAuth flows.

Client authentication ☒ Client secret ☐ Public key / Private key

Proof Key for Code Exchange (PKCE) ☐ Require PKCE as additional verification

CLIENT SECRETS

Generate new secret

Creation date	Secret	Status
Aug 2, 2023	***** 👁 🔑	Active

12. Take note of the domain from the URL in your browser's address bar. For example:

```
1234567-admin.okta.com
```

13. Take note of the client ID.

14. Take note of the client secret.
15. Follow the steps in [Complete the Broker Setup](#).

Complete the Broker Setup

After you [configure LDAP](#) on the platform, set up a broker and an identity provider, you can now complete the broker setup.

1. Log in to Keycloak.
2. Click **Identity providers**.
3. Click the **Add provider** menu, and select **OpenID Connect v1.0**.

The **Add OpenID Connect provider** page appears.

The screenshot shows the 'Add OpenID Connect provider' page in the Keycloak administration console. The left sidebar contains navigation links: Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers (selected), and User federation. The main content area is titled 'Add OpenID Connect provider' and includes the following fields:

- Redirect URI**: `https://privitar.cloud/keycloak/realms/PrivitarAuth/broker/oidc/endpoint`
- Alias**: `acme-okta`
- Display name**: `Acme's Okta`
- Display order**: (empty)
- OpenID Connect settings**:
 - Use discovery endpoint**: ☒ On
 - Discovery endpoint**: `https://admin.okta.com/.well-known/openid-configuration` (with a green checkmark icon)
 - Client authentication**: `Client secret sent as post`
 - Client ID**: `097`
 - Client Secret**: (masked with dots)
 - Client assertion signature algorithm**: `Algorithm not specified`
- Show metadata**: (link)

4. **Discovery endpoint**—Enter `https://<identity-provider-domain>/.well-known/openid-configuration`. For example:

```
https://1234567-admin.okta.com/.well-known/openid-configuration
```

The rest of the fields in this section should auto-populate after you enter this URL.

5. **Scopes**—Enter "openid profile email."
6. **Trust email**—Select **On**.
7. **First login flow**—Select "Automatically link existing first login flow."
8. Click **Save**.

Your identity provider now appears in the list of identity providers.

9. Go to **Authentication > Flows**.
10. Click **browser**.
11. **Identity Provider Redirector**—Click the **Settings (gear)** icon.
12. **Default Identity Provider**—Enter the provider alias (for example, `acme-okta`).

13. Click **Save**.

Complete Identity Mappings

After you [configure LDAP](#) on the platform, set up a broker and an identity provider, and complete the broker setup, you can now complete mappings between your broker and your identity provider.

In order to authenticate a user, the platform requires that the following information exists in the token coming from the identity provider:

- username
- email address
- first name
- last name

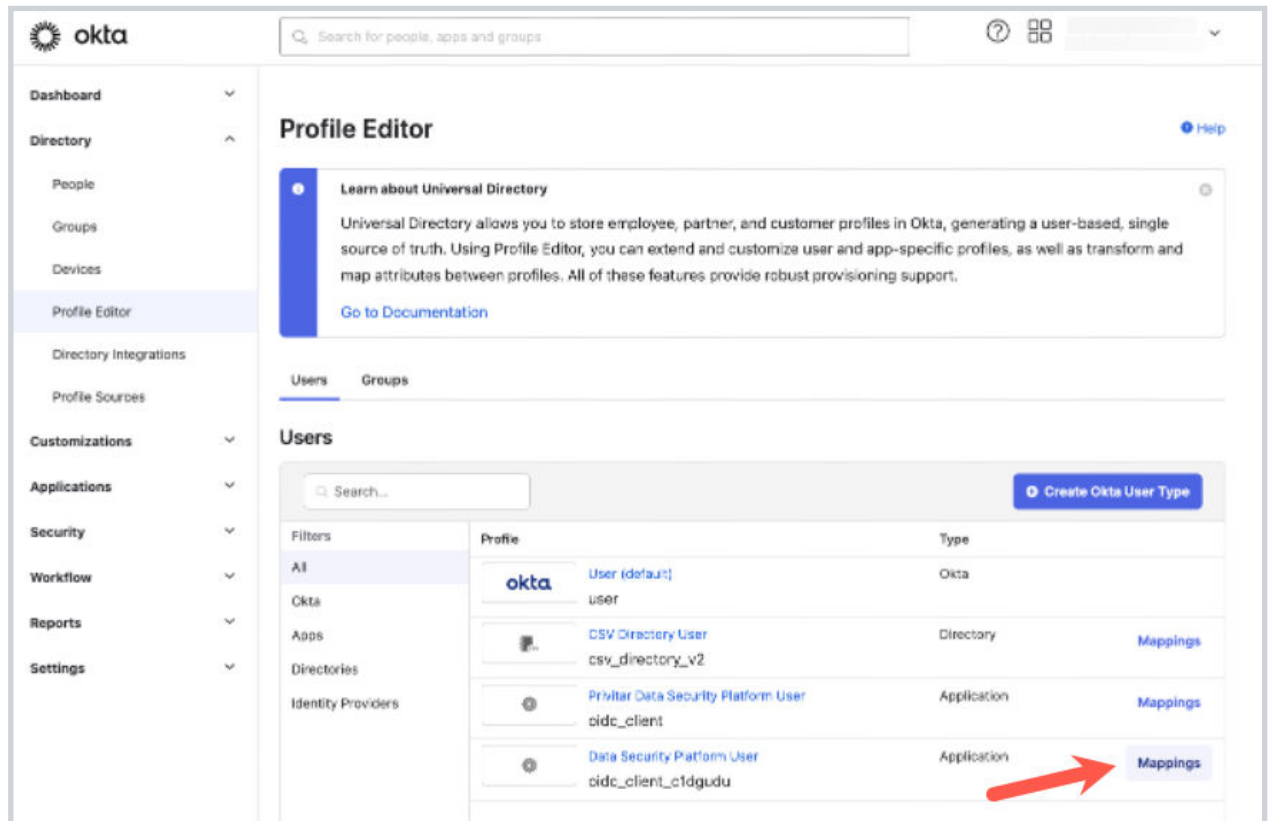
Map these to the following broker user model:

- userName
- email
- firstName
- lastName

Reference your broker's documentation for specific details. For example, if you use Keycloak as a broker, see https://www.keycloak.org/docs/latest/server_admin/#_mappers.

To complete identity mappings between your broker and your identity provider:

1. Log in to your identity provider.
These instructions use Okta as an example.
2. Go to Directory > Profile Editor > [the name of your Privitar DSP profile, for example "Data Security Platform User"]
3. Click **Mappings**.



The User Profile Mappings window appears.

4. Click **Override with mapping**.

Data Security Platform User Profile Mappings

[Data Security Platform to Okta U...](#) [Okta User to Data Security Plat...](#)

Okta User User Profile
 user

Data Security Platform User Profile
 appuser

Username is set by [Data Security Platform](#) · [Override with mapping](#)

user.displayName	→	userName	string
user.nickName	→	name	string
user.firstName	→	nickname	string
user.middleName	→	given_name	string
user.lastName	→	middle_name	string
user.email	→	family_name	string
user.profileUrl	→	email	string
Choose an attribute or enter an expression...	→	profile	string
Choose an attribute or enter an expression...	→	picture	string
Choose an attribute or enter an expression...	→	website	string

- Specify `nickName` as the field from which to populate the username claim.

Data Security Platform User Profile Mappings

Data Security Platform to Okta U... **Okta User to Data Security Plat...**

Okta User User Profile
user

Data Security Platform User Profile
appuser

Choose an attribute or enter an expression...

email	email	Primary email	
title	string	Title	string
displayName	string	Display name	string
nickName	string	Nickname	string
profileUrl	uri	Profile Url	string
secondEmail	email	Secondary email	string
mobilePhone	string	Mobile phone	string
primaryPhone	string	Primary phone	string
streetAddress	string	Street address	string

[Expression Language Reference](#)

userName string

6. Click **Save**.

A confirmation page appears.

Data Security Platform User Profile Mappings

Data Security Platform to Okta U... **Okta User to Data Security Plat...**

Okta User User Profile
user

Data Security Platform User Profile
appuser

user.nickName

[Use default username setting for Data Security Platform](#)

user.displayName

userName string

name string

7. Click **Apply updates now**.

Your OIDC configuration is complete. Users can now sign in to the Privitar Data Security Platform using your identity provider's SSO.

3.2.2. Set Up SAML SSO

Security Assertion Markup Language (SAML) single sign-on (SSO) requires the following services to communicate with each other:

- **Service provider**—The Privitar Data Security Platform (the platform) is the service provider.
- **Identity provider**—The identity provider (IdP) is the service that manages identity information for your organization. Examples include Okta, Microsoft Entra ID (formerly Azure Active Directory), auth0, Ping, and VMware. The examples in this document use Okta as the IdP, but the instructions apply to all IdPs.
- **Broker**—The platform supports Keycloak as the SSO broker.

Start the Broker Setup

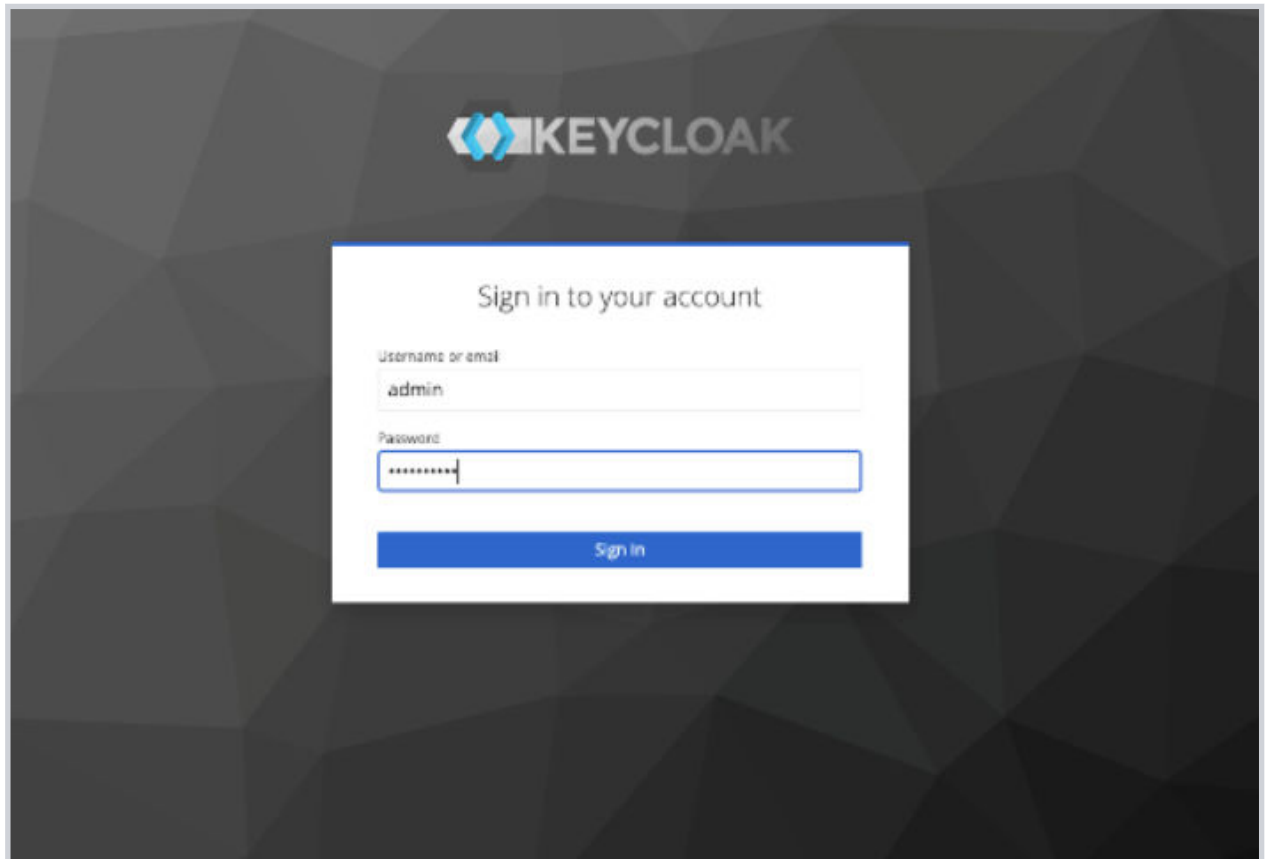
After you [configure LDAP](#) on the platform, set up a broker to work with your identity provider. The platform supports Keycloak as the SSO broker.

1. Obtain the Keycloak admin password by entering the following Kubernetes command (replace `<control-plane-namespace>` with the name of your control plane):

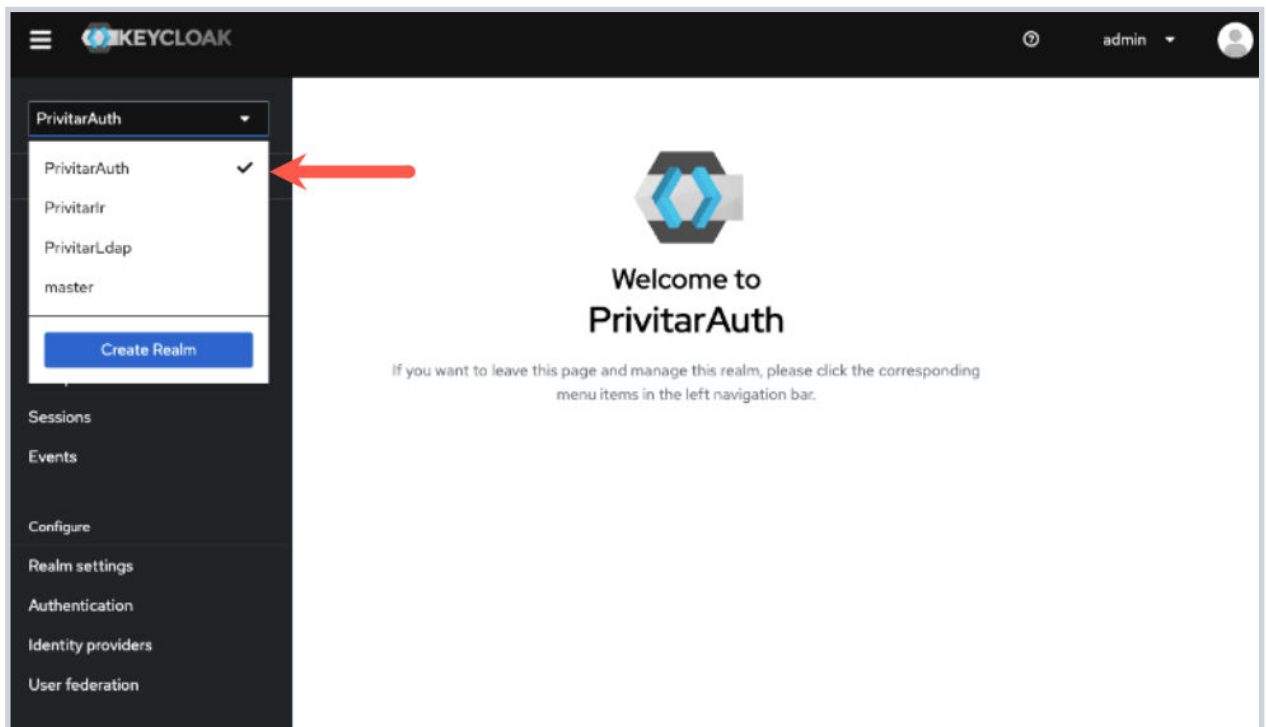
```
kubectl get secret dpp-keycloak-admin-secret -n <control-plane-namespace> -o jsonpath='{.data.admin-password}' | base64 --decode
```

2. Copy the value for `dpp-keycloak-admin-secret` (generated by the DSP installation package).
3. Go to the Keycloak admin console in your browser at `https://<domain>/keycloak/admin`.

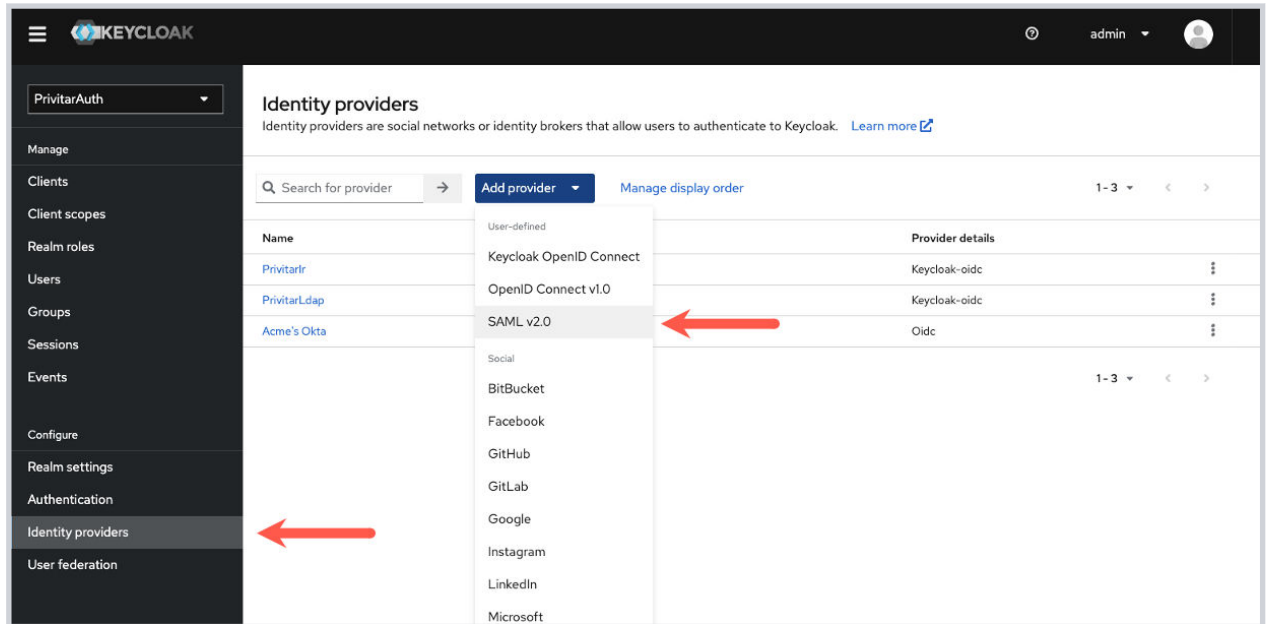
A login page appears.



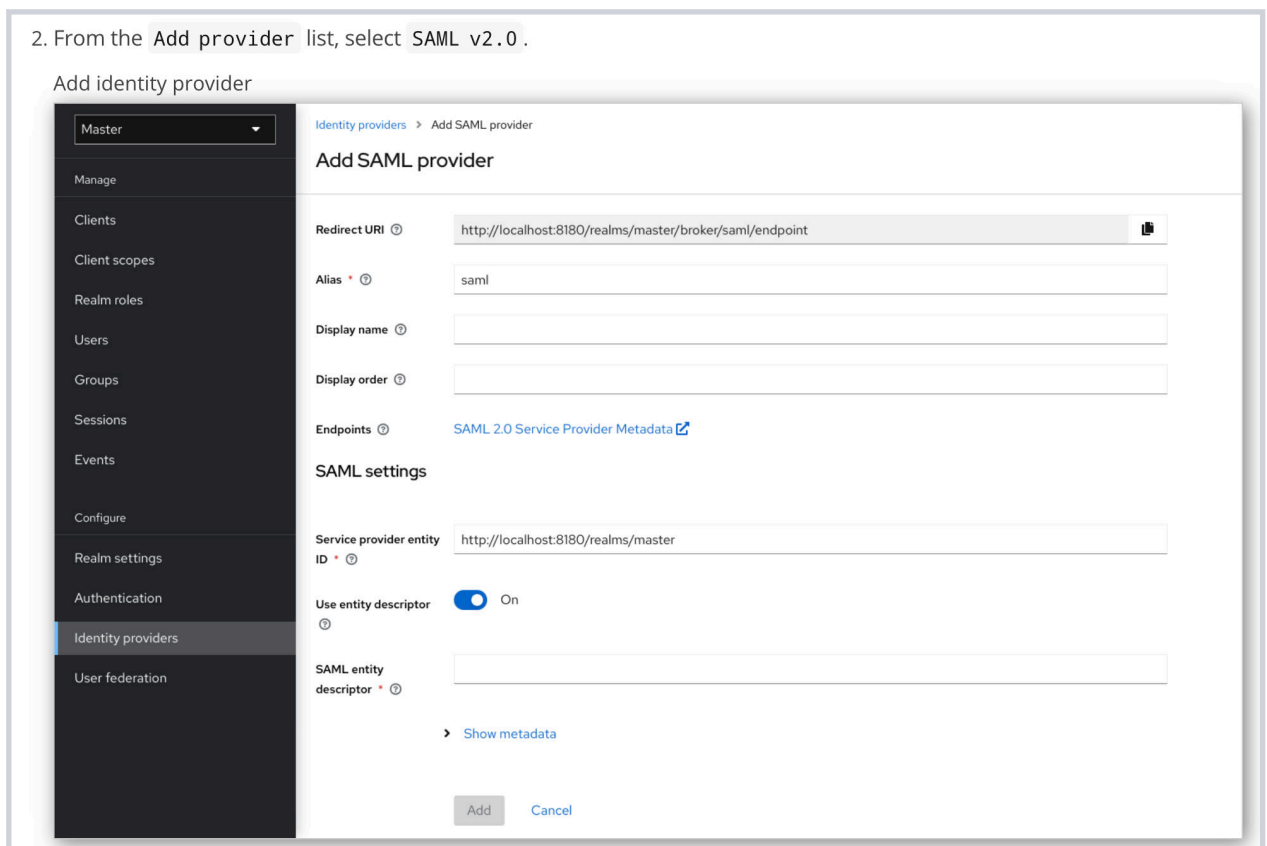
4. Log in to the Keycloak admin console:
 - **Username or email**—admin
 - **Password**—[This is the value for `dpp-keycloak-admin-secret` that you copied in a previous step.]
5. Select **PrivitarAuth** from the Realm menu.



6. Click **Identity providers** in the left navigation.
7. Select **SAML v2.0** from the **Add provider** menu.



The **Add SAML provider** page appears.



8. **Redirect URI**—Copy the redirect URI, which is in the following format:

```
https://<domain>/<service_provider>/realms/<realm_name>/broker/<alias>/endpoint
```

For example:

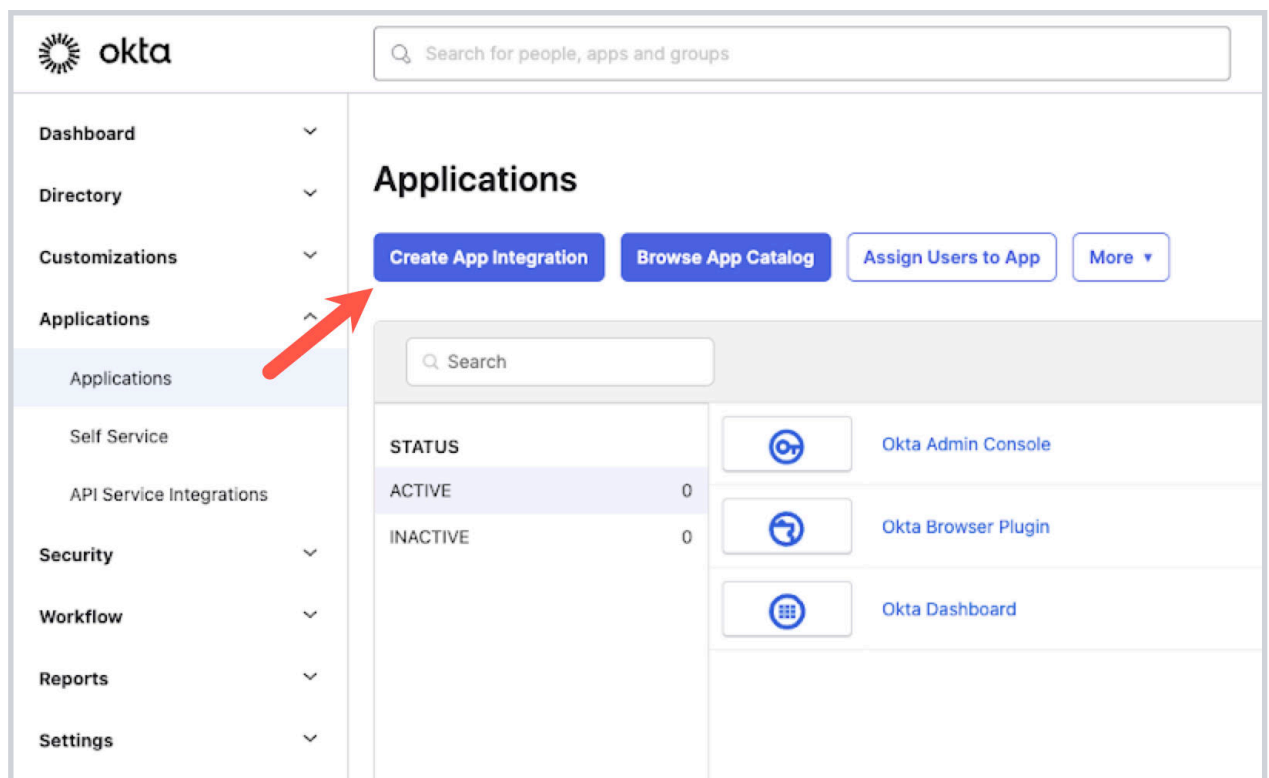
```
https://acme.privitar.cloud/keycloak/realms/PrivitarAuth/broker/acme-okta/endpoint
```

9. **Alias**—Specify a unique alias for the identity provider.
This alias will be a component of the redirect uniform resource identifier (URI).
10. **Display name**—Specify a display name for the identity provider that will show up in the user interface (UI) of the Keycloak admin console.
11. **Service provider entity**—Copy this URL for later use.
12. Follow the steps in [Set Up an Identity Provider](#).

Set Up an Identity Provider

After you [configure LDAP](#) on the platform and set up a broker, you can set up an identity provider.

1. Log in to your identity provider.
These instructions use Okta as an example.
2. Go to **Applications**.
3. Click **Create App Integration**.



The **Create a new app integration window** opens.

4. Select **SAML**.

Applications

Create a new app integration ✕

Sign-in method
[Learn More](#)

- ☐ **OIDC - OpenID Connect**
 Token-based OAuth 2.0 authentication for Single Sign-On (SSO) through API endpoints. Recommended if you intend to build a custom app integration with the Okta Sign-In Widget.
- ☒ **SAML 2.0**
 XML-based open standard for SSO. Use if the Identity Provider for your application only supports SAML.
- ☐ **SWA - Secure Web Authentication**
 Okta-specific SSO method. Use if your application doesn't support OIDC or SAML.
- ☐ **API Services**
 Interact with Okta APIs using the scoped OAuth 2.0 access tokens for machine-to-machine authentication.

Cancel Next

The **Application type** section appears.

5. Click **Next**.

The **Create SAML Integration** page appears.

Create SAML Integration

1 General Settings 2 Configure SAML 3 Feedback

1 General Settings

App name:

App logo (optional):

App visibility: ☐ Do not display application icon to users

Cancel Next

6. **App name**—Enter a name for the app. For example, enter "Data Security Platform SAML".
7. **App logo (optional)**—Upload a logo for the app.

8. **App visibility**—Indicate whether the app icon should appear for users.
9. Click **Next**.

The **Configure SAML** tab appears.

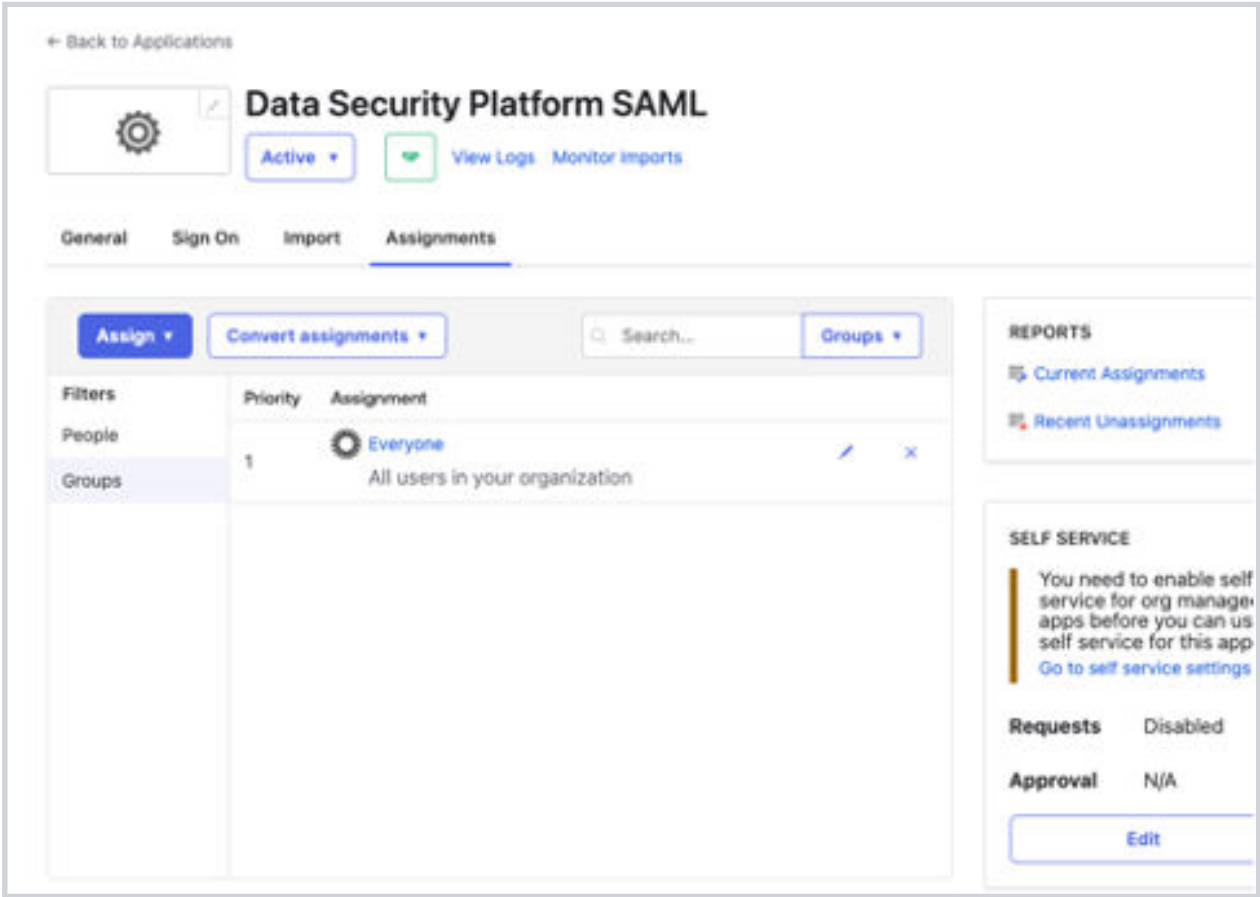
10. **Single sign-on URL**—Enter the redirect URI that you noted down while following the steps in [Start the Broker Setup](#). For example:

```
https://acme.privitar.cloud/keycloak/realms/PrivitarAuth/broker/acme-okta/endpoint
```

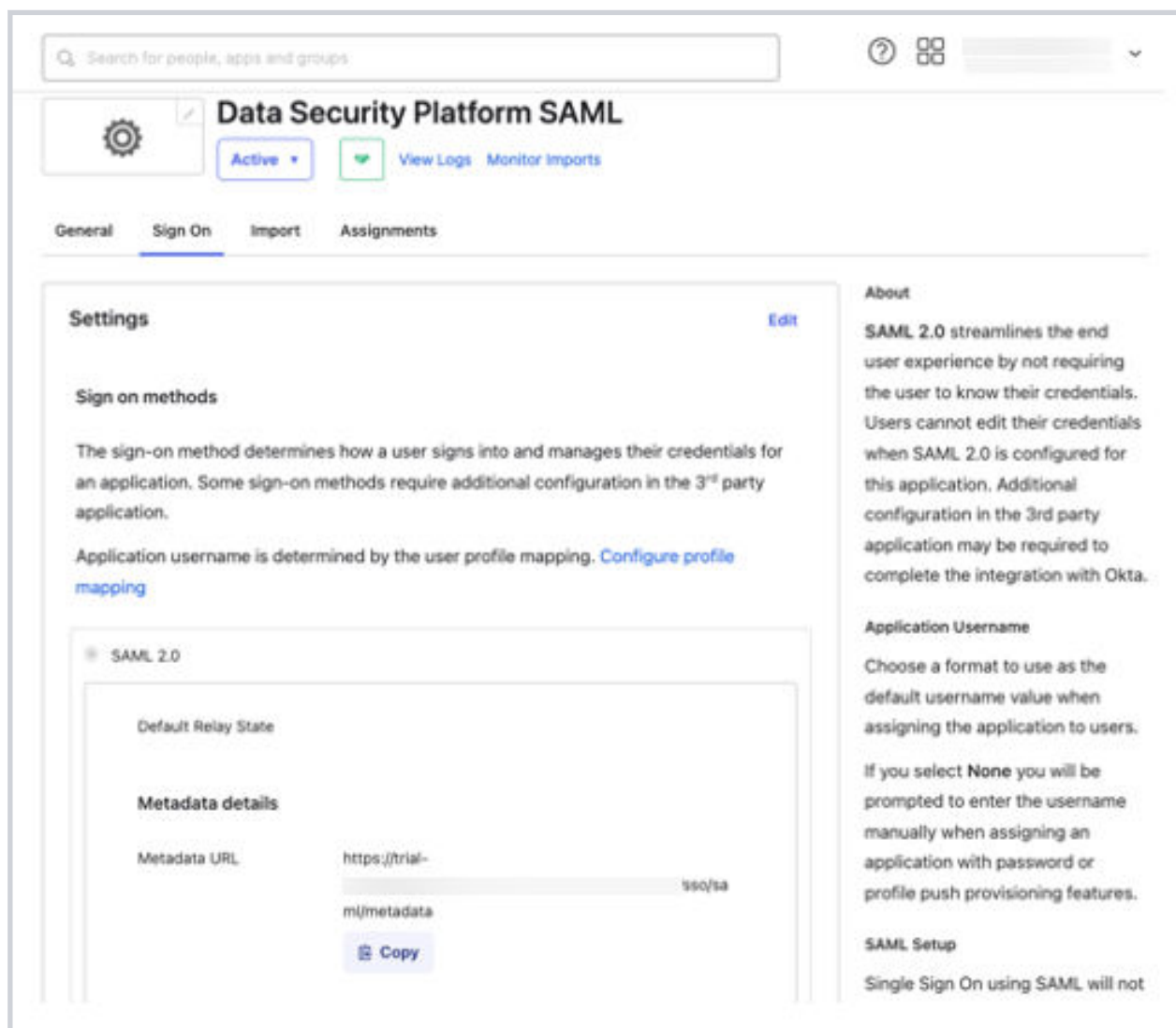
11. Select whether to use this URL for both the recipient URL and destination URL.
12. **Audience URI (SP Entity ID)**—Enter the service provider entity URL that you noted down while following the steps in [Start the Broker Setup](#). For example:

```
https://acme.privitar.cloud/keycloak/realms/PrivitarAuth/
```

13. **Default RelayState**—Leave this field blank.
14. **Name ID format**—Select **X509SubjectName**.
15. **Application username**—Select **Email prefix**.
16. **Update application username on**—Select **Create and update**.
17. Click **Save**.
18. Click the **Assignments** tab.
19. Specify who should have access to the platform.



20. Click the **Sign On** tab.



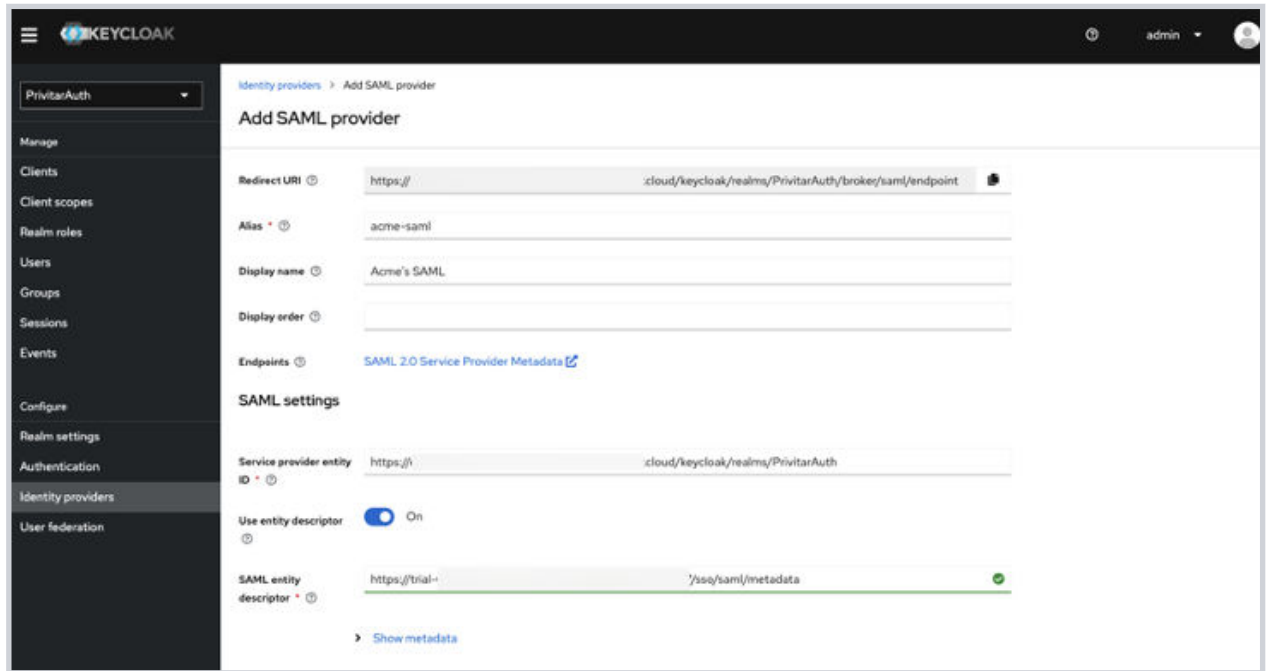
21. **Metadata details**—Click **Copy**.
22. Click **Save**.
A confirmation page appears.
23. Follow the steps in [Complete the Broker Setup](#).

Complete the Broker Setup

After you [configure LDAP](#) on the platform, set up a broker and an identity provider, you can now complete the broker setup.

1. Log in to Keycloak.
2. Click **Identity providers**.
3. Click the **Add provider** menu, and select **SAML v2.0**.

The **Add SAML provider** page appears.



4. **SAML entity descriptor** —Paste the metadata URL that you copied when completing the steps in [Set Up an Identity Provider](#).

The rest of the fields in this section should auto-populate after you enter this URL.

5. Click **Add**.
6. **Trust email** —Select **On**.
7. **First login flow** —Select **Automatically link existing first login flow**.
8. Click **Save**.

Your identity provider now appears in the list of identity providers.

9. Go to Authentication > Flows.
10. Click **browser**.
11. **Identity Provider Redirector** —Click the **Settings (gear)** icon.
12. **Default Identity Provider**—Enter the provider alias (for example, `acme-okta`).
13. Click **Save**.

Complete Identity Mappings

After you [configure LDAP](#) on the platform, set up a broker and an identity provider, and complete the broker setup, you can now complete mappings between your broker and your identity provider.

In order to authenticate a user, the platform requires that the following information exists in the token coming from the identity provider:

- username
- email address
- first name
- last name

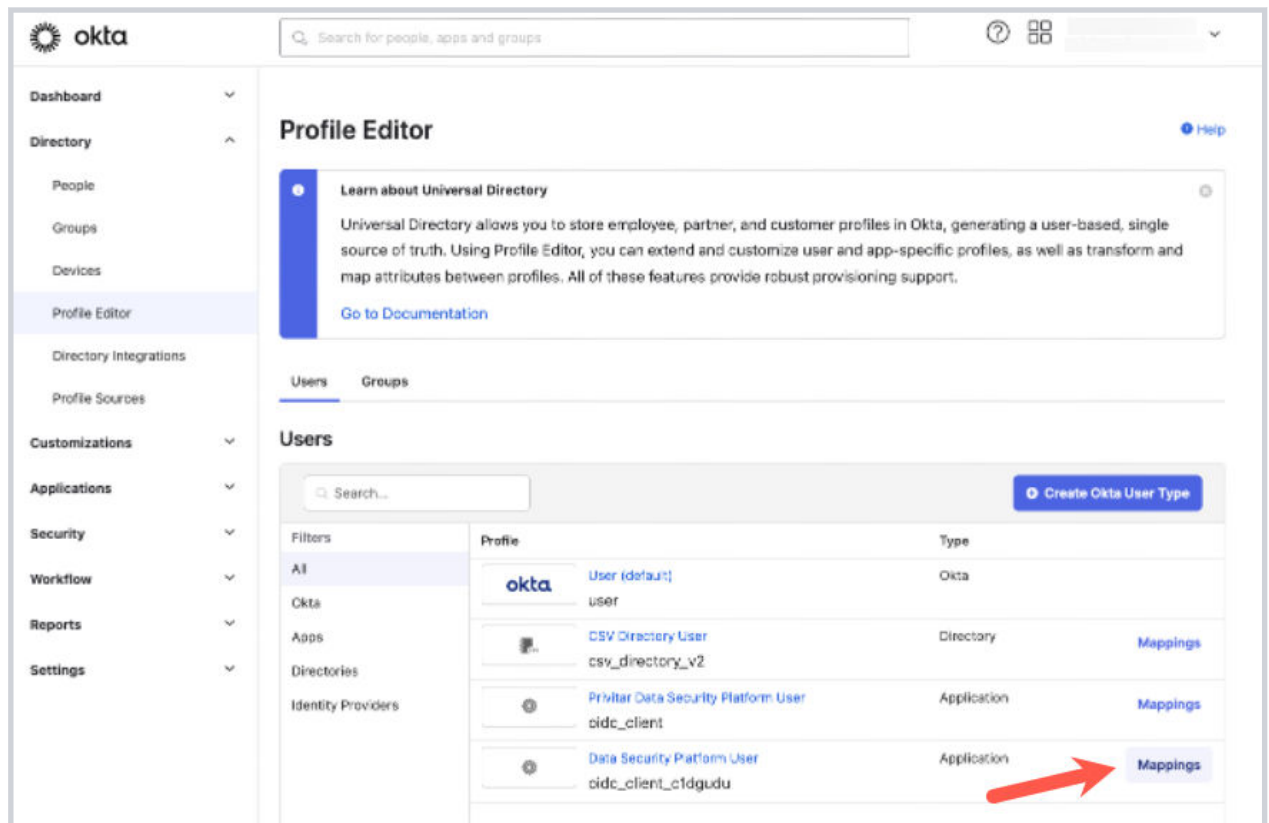
Map these to the following broker user model:

- userName
- email
- firstName
- lastName

Reference your broker's documentation for specific details. For example, if you use Keycloak as a broker, see https://www.keycloak.org/docs/latest/server_admin/#_mappers.

To complete identity mappings between your broker and your identity provider:

1. Log in to your identity provider.
These instructions use Okta as an example.
2. Go to Directory > Profile Editor > [the name of your Privitar DSP profile, for example "Data Security Platform User"]
3. Click **Mappings**.



The User Profile Mappings window appears.

4. Click **Override with mapping**.

Data Security Platform User Profile Mappings

[Data Security Platform to Okta U...](#) [Okta User to Data Security Plat...](#)

Okta User User Profile
 user

Data Security Platform User Profile
 appuser

Username is set by [Data Security Platform](#) · [Override with mapping](#)

user.displayName	→	name	string
user.nickName	→	nickname	string
user.firstName	→	given_name	string
user.middleName	→	middle_name	string
user.lastName	→	family_name	string
user.email	→	email	string
user.profileUrl	→	profile	string
Choose an attribute or enter an expression...	-/>	picture	string
Choose an attribute or enter an expression...	-/>	website	string

- Specify `nickName` as the field from which to populate the username claim.

Data Security Platform User Profile Mappings

Data Security Platform to Okta U... **Okta User to Data Security Plat...**

Okta User User Profile
user

Data Security Platform User Profile
appuser

Choose an attribute or enter an expression...

email	email	Primary email	
title	string	Title	string
displayName	string	Display name	string
nickName	string	Nickname	string
profileUrl	uri	Profile Url	string
secondEmail	email	Secondary email	string
mobilePhone	string	Mobile phone	string
primaryPhone	string	Primary phone	string
streetAddress	string	Street address	string

[Expression Language Reference](#)

userName string

6. Click **Save**.

A confirmation page appears.

Data Security Platform User Profile Mappings

Data Security Platform to Okta U... **Okta User to Data Security Plat...**

Okta User User Profile
user

Data Security Platform User Profile
appuser

user.nickName

[Use default username setting for Data Security Platform](#)

user.displayName

userName string

name string

7. Click **Apply updates now**.

Your SAML configuration is complete. Users can now sign in to the Privitar Data Security Platform using your identity provider's SSO.

3.3. Customize the User Interface

As an enterprise administrator, you can customize the theme of the platform's user interface (UI) to match your organization's logo and color scheme.

1. Locate the `uiTheme` property in the platform's control plane installation package (`control-plane-values.yaml`).
2. Reference the schema called `theme-schema.json` in the `installer` folder. This schema describes the permitted structure of the `uiTheme` value.

A copy of the schema is also in this guide under [User Interface Theme Schema](#).

3. Modify `uiTheme` inside of `control-plane-values.yaml` as needed to match your organization's logo and color scheme by constructing a JSON value that matches the schema described in `theme-schema.json`.

For example, to change the user interface primary colors from the platform's default color scheme to your organization's color palette, you could replace the colors with the appropriate hexadecimal values. The entry for "5" is the lightest color variation, and "100" is the darkest variation.

```
{
  "colors": {
    "components": {
      "hero": {
        "primary": {
          "5": "#FC0518",
          "10": "#F40517",
          "20": "#E40415",
          "40": "#DB0011",
          "80": "#CA0615",
          "60": "#BD0412",
          "100": "#B60210"
        }
      }
    }
  },
}
```

4. To return to the platform's default color scheme, edit `uiTheme` to simply include `{}`.

3.3.1. User Interface Theme Schema

The following is a copy of the schema for the platform's user interface (UI) theme. You can find this schema (a JSON file called `theme-schema.json`) in the `installer` folder in the platform's control plane installation package (`control-plane-values.yaml`).

```
{
  "type": "object",
  "properties": {
    "colors": {
      "type": "object",
      "properties": {
        "primary": {
          "type": "object",
          "properties": {
```

```

        "5": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "10": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "20": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "40": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "60": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "80": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "100": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        }
    },
    "required": [
        "5",
        "10",
        "20",
        "40",
        "60",
        "80",
        "100"
    ],
    "additionalProperties": false,
    "description": "The palette for the primary color of the
theme. `5` is the lightest variation whereas `100` is the darkest variation."
},
    "primaryDark": {
        "type": "object",
        "properties": {
            "5": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            },
            "10": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            },
            "20": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            }
        }
    }
}

```

```

        },
        "40": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "60": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "80": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "100": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        }
    },
    "required": [
        "5",
        "10",
        "20",
        "40",
        "60",
        "80",
        "100"
    ],
    "additionalProperties": false,
    "description": "The palette for the dark variation of
the primary color of the theme. `5` is the lightest variation whereas `100`
is the darkest variation."
},
    "secondary": {
        "type": "object",
        "properties": {
            "5": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            },
            "10": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            },
            "20": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            },
            "40": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            },
            "60": {
                "description": "A hex color string representing
a single color. Example: \"#000000\"."
            },
            "80": {

```

```

        "description": "A hex color string representing
a single color. Example: \"#000000\"."
    },
    "100": {
        "description": "A hex color string representing
a single color. Example: \"#000000\"."
    }
},
"required": [
    "5",
    "10",
    "20",
    "40",
    "60",
    "80",
    "100"
],
"additionalProperties": false,
"description": "The palette for the secondary color of
the theme. `5` is the lightest variation whereas `100` is the darkest
variation."
},
"secondaryDark": {
    "type": "object",
    "properties": {
        "5": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "10": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "20": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "40": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "60": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "80": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        },
        "100": {
            "description": "A hex color string representing
a single color. Example: \"#000000\"."
        }
    }
},
"required": [
    "5",

```

```

        "10",
        "20",
        "40",
        "60",
        "80",
        "100"
    ],
    "additionalProperties": false,
    "description": "The palette for the dark variation of
the secondary color of the theme. `5` is the lightest variation whereas
`100` is the darkest variation."
},
    "components": {
        "type": "object",
        "additionalProperties": {
            "type": "object",
            "properties": {
                "primary": {
                    "type": "object",
                    "properties": {
                        "5": {
                            "description": "A hex color string
representing a single color. Example: `'#000000'`."
                        },
                        "10": {
                            "description": "A hex color string
representing a single color. Example: `'#000000'`."
                        },
                        "20": {
                            "description": "A hex color string
representing a single color. Example: `'#000000'`."
                        },
                        "40": {
                            "description": "A hex color string
representing a single color. Example: `'#000000'`."
                        },
                        "60": {
                            "description": "A hex color string
representing a single color. Example: `'#000000'`."
                        },
                        "80": {
                            "description": "A hex color string
representing a single color. Example: `'#000000'`."
                        },
                        "100": {
                            "description": "A hex color string
representing a single color. Example: `'#000000'`."
                        }
                    }
                },
                "required": [
                    "5",
                    "10",
                    "20",
                    "40",
                    "60",
                    "80",

```

```

        "100"
      ],
      "additionalProperties": false,
      "description": "The palette for the primary
color of the theme. `5` is the lightest variation whereas `100` is the
darkest variation."
    },
    "primaryDark": {
      "type": "object",
      "properties": {
        "5": {
          "description": "A hex color string
representing a single color. Example: `'#000000'`."
        },
        "10": {
          "description": "A hex color string
representing a single color. Example: `'#000000'`."
        },
        "20": {
          "description": "A hex color string
representing a single color. Example: `'#000000'`."
        },
        "40": {
          "description": "A hex color string
representing a single color. Example: `'#000000'`."
        },
        "60": {
          "description": "A hex color string
representing a single color. Example: `'#000000'`."
        },
        "80": {
          "description": "A hex color string
representing a single color. Example: `'#000000'`."
        },
        "100": {
          "description": "A hex color string
representing a single color. Example: `'#000000'`."
        }
      },
      "required": [
        "5",
        "10",
        "20",
        "40",
        "60",
        "80",
        "100"
      ],
      "additionalProperties": false,
      "description": "The palette for the dark
variation of the primary color of the theme. `5` is the lightest variation
whereas `100` is the darkest variation."
    },
    "secondary": {
      "type": "object",
      "properties": {

```

```

        "5": {
            "description": "A hex color string
representing a single color. Example: \"#000000\"."
        },
        "10": {
            "description": "A hex color string
representing a single color. Example: \"#000000\"."
        },
        "20": {
            "description": "A hex color string
representing a single color. Example: \"#000000\"."
        },
        "40": {
            "description": "A hex color string
representing a single color. Example: \"#000000\"."
        },
        "60": {
            "description": "A hex color string
representing a single color. Example: \"#000000\"."
        },
        "80": {
            "description": "A hex color string
representing a single color. Example: \"#000000\"."
        },
        "100": {
            "description": "A hex color string
representing a single color. Example: \"#000000\"."
        }
    },
    "required": [
        "5",
        "10",
        "20",
        "40",
        "60",
        "80",
        "100"
    ],
    "additionalProperties": false,
    "description": "The palette for the
secondary color of the theme. `5` is the lightest variation whereas `100`
is the darkest variation."
},
    "secondaryDark": {
        "type": "object",
        "properties": {
            "5": {
                "description": "A hex color string
representing a single color. Example: \"#000000\"."
            },
            "10": {
                "description": "A hex color string
representing a single color. Example: \"#000000\"."
            },
            "20": {
                "description": "A hex color string

```

```

representing a single color. Example: \"#000000\"."
    },
    "40": {
      "description": "A hex color string
representing a single color. Example: \"#000000\"."
    },
    "60": {
      "description": "A hex color string
representing a single color. Example: \"#000000\"."
    },
    "80": {
      "description": "A hex color string
representing a single color. Example: \"#000000\"."
    },
    "100": {
      "description": "A hex color string
representing a single color. Example: \"#000000\"."
    }
  },
  "required": [
    "5",
    "10",
    "20",
    "40",
    "60",
    "80",
    "100"
  ],
  "additionalProperties": false,
  "description": "The palette for the dark
variation of the secondary color of the theme. `5` is the lightest variation
whereas `100` is the darkest variation."
},
{
  "additionalProperties": false,
  "description": "The set of primary, primaryDark,
secondary and secondaryDark color ranges to override for the given
component."
},
{
  "propertyNames": {
    "enum": [
      "hero"
    ]
  },
  "description": "A map whereby the key represents a
component name to override the core palettes (e.g. \"hero\") for, and the
value a core palette object that will act as the override."
},
{
  "additionalProperties": false,
  "description": "Settings to control the colors of the
application theme."
},
{
  "images": {
    "type": "object",
    "properties": {

```



```
        "logoNav": {
            "description": "The data URI representing the logo used
in the top left corner of the application."
        },
        "additionalProperties": false,
        "description": "A set of images whereby the key is the image to
override and the value is a data uri for an image."
    },
    "additionalProperties": false,
    "$schema": "http://json-schema.org/draft-07/schema#"
}
```

4. Architectural Overview

This section describes the architectural concepts of the Privitar Data Security Platform that you should read and understand before continuing to configure it. There are several components:

- enterprise
- [data exchange](#)
- [control plane](#)
- [data plane](#)
- [data agent](#)
- [data proxy](#)
- [user registry](#)

The platform creates a clear division between:

- The [data plane](#): The independent software services that can directly access and transform data. These services include the data agent and data proxy components.
- The [control plane](#): Those services that control access to the data, but do not directly access it. These services include the data exchange, which allows data owners to classify sensitive datasets, and data consumers to access them, without compromising data safety.

Each organization has a control plane. Within this, one or more enterprises are deployed. An enterprise can have multiple data exchanges, and there can be commonalities across all the data exchanges in an enterprise, such as in user management.

The user registry is the first part of the platform that an enterprise administrator configures. They make decisions on the use of LDAP or the internal user registry. They also make fundamental platform decisions on user and group management. See [Enterprise Administration Tasks](#).

4.1. The Enterprise and the Data Exchange

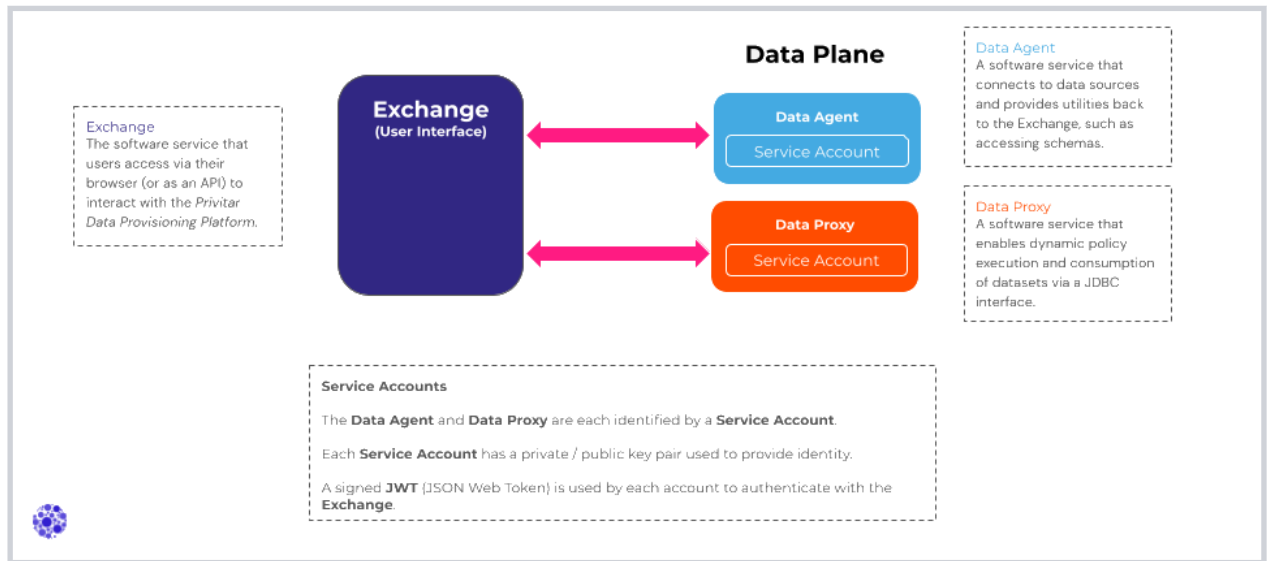
Within each enterprise, you can create multiple data exchanges, and within each data exchange, multiple data planes.

Initial login takes the enterprise admin to an enterprise console. User group management and registry setup happens here. You create and manage data exchanges here too.

Each data exchange is separate and different from other data exchanges, being a discrete entity within an enterprise.

No information is shared between exchanges. If you can access one data exchange it does not necessarily mean that you can access another.

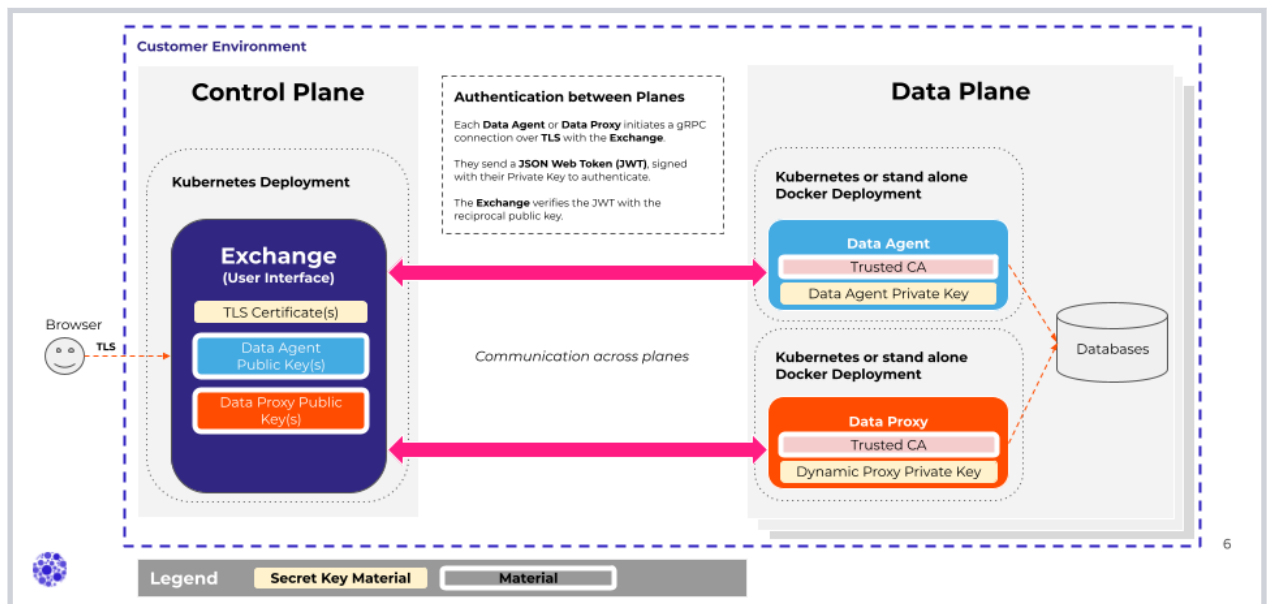
The following diagram summarizes the interaction between a data exchange and a data plane.



4.2. What Is the Control Plane?

The control plane is a logical perimeter that does not have direct access to data but may host components that drive operations in the data plane.

The following diagram shows, for the sake of simplicity, how the control plane interacts with a single data plane.



For the secure communication of services across the control plane and a data plane, refer to [cross-plane authentication](#).

The control plane is set, by default, to run in a [Kubernetes cluster](#), and is created with [sensible default configuration values](#). It contains a variety of services that are beyond the scope of this document, but the following are worthy of further explanation:

- **Data bridge**—This is the component in the control plane that handles communication with the data plane. It acts as a gRPC Remote Procedure Call ([gRPC](#)) server. It is replicated, and it sits behind an [ingress](#) with a load-balancer.

- **Data bridge load-balancer**—Because the data bridge is a server with multiple replicas (for redundancy and scalability), it lives behind a load-balancer to which clients connect. The platform does not interact with it directly.

Configuration Values

The control plane has the following configuration values:

- The data bridge requires the maximum JSON Web Token ([JWT](#)) expiration period for the client. The expiration period is typically only one minute, as it only needs to be long enough to allow token generation and token verification.
- The data bridge also requires the maximum amount of [clock skew](#) to tolerate.

4.3. What Is a Data Plane?

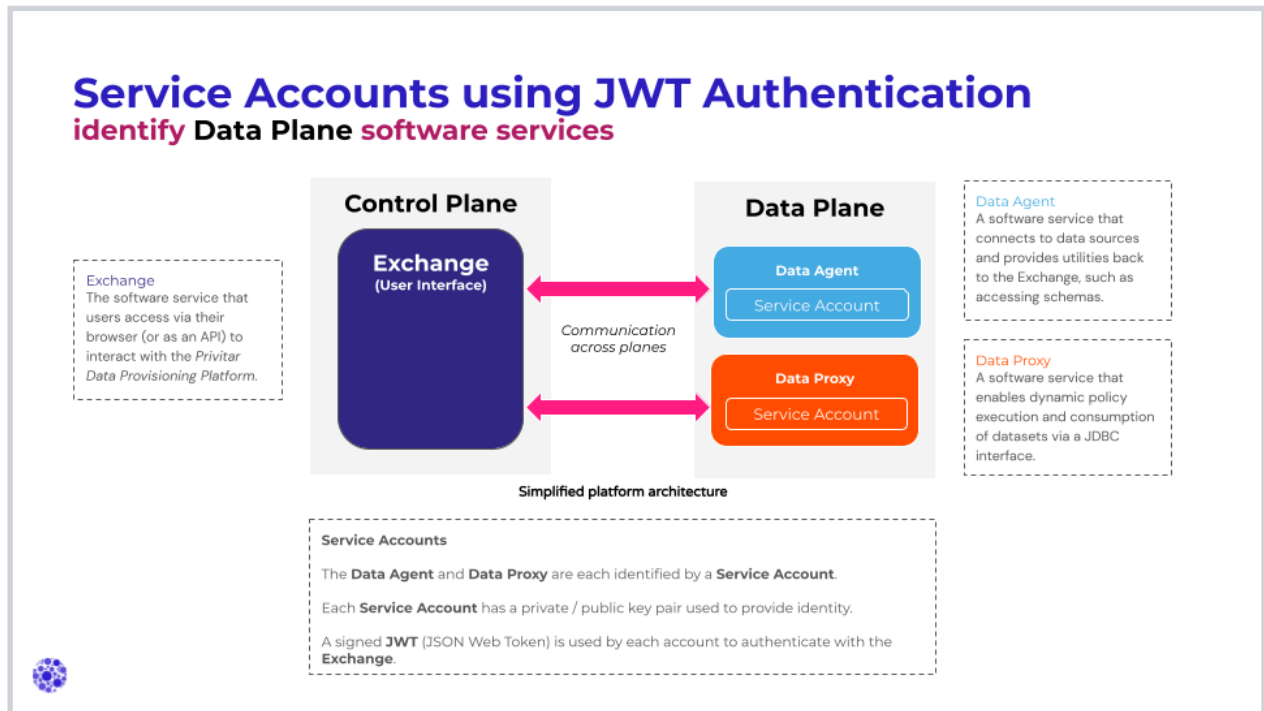
A data plane is a set of services used for the reading, writing, and processing of data. It contains a data agent and services capable of provisioning data, such as a data proxy or an integration using the Privitar SDK.

You create a data plane following the creation of a data exchange. See [Create and Edit a Data Plane](#).

A data plane has the following components:

- The data agent provides access to the data plane whenever required by the control plane, for example to retrieve the schema for a data asset. It makes a long-lived connection to the [data bridge](#) on startup.
- The data proxy is a Java Database Connectivity proxy ([JDBC proxy](#)) that allows data consumers to access sensitive data to which de-identification policies have been applied. It makes calls to the [data bridge](#) to fetch the information it needs, for example the details of how to connect to the sensitive data and the policies to be applied.

The data agent and data proxy components share logic for creating the gRPC Remote Procedure Call ([gRPC](#)) clients that call the [data bridge](#). Each data plane contains, at most, one (replicated) installation of each component.



Containerization

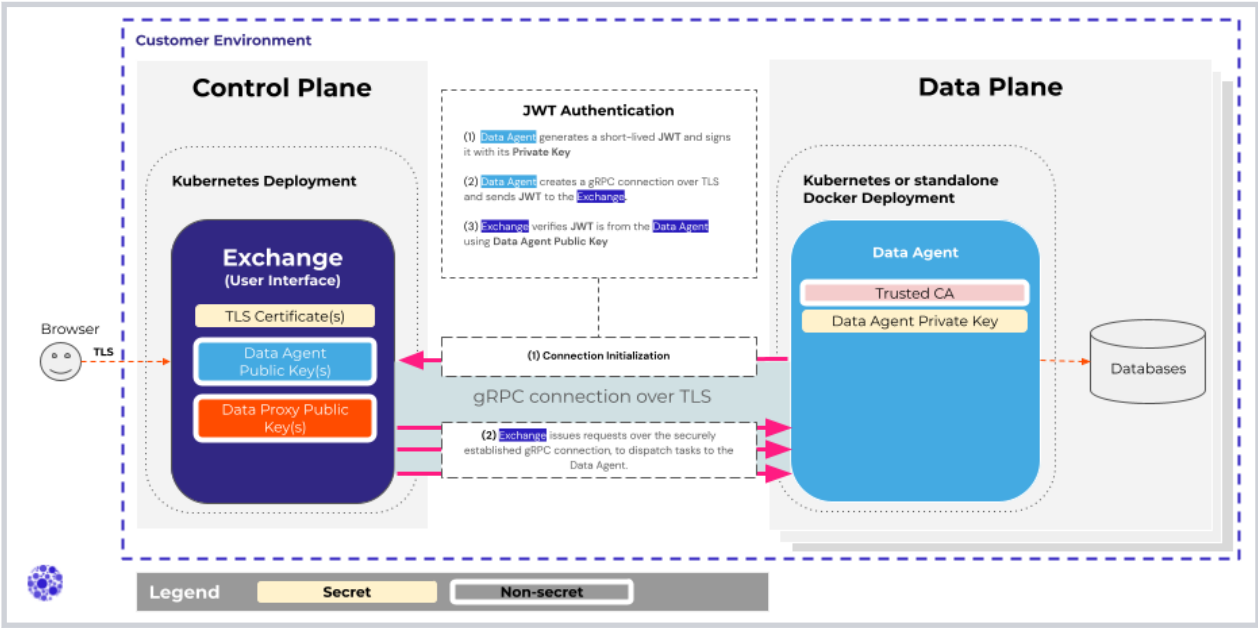
The platform bundles the data plane components in containers (by default [Kubernetes](#)).

You can deploy the data agent and data proxy to two separate Kubernetes clusters. However, you can install the required [JDBC drivers](#) in a single location.

4.4. Cross-Plane Authentication

The data plane initializes all connections with the control plane. To ensure secure communication with the data exchange, the platform:

- Implements service accounts using JSON Web Token ([JWT](#)) authentication
- Communicates using the gRPC Remote Procedure Call ([gRPC](#)) framework over Transport Layer Security ([TLS](#))



5. Enterprise Administration Tasks

An enterprise administrator is the first person to log in to a newly deployed Privitar Data Security Platform. Once logged in:

1. Configure user and group account management, including the registry settings.

If you intend to use the internal registry, there is no more user configuration required at the enterprise administration level, because the platform uses the internal registry by default. You may wish to continue working with the internal registry when you first explore the system, but it is more practical to use your existing user directory by configuring LDAP. See [LDAP Configuration](#).



Warning

If you use the internal registry and later enable LDAP, the existing registry is disabled. This removes all existing users and groups.

2. Create one or more data exchanges.

5.1. LDAP Configuration

Input your LDAP server's configuration values to configure LDAP on the platform. This essentially means configuring how your users authenticate to the platform and how and where user search happens in the LDAP directory.

1. Click **User Management** in the left navigation.
2. Click the **User Authorization & Authentication** tab.
3. On the LDAP tile, click **Start Configuration**.



Note

This button title changes to **Finish Configuration**, when LDAP configuration is still in a draft state (not enabled). When LDAP configuration is enabled, this button is re-labelled, **Update Configuration**.

The **Configure LDAP** page appears. It is divided into several sections, which have explanatory text that briefly outlines the information you need to supply. Complete all sections:

1. [LDAP Authentication](#)
2. [LDAP User Setting](#)
3. [LDAP Group Setting](#)
4. [User Attribute Mappings](#)
5. [Test LDAP Connection](#)
6. [LDAP Save and Enable](#)

5.1.1. LDAP Authentication

In this section, enter the credentials used by the platform itself to communicate with your LDAP server.

User Authentication

Configure LDAP

LDAP Authentication

You must enable password authentication on your LDAP server in order to connect it to the platform. Enter the URL of the LDAP server, the user distinguished name (UDN), and the password to authenticate with the LDAP server.

Connection URL

For example: ldap://172.31.27.208:1389

Bind Type

☐ None
☒ Simple

Bind DN

For example: cd=admin,dc=example,dc=org

Bind Credential

Enter Password

1. **Connection URL**—Enter the URL of your LDAP server.
2. Select the bind type. The platform supports simple bind authentication, which is also known as password-based authentication. Click **Bind** to access the user privileges set on your LDAP server. A simple bind will bind with a client's full name. All clients must be located in the same branch specified with the DN (distinguished name).

You should only select **None** if your LDAP server supports an anonymous bind. This option requires no further input because it does not require bind credentials.

3. Enter the user distinguished name in **Bind DN**. It is up to you to define what it is, but from the platform perspective, it is similar to an absolute path in a file system. It could be a synthetic ID (just a collection of numbers), but usually it's a comma-separated list of relative DNs, each DN being a node in the LDAP directory tree. An example would look like this: `cn=admin,cn=users,dc=privitar,dc=com`

Any fragment of the distinguished name, such as `cn=admin` or `dc=privitar`, is a relative distinguished name (RDN). If you use the file system analogy, each RDN represents one relative directory hop.

5.1.2. LDAP User Setting

In this section, enter information that tells the platform where to look for users in the LDAP server. The first three fields are LDAP user attributes that map the search to the correct attributes in the server.

Configure LDAP

LDAP User Setting

For a user-based search, enter the LDAP search base, the search filter, the name of the attribute, and the field name within that attribute that contains the user entry.

Username LDAP Attribute

Username

RDN LDAP Attribute

For example: uid

UUID LDAP Attribute

For example: entryUUID

User Object Classes

For example: inetOrgPerson, organizationalPerson

Users Search Base

For example: dc=example,dc=org

User Search Scope

☒ One Level
☐ Subtree

Username LDAP Attribute—Enter the attribute that the platform should treat as the username.

RDN LDAP Attribute—Enter the main identifier attribute, which is part of the DN and meant to uniquely identify the user with the given LDAP path (search base). It is typically uid, dn, or cn.

UUID LDAP Attribute—Enter the universally unique identifier of a user. This is a globally unique string of 16 octets (128-bit).

The **User Object Classes** used depends on the schema of your LDAP server. For example:

- posixAccount for an nis.schema
- inetOrgPerson for an inetorgperson.schema
- organizationalPerson for a core.schema

In **Users Search Base**, enter the location in the LDAP directory where the search for a particular directory object begins. It is denoted as the distinguished name of the search base directory object. For example: CN=Users,DC=domain,DC=com

Select **One Level** or **Subtree**. "One Level" means that all users must have the same path and they only differ in the RDN part of their DN. Conversely, "Subtree" means that the platform searches all sub-vertices of the base path too. For example, when the base path is cn=users,dc=example,dc=com then it also searches cn=externalUsers,cn=users,dc=example,dc=com.

5.1.3. LDAP Group Setting

In this section, enter the LDAP group information.

Configure LDAP

LDAP Group Setting

For a group-based search, enter the LDAP search base, the search filter, the name of the attribute, and the field name within that attribute that contains the group entry.

Group Search Base

For example: dn

Group Name LDAP Attribute

For example: cn

Admin Group Name

For example: admins

Group Object Classes

For example: GroupOfNames

User Groups Retrieve Strategy

☒ Member
☐ Member-Of

Membership LDAP Attribute

member

Membership Attribute Type

☒ DN
☐ UID

LDAP Filter

For example: (dn=*)

Group Search Base—Enter the location in the LDAP directory where the search for a particular group directory object begins. It is denoted as the distinguished name of the search base directory object. For example:

```
ou=Group,dc=HRES,dc=ADROOT,dc=yourCompany
```

Group Name LDAP Attribute—Enter the name of the group name attribute value as used on your LDAP server. While this is typically `cn`, it's important to consider other options when using subtree. In this case LDAP usually doesn't enforce the uniqueness. For example, `cn` is typically expected to be unique only within its own level, not necessarily the whole subtree. To avoid these collisions, it's best to use a globally unique `dn` or `uuid` in place of the usual `cn`.

Admin Group Name—Enter the name of the LDAP group to which the platform will automatically give the role of enterprise administrator. Members of this group can log in to the enterprise console to manage users and create exchanges.

Group Object Classes—Enter the name of the group object class as used on your LDAP server.

Select a **User Groups Retrieve Strategy** option:

- Use the **Member** attribute to find a group and search each member of the group
- Use the **Member-Of** attribute to find the group members and then the group that they are a member of

Membership LDAP Attribute—Enter the value of the attribute for group membership.

Select a **Membership Attribute Type** option:

- DN to use an absolute path
- UID to use a relative path

LDAP Filter—Enter an object value to filter the user search to a particular group. This could be a certain department or division within your organization. It could also be based on attributes such as disabled, enabled, or suspended.

5.1.4. User Attribute Mappings

The **User Attribute Mappings** are important because this maps what you see in the platform as First Name, Last Name, and Email Address to the specific LDAP properties that the LDAP registry defines. In each field, enter the LDAP attribute value as used in your LDAP server.

Configure LDAP

User Attribute Mappings

Enter the names of the LDAP attributes which will map to the user attributes in the platform.

First Name Attribute

For example: cn

Last Name Attribute

For example: sn

Email Attribute

For example: email

5.1.5. Test LDAP Connection

Test LDAP connection ensures that you don't lock yourself out of the platform. If you don't check before switching the identity provider, and there is a mistake in the URL, no one will be able to log in. Before you enable LDAP, or save LDAP settings, you must test the configuration first. The platform does not allow you to enable or save your LDAP configuration without the test, which checks whether:

- the URL is an actual reachable LDAP server
- the bind credentials allow the platform to log in there

Enter your administrator username and password, and click **Test Connection**. If the test is successful, you can enable the LDAP connections in the **LDAP Status** section.



Note

The platform does not currently check the admin password. Be sure you have this correct.

5.1.6. LDAP Save and Enable

In the **LDAP Status** section, you can choose to:

- Save your settings but leave LDAP disabled. You may want to do this if you are still entering your settings but wish to save a draft. Note that you must at least have your username and password correct, otherwise you cannot save the settings.
- Enable the connection to your LDAP server from the platform. This automatically disables the internal registry, as the two modes are mutually exclusive.

Click **Save** to save all LDAP server settings.

5.2. Create Your Initial Data Exchange

Enterprise administrators create data exchanges within the enterprise, which exchange administrators subsequently edit and manage.

1. Log in as enterprise administrator.
2. From the enterprise console, click **Create Exchange**.

The **Create a New Exchange** page appears.

Define the Exchange

Create A New Exchange

Name of Exchange
This is the name of your exchange

Add name

This field is required

Add description

0/250

Exchange URL
URL of the new exchange.

https://onpremeu.control-plane. -demo.privitar-internal.com/

Create

3. **Name of Exchange** —Enter a name for the new data exchange.

This automatically becomes the first part of the data exchange URL in the **Exchange URL** field.

If you prefer the name of the exchange to be logical and business friendly, edit the URL. For example, "The Acme Bank of North America and Canada" would give you "acme-bank-of-nort". You can change that to a simplified and supported URL, such as "acme-bank".

4. Add an optional description of the exchange in **Add Description**.
5. Click **Create** to create the new data exchange.

The **Data Exchanges** page appears showing the exchange you have created. When you return to create other exchanges, this page displays all previously created data exchanges.

6. Click **Open Data Exchange** (the box with the arrow).

The enterprise administrator who created the exchange is the default administrator for that data exchange. However, it is usually the exchange administrator who subsequently edits the data exchange to create data planes and policies, rules, and assets.

Click the top-right button, which displays either initials or an avatar, then click **View Exchange** to access the exchange admin console.

The **Overview** tab shows the date of creation and who created the exchange. It also shows which enterprise the data exchange is tethered to. Users can use the URL as the address to directly log in to that data exchange.

6. Exchange Administration Tasks

The enterprise administrator must complete initial enterprise administration before exchange administration takes place. An enterprise administrator invites a user to log in to the system as an exchange administrator.

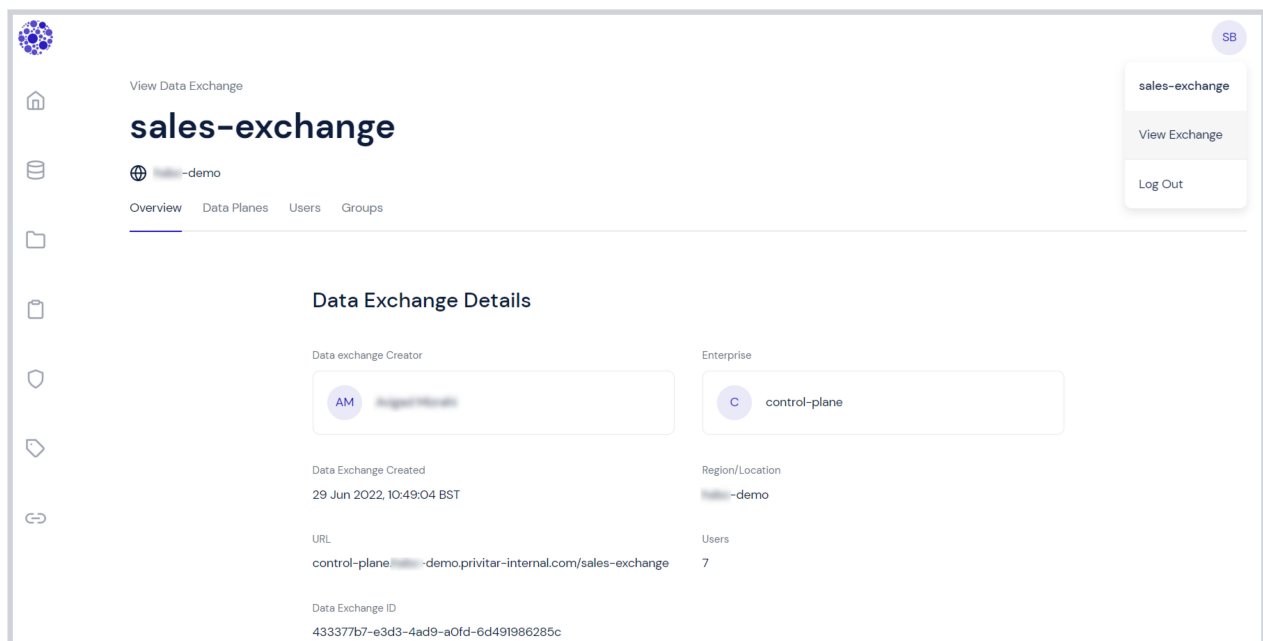
Your exchange admin login takes you straight to the exchange.

The following left navigation options are used by data guardians, data owners, and data consumers for further platform configuration and maintenance. They are described in the DSP User Guide:

- Policies, rules, and transformations
- Projects
- Business information (tags, data classes, terms)
- Connections
- Tasks

Exchange administration focuses on the activities available in each tab of **View Exchange**.

Click the avatar symbol in the top right corner of the page, and select **View Data Exchange**. This takes you to an exchange admin console.



During configuration, the exchange administrator:

1. Creates a data plane, including:
 - a. Create public keys
 - b. Configure the data proxy host
 - c. Deploy the data agent and the data proxy
2. Manages users and groups, including:

- a. Add and delete users, and assign user roles
- b. Add and delete groups, and assign group roles

After initial platform configuration, the exchange administrator continues to update and manage the data plane, users, and groups.

If you click the data exchange symbol, you can also add and configure datasets. The DSP User Guide covers this data owner work.

Prerequisites

This guide assumes that the data plane is deployed in the same Kubernetes cluster as the control plane. This is not a specific requirement. The data plane can be deployed (and is intended to be deployed) close to the data itself. This may mean that the data plane lives in a separate Kubernetes cluster.

Make sure you are logged in to your Kubernetes cluster and your container repository. Installation is the same for all cloud environments except for one step in which you create a volume with a JDBC driver. You may need to seek guidance on how to connect to Amazon Elastic Kubernetes Service (EKS) and Microsoft Azure Kubernetes Services (AKS) from colleagues involved in IT infrastructure installation.

6.1. Create a PKI Infrastructure

The following happens outside the platform's user interface and it will most probably be done by the likes of a system administrator or infrastructure engineer.

Both the data agent and the data proxy require a key infrastructure to authenticate with the control plane. The platform uses a private/public key pair mechanism for authentication, in the same way you would authenticate to, for example, GitHub.

Before deployment, you must generate a private/public key pair for each service. You then configure the data plane components with the private key and inform the control plane of the public key.

First, create a public/private key pair for the data agent:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out ./rsa-private.pem openssl rsa -pubout -in ./rsa-private.pem -out ./rsa-public.pem
```

In this guide, we use the same private/public key pair for both the data agent and the data proxy. In production, use separate key pairs.

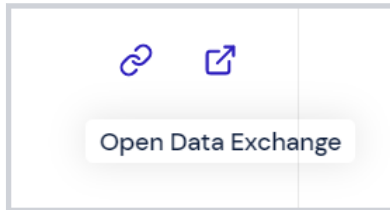
You will also need a copy of the public certificate used by the control plane. Use the following command to grab the certificate file from your control plane:

```
kubectl get secret dpp-control-plane-tls-ingress-cert -n istio-system -o jsonpath='{.data.tls\.cert}' | base64 --decode > ./control-plane.crt  
kubectl get secret dpp-control-plane-tls-ingress-cert -n istio-system -o jsonpath='{.data.tls\.cert}' | base64 --decode &gt; ./control-plane.crt
```

6.2. Create and Edit a Data Plane

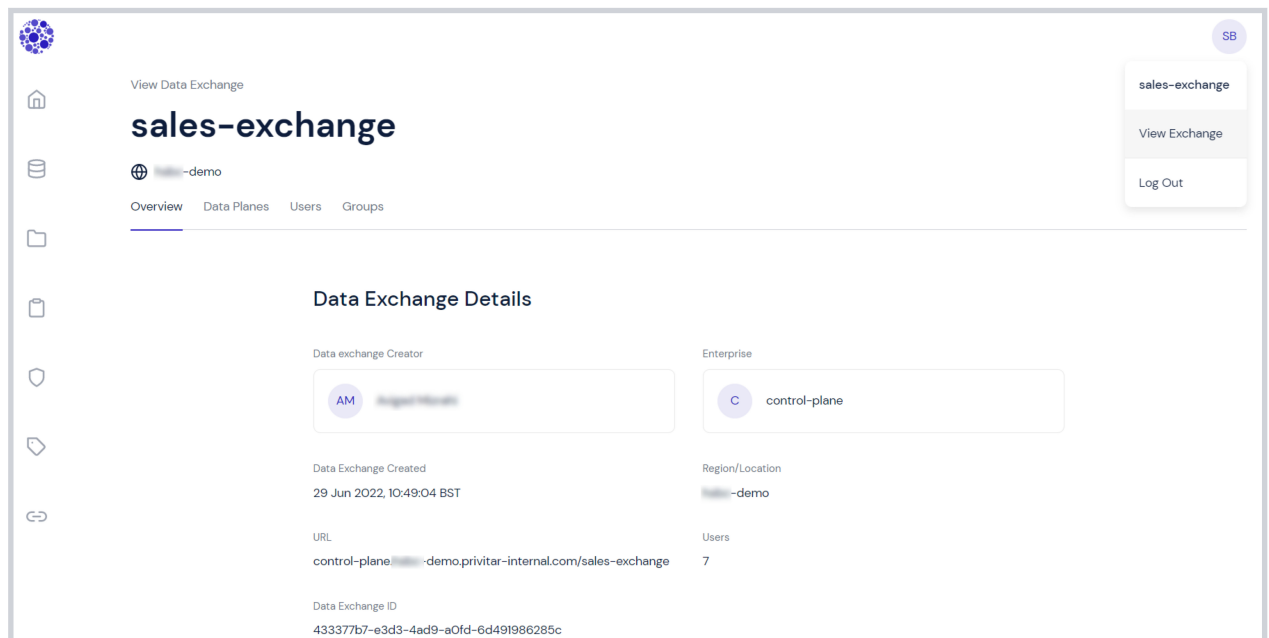
The first thing to do as an exchange administrator is edit the data exchange to create the data plane.

1. Click **Data Exchanges** in the left navigation, and select **Open Data Exchange**.



2. Click the avatar symbol in the top right corner of the page and select **Manage Exchange**.

The **View Data Exchange** page appears in a new window.



3. Click the **Data Planes** tab.

If there are no data planes, a message confirms that there are **No Data Planes in this Data Exchange**.



Note

At a later date, when data planes already exist, they are listed on this page and can be edited further. For example, you can add data agent public keys or a data proxy.

4. Click **Create** (the plus sign).

The **Create Data Plane** tab opens. This is the start of the data plane creation wizard. It is also possible to create a data plane with a script, but that is outside the scope of this document.

The screenshot shows a 'Data Plane' wizard window titled 'Create Data Plane'. It has a close button (X) in the top right corner. The main content area is divided into two sections: 'Title and Description' and 'Location'. The 'Title and Description' section includes a text input field for 'Title' and a larger text area for 'Description' with a character count '0/250' and a small edit icon. The 'Location' section includes a dropdown menu labeled 'Select Location'. At the bottom, there is a navigation bar with a 'Back' button, a progress indicator showing three steps: '1 Create' (active), '2 Configure', and '3 Deploy', and a 'Next' button.

Data Plane

Create Data Plane

Title and Description

Add a name for your data plane and a description to help identify it, such as the environment where it is deployed.

Title

Description

0/250

Location

Select the country where the data plane is physically deployed.

Select Location

Back

1 Create 2 Configure 3 Deploy

Next

As you progress through the wizard, click **Back**, if required, to return to previous steps.

5. **Title**—Enter a name for the data plane. The data plane description is optional.
6. Choose a location from **Select Location**.
7. Click **Next**.


Configure the data agent public keys (next step) or click **Skip and Save** if you want to do it later.

8. Click **Add a Public Key**.


The **Add a Public Key** dialog appears.

Data Plane

Configure Your Services

 I will generate my data agent keys later. [Skip](#)

Data Agent Public Keys



Add a Public Key

You must generate a public/private key pair to use with a deployed data agent. Click **Add a Public Key** and enter the public key. Retain the private key for deploying the data agent. You must generate key pairs as RSA (using at least 2048-bit) or EC using a P-256 curve.

[Add a Public Key](#)

[Back](#) ✓ Create **2 Configure** 3 Deploy [Next](#)

9. In the **Name** field, enter a name for the data agent key.
10. In the **Key** field, paste in a valid X.509 Subject Public Key Info (SPKI), PEM-encoded public key. It will start with -----BEGIN PUBLIC KEY and end with -----END PUBLIC KEY-----

You generate a key in a separate PKI creation procedure outside the the platform's user interface.

11. Click **Save**.

The **Configure Your Services** page appears. The newly created data agent public key appears.


Data Plane

Configure Your Services

Data Agent Public Keys

[Add a Public Key](#)

Available keys



LargoFLA

-----BEGIN PUBLIC KEY----- MIIBjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmDla4gwgOirMFIBF4Hnw 9kD4oUuxPOPNAIT/3i4ztODg3I:

[Back](#) ✓ Create **2 Configure** 3 Deploy [Next](#)

Note that the wizard is still in the configure stage as shown in the steps at the bottom of the page.

12. Add another public key or click **Next** to continue with the data plane wizard.

The **Data Proxy Host Details** page appears.

13. Click **Skip** to generate a key later or click **Add a Public Key**.

The **Add a Public Key** dialog appears.

14. In the **Name** field, enter a name for the data proxy public key.
15. In the **Key** field, paste in a valid X009 Subject Public Key Info (SPKI), PEM-encoded public key. It will start with `- - - - - BEGIN PUBLIC KEY` and end with `END PUBLIC KEY- - - - -`.

You generate a key in a separate PKI creation procedure outside the the platform's user interface.

16. Click **Save**.

The **Configure Your Services** page appears. The newly created data proxy public key appears.

17. Add another public key or click **Next** to continue with the data plane wizard.

The **Data Proxy Host Details** page appears. Click **Skip and Save** if you wish to finish configuration at this point. You still need to configure the proxy host later.

18. In the **Host** field, enter the IP or hostname of the data proxy host.

Use this hostname in the YAML for the deployment of the data plane components. Typically, this hostname should be a subdomain of the domain value you specified for the control plane. For example, `proxy.privitar.services`, if the control plane is deployed to `dsp.privitar.services`.

19. In the **Port** field, enter the port number to which the JDBC client will connect.

Select a port number, such as 9000.

20. Click **Save**.

The **Deployment Configuration** page appears.

From here you can click **Download** to download the data agent configuration and the data proxy configuration. The two configuration files make up the data plane, which is [deployed in Kubernetes in a separate procedure](#).

You will need these values to configure the YAML for data plane [deployment](#).

21. Click **OK** to finish the configuration.

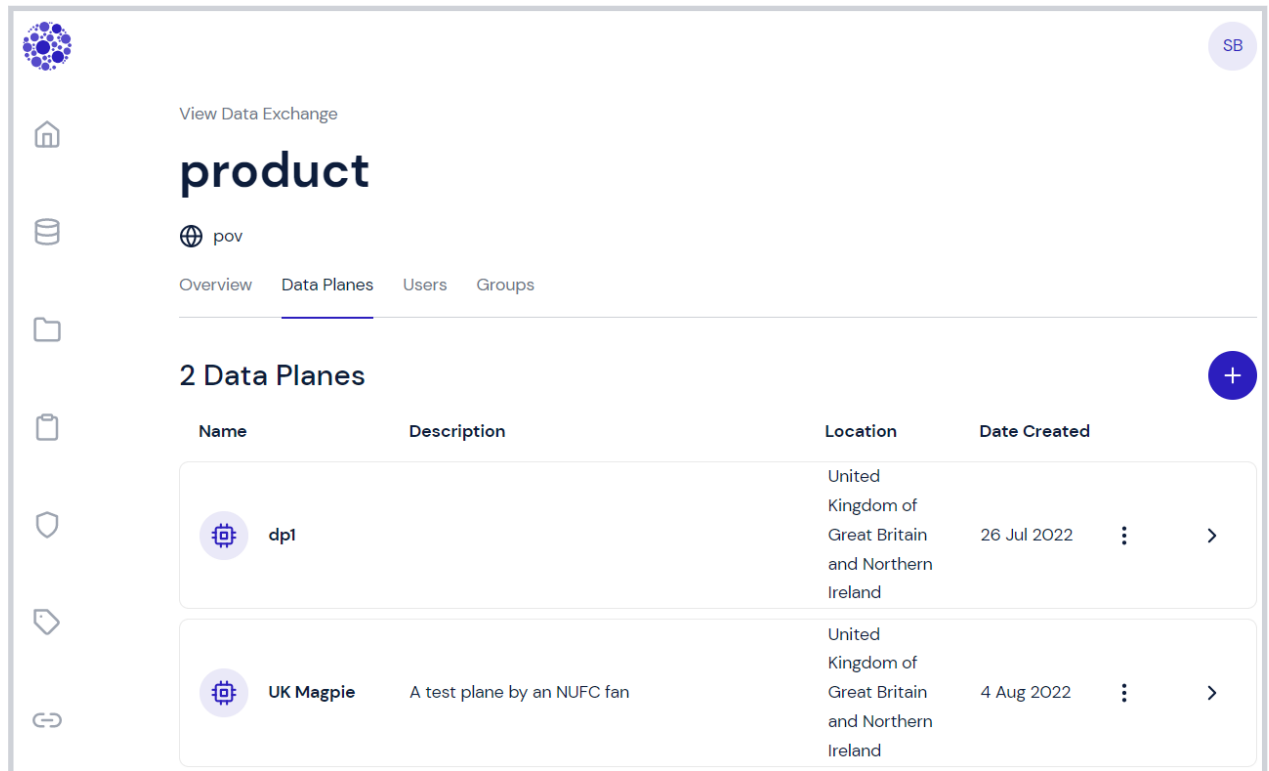
You now have the data plane services required to talk to the data source, so that actions such as data provisioning and data registration can take place. They are services required in addition to what was initially deployed.

Edit an Existing Data Plane

Re-edit a data plane configuration to, for example, change a proxy host port number.

1. From the **View Exchange** page, click the **Data Planes** tab.

A list of data planes appears.



2. Click the data plane you wish to edit.

The **Data Planes** page appears.

3. Click the **Data Agent** tab or **Data Proxy** tab to edit the configuration details.

The configuration steps within each dialog are the same as described above during initial data agent and data proxy configuration. Once you have changed the artifacts they can be downloaded again for [deployment](#).

6.3. Deploy the Data Plane Artifacts

This section is an overview of data plane deployment. There are slight differences in how this happens, depending on the deployment environment. For example, on premises or on a cloud such as Microsoft Azure or Google Cloud Platform (GCP).

Installation is the same for all cloud environments except for one step in which you create a volume with a JDBC driver. You deploy a data plane as a system administrator at the command line.

Once you have registered a new data plane in the control plane, you must deploy the data plane configuration to your Kubernetes cluster.

In the `dpp-data-plane-[version]` folder extracted from the release bundle tarball, edit the `data-plane-values.yaml` configuration file. Specify the following values:

- Set `clean` to `false`.
- Set `externalHostname` to [the hostname you configured](#) for the data proxy in the control plane.
- Set `controlPlane → hostName` to the full hostname of the control plane you deployed previously.

- Set `controlPlane → dataAgentId` to the `dataPlaneId` in the data agent configuration file you downloaded from the control plane.
- Set `controlPlane → exchangeId` to the `exchangeId` in the data agent configuration file you downloaded from the control plane.
- Set `controlPlane → certificateFile` to the full path to the control plane public certificate you downloaded when creating the [public key infrastructure](#).
- Set `images → load` to `true`.
- Set `images → push` to `true`.
- Set `images → targetRegistry` to your container registry path. This setting will depend on the various flavours of Kubernetes (and cloud providers).
- Set `dataPlaneConfigClaimname` to the name of the `PersistentVolumeClaim` you configured as part of the [prerequisites](#).
- Set `clientAuth → dataAgentPrivateKeyFile` to the full path of the private key for [the data agent you created in the data plane](#).
- Set `clientAuth → dynamicProxyPrivateKeyFile` to the full path of the private key for [the data proxy you created in the data plane](#).

Once this configuration is complete, you can deploy the components:

```
./dppctl install -f ./data-plane-values.yaml
```

Once deployed, you must configure the Kubernetes gateway to allow communication with your data agent. This is simply a guide to how that command will look:

```
kubectl patch gateway data-plane-gateway -n [data plane namespace]
--type=merge -p '{"spec":{"selector":{"istio":{"istio-ingress-[data plane namespace]"}}}}'
```

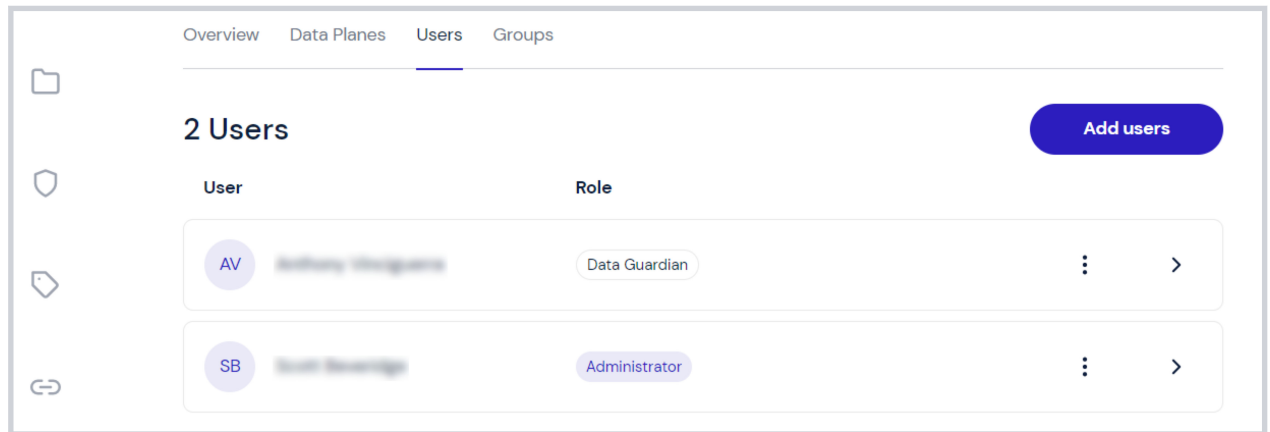
6.4. Managing Users and Groups

1. Click **Data Exchanges** in the left navigation, and select **Open Data Exchange**.
The **View Data Exchange** page appears in a new window.
2. Click the avatar symbol in the top right corner of the page and select **Manage Exchange**.

An overview of the exchange appears. There are also tabs in which you can configure data planes, users, user groups, and more.

6.4.1. Add Users and Assign Roles

1. From the **View Exchange** page click the **Users** tab. The **Users** page appears.



2. Click **Add users**.

The **Add Users to Data Exchange** page appears.

3. Click **Select Users** and search for the user. Search is case sensitive. Select the user you wish to add.

The platform will search for users in either the internal registry or LDAP. This depends on the user and group registry settings made during [enterprise administration](#).

Once you have added your users, you can either:

- Click **+Role** to assign roles individually to each user
- Click **Assign Roles to All Users** to assign roles to all the users in the current list

For individual role assignment, click **+Role** and select a role for that user from the list. Add as many of the available roles as you want for each user.

In **Assign Roles to All Users**, select roles. You can opt to override the users' original roles by selecting the override box.



Note

When viewing the list of users who may access the enterprise or exchange, the list only displays those users that have been directly added. Users who are members of a group are not shown. Group membership and user information may be viewed from LDAP.

Click **Save** to assign the roles.

Click **Save** to save the added users and role settings.

To return to viewing users and updating roles, click the **Users** tab.

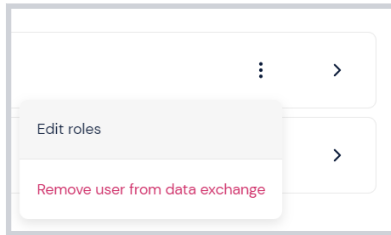
Edit User Roles

To edit user roles:

1. Click the **User** tab in **View Data Exchange**.

2. Click More (the three vertical dots) and select **Edit Roles**.

A box appears on the right of the screen. It shows existing roles



3. Click the **X** next to a role to remove that role, or select new roles from the menu.
4. Click **Save**.

Remove a User from the Data Exchange

To remove a user from the exchange:

1. Click the **User** tab in **View Data Exchange**.
2. Click More (the three vertical dots) and select **Remove user from data exchange**.

A message states, **This will remove all permissions for the user, and they will lose access to the platform unless they belong to a user group which is allowed access**. This means the user could still have rights on the exchange if they are part of a group.

3. Check the box to confirm that you understand and would like to proceed.
4. Click **Remove**.

The user is removed and the list of existing users appears.

6.4.2. Add Groups and Assign Roles

To add groups and assign roles to groups:

1. Click the **Groups** tab.
2. Click **Add Groups** to search groups in the registry.

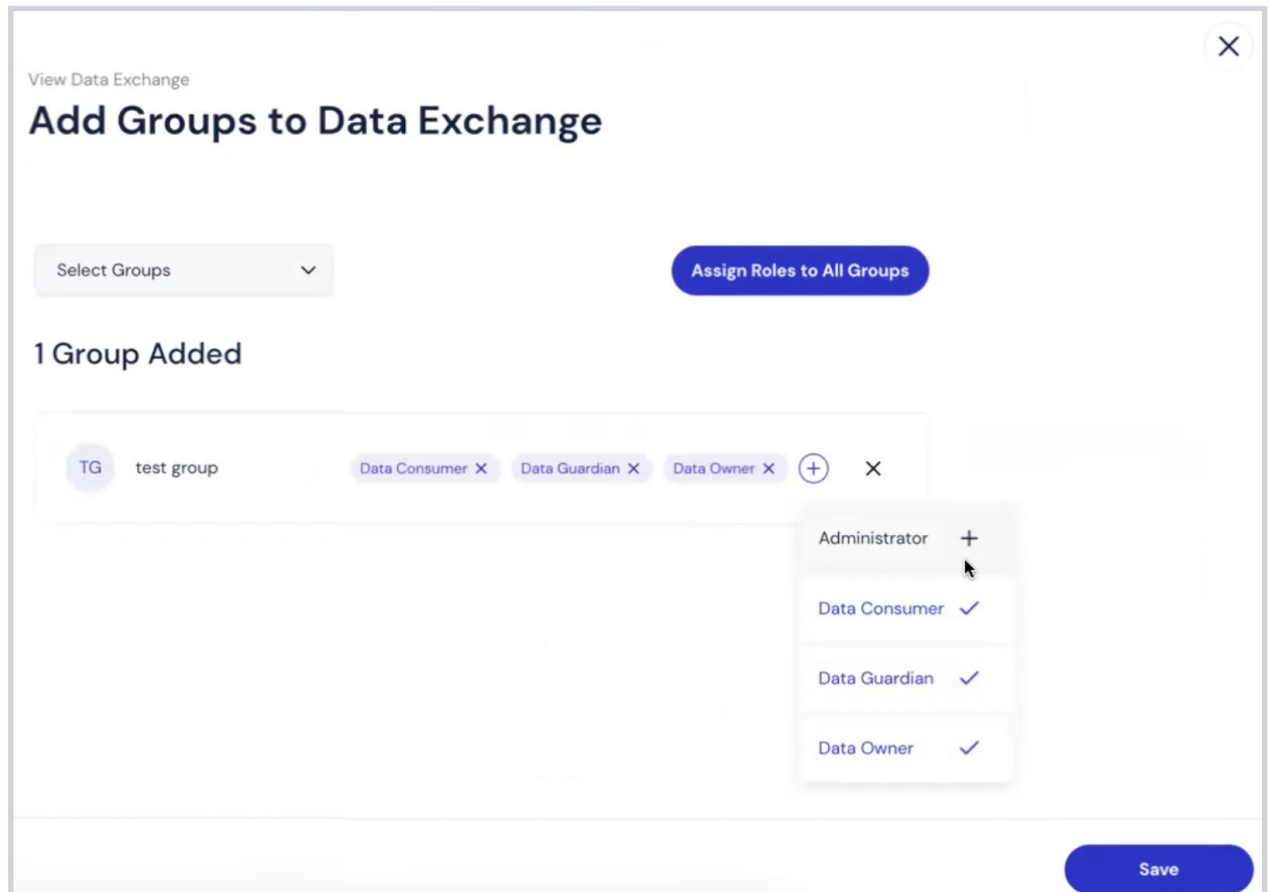
The platform will search for the group in either the internal registry or LDAP. This depends on the user and group registry settings made during [enterprise administration](#).

3. Click **Select Groups**, and search for the group.

Search is case sensitive. Select the group you wish to add. Once you have added your groups you can either:

- Click **+Role** to assign roles individually to each group
- Click **Assign Roles to All Groups** to assign roles to all the users in the current list

To assign a role to a single group, click **+Role** and select a role for that group from the list. Add as many of the available roles as you want for each group.



4. In **Assign Roles to All Groups** select roles.
Check the **Override** box if you want to override the groups' original roles.
5. Click **Save** to assign the roles.
6. Click **Save** to save the added groups and role settings.



Note

Assigning a role to a group enables that role for all members of the group. Users may be members of multiple groups, so roles could become compounded if you do not manage users and groups regularly.

Edit Group Roles

To edit group roles:

1. Click the **Groups** tab in **View Data Exchange**.
2. Click More (the three vertical dots) and select **Edit Roles**.
A box appears on the right of the screen. It shows existing roles.
3. Click the **X** next to a role to remove that role. Select new roles from the menu.
4. Click **Save**.

Remove a Group from the Data Exchange

To remove a group from the data exchange:

1. Click the **Groups** tab in **View Data Exchange**.
2. Click More (the three vertical dots) and select **Remove Group from Data Exchange**.
Members of the group will lose the group permission but retain any permissions individually assigned to their user profile.
3. Check the box to confirm that you understand and would like to proceed.
4. Click **Remove**.

The group is removed and the list of existing groups appears.

6.5. Manage Policy Settings

6.5.1. Change Default Access Control Policy Behavior

As an exchange administrator, you can set whether access control policies allow or deny access to data by default.



Important

Upon deployment of the platform, we recommend that you immediately configure the behavior of access control rules to allow or deny access. If however, you wish to change this behavior at a later stage, you must delete all access control rules prior to making this change.

1. Click **Data Exchanges** in the left navigation, and select **Open Data Exchange**.
The **View Data Exchange** page appears in a new window.
2. Click the avatar symbol in the top right corner of the page and select **Manage Exchange**.
3. Click the **Policy Settings** tab.
4. Select whether access control policies allow or deny access to data by default.

Allow access to data—If you select this option, users will see “Allow Access to Records” as they create access control rules.

Deny access to data—If you select this option, users will see “Deny Access to Records” as they create access control rules.

6.6. Allow External Configuration of Correlation ID Queries

In order to set up the ability to pass external correlation IDs to queries, you need to configure both the data proxy client driver and the data proxy server.

The external correlation ID passed through the data proxy client driver will replace the randomly generated correlation IDs that are associated with connection events and query events.

6.6.1. Configure the Data Proxy

You can perform this configuration if all of the following are true:

- You are an exchange administrator on the platform.
- You have access to the Kubernetes namespace where the data proxy pods are running.
- You have permissions to change the configmap.

Configure the data proxy as follows:

1. Add the property `privitar.jdbcproxy.allowCorrelationId=true` to the `application.properties` section of the configuration `dynamic-proxy` configmap.
2. Enter the following command:

```
kubectl edit configmap dynamic-proxy-config -o yaml -n {namespace}
```

Replace `namespace` with the namespace where the data proxy pods are deployed.

3. Save the configuration.
4. Restart the pods.

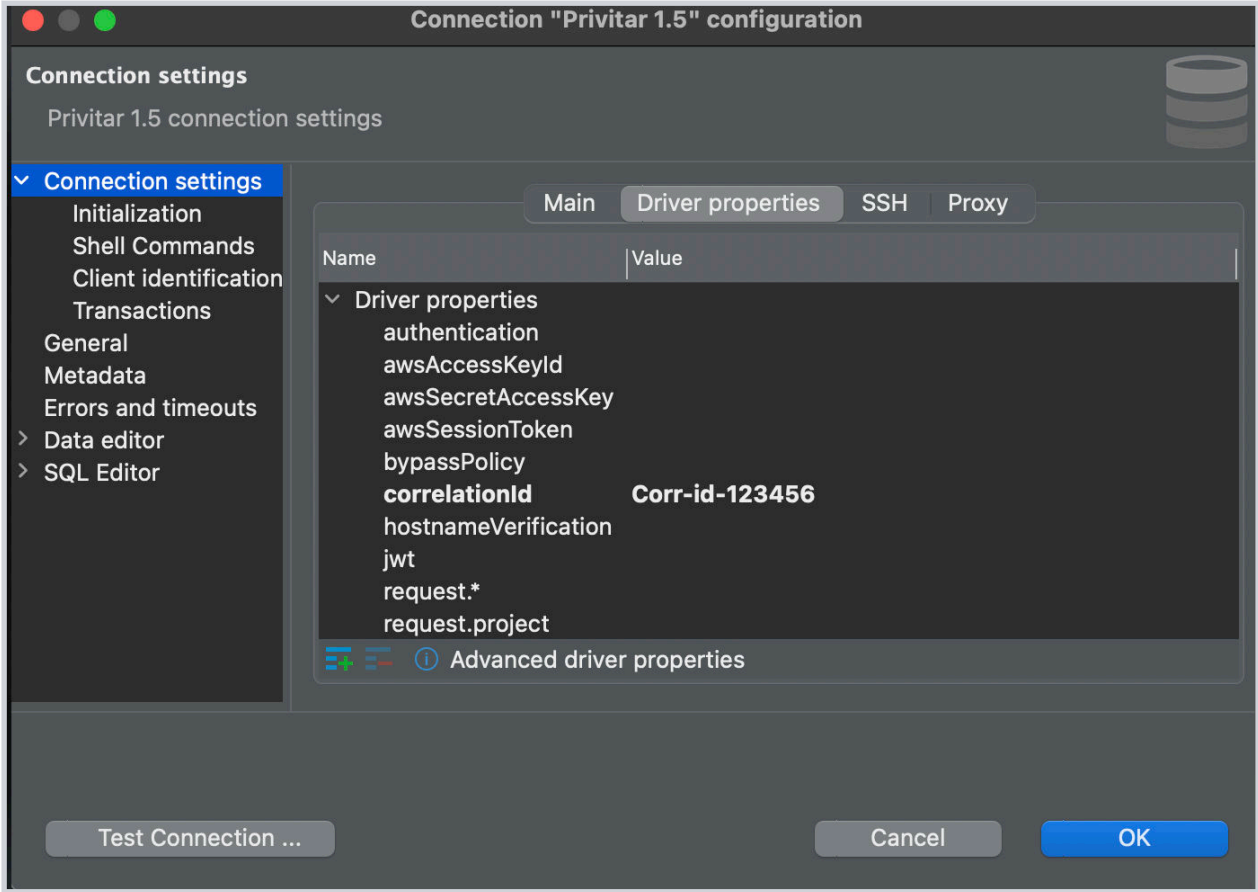
The pods are now configured to accept external correlation IDs.

The configuration will look similar to the following example:

```
apiVersion: v1
data:
  application.properties: |
    privitar.jdbcproxy.controlplane.target=controlplane.example.com:443
    privitar.controlplane.client.auth.mode=JWT_BEARER_TOKEN
    privitar.jdbcproxy.allow.context.header.auth=false
    privitar.jdbcproxy.authentication.method=INTERNAL
    privitar.jdbcproxy.bypasspolicyallowlist=8951fc29-4607-43e2-998a-
d663e0a5b4ec,e45992b2-1311-4d4f-be9a-9ac101d62bc8
    privitar.jdbcproxy.bypasspolicymode=RESTRICTIVE
    privitar.jdbcproxy.controlplane.client.auth.jwt.signing.key.path=/config/
dynamic-proxy/clientauth/jwt-signing-key.pem
    privitar.jdbcproxy.dataplane.id=49clf6d5-6446-46ef-ba94-e9624a13cbf4
    privitar.jdbcproxy.defaultFetchSize=100
    privitar.jdbcproxy.exchange.id=1c2235c0-240e-4f9d-9403-97f512512003
    privitar.jdbcproxy.healthcheckserver.port=8683
    privitar.jdbcproxy.healthcheckserver.timeout.seconds=5
    privitar.jdbcproxy.metricsserver.port=9095
    privitar.jdbcproxy.policy.cache.expirationInSeconds=10
    privitar.jdbcproxy.queryengine.transparentMode=false
    privitar.jdbcproxy.server.port=51320
    privitar.jdbcproxy.server.tls.enabled=false
    privitar.jdbcproxy.secretmanager=false
    privitar.jdbcproxy.allowCorrelationId=true
  auditServiceUrl: http://fluent-bit:8888
....
```

6.6.2. Configure the Data Proxy Driver

Once you have configured the data proxy server to accept external correlation IDs, you can now use the JDBC driver property `correlationId` to pass a custom correlation ID to be used in the logs for connection events and query events.



Notice

This property is only available on versions 1.5+ of the data proxy client driver. Please ensure that you're using the latest version of the driver.

Table 23. Expected Behavior with Correlation IDs

Driver	Data Proxy set with <code>privitar.jdbcproxy.allowCorrelationId=true</code>	Data Proxy set with <code>privitar.jdbcproxy.allowCorrelationId=false</code>
JDBC Driver property <code>correlationId</code> unset, set to empty string, or to one or more spaces	Correlation ID in audit logs for connection events and query events is randomly generated by the driver .	Correlation ID in audit logs for connection events and query events is randomly generated by the driver .

Driver	Data Proxy set with	Da
JDBC Driver property correlationId={string}	privitar.jdbcproxy.allowCorrelationId=true Correlation ID in audit logs for connection events and query events is set to the {string}.	privitar.jdbcpro This error message a AvaticaRuntimeExce configured to accep 'Proxy is not configur privitar.jdbcpro be set to true or set property on the driv '0000', Severity: 'ERR allowCorrelation empty correlationID code: '400', SQL Stat 500 (00000) ERROR

7. System Administration Tasks

This section covers administrative tasks which are largely done outside the platform's user interface, usually by a system administrator, or somebody in a similar role. Most of the activity described here happens at the command line or through an integrated development environment (IDE), such as IntelliJ IDEA or Microsoft Visual Studio.

7.1. Backup, Restore, and Business Continuity

This section is an overview of the Privitar Data Security Platform component characteristics, and their behavior in the event of failovers. It covers platform resiliency, availability, and backup and recovery.

This guidance assumes at least a medium-sized deployment, that is one that deploys components with replicas of individual Kubernetes pods. (Smaller deployments deploy the platform without replicas.)

7.1.1. Stateless and Stateful Components

The following table characterizes components by statefulness, how resilience and redundancy are achieved, and the behavior of the platform (including side effects) should an individual replica fail.

Table 24. Component Behavior During Failover

Component	Statefulness	Behavior and Side Effects on Failover
<ul style="list-style-type: none"> Enterprise Management API GraphQL Gateway Data Exchange API Data Bridge Data Agent Keycloak 	Stateless with replicas, but with impact to the requests.	If a component fails during a request, a request failure error appears in the user interface. The user should then re-try request submission.
Enterprise Management UI Data Exchange UI	Stateless with replicas.	<p>If a component goes down, the load balancer should re-apportion traffic to the replica pod. There will be no significant impact on the user.</p> <p>If the user sees an error in the control plane user interface, they can refresh to retry. The cookie state is not connected to these components, so no session or cookie information is lost on failover.</p>

Component	Statefulness	Behavior and Side Effects on Failover
<p>Your tools (external to the platform):</p> <ul style="list-style-type: none"> • Secret Store/Key Management Service (KMS) 	External tooling.	<p>Consider deploying a KMS in a highly available manner, so that individual “nodes” are replaced upon failover.</p> <p>Key material is cached in the data proxy for a specific time to live (TTL). If the key management service (KMS) was to become unavailable, some queries involving consistent tokenization may fail if the relevant key material hasn’t been cached.</p>
<ul style="list-style-type: none"> • Redis in-memory database storage. Stores session identifiers and JWT tokens • RabbitMQ message broker: request queueing handling 	Stateless storage.	<p>Deployed in a high availability setup (master/slave).</p> <p>Redis</p> <ul style="list-style-type: none"> • If any Redis nodes go down, the high availability (HA) setup takes over. • If all Redis nodes fail, a replacement pod will spin up automatically in its place. Login may be disrupted during this deployment phase. After re-deployment, users are re-directed to the login screen. <p>RabbitMQ</p> <ul style="list-style-type: none"> • If a single RabbitMQ pod fails, the HA setup takes over. • If all RabbitMQ pods fail, a replacement pod spins up automatically in its place. Transient requests in the queue at the time of failure fail (user will receive an error in the UI) and the user will have to retry.

Component	Statefulness	Behavior and Side Effects on Failover
<ul style="list-style-type: none"> • PostgreSQL for Keycloak • PostgreSQL for Flowable 	Persistent, stateful storage.	<p>PostgreSQL for Keycloak:</p> <ul style="list-style-type: none"> • PostgreSQL databases are deployed with replicas. • If the PostgreSQL pod dies, a replacement pod spins up automatically in its place. • If backups of the underlying storage were taken from the persistent storage, they can be restored. Without backups, the LDAP configuration in the database will need to be re-constituted to continue using LDAP for authentication. <p>PostgreSQL for Flowable:</p> <ul style="list-style-type: none"> • PostgreSQL databases are deployed with replicas. • There is also an option to configure Flowable to use an external PostgreSQL database. • If the pod dies, a replacement pod spins up automatically in its place. While this happens, some user interface screens may display errors. • This database stores in-flight workflow elements, such as requests in draft. Approved and completed requests are persisted to the graph database (in Cassandra). In the event of a complete loss of the underlying data storage, in-flight requests are lost and must be resubmitted.

Component	Statefulness	Behavior and Side Effects on Failover
<ul style="list-style-type: none"> • Apache Cassandra (NoSQL database) • Apache Solr (search platform) • Apache ZooKeeper (server) 	Persistent, stateful storage.	<p>Graph database, comprised of:</p> <p>Apache Cassandra (NoSQL database):</p> <ul style="list-style-type: none"> • Deployed in high availability mode, therefore, resilient to individual pod failures. • Cassandra works on a quorum system. This means two out of three pods must be running for Cassandra to function. • If all Cassandra pods are lost, most workflows in the user interface will have errors. • When Cassandra is restarting, unpredictable user interface failures can happen. Users should refresh their page. <p>Apache Solr (search platform)</p> <ul style="list-style-type: none"> • JanusGraph uses Apache Solr as an index backend. • Solr leverages ZooKeeper to coordinate collection and configset information between the Solr servers. <p>Apache ZooKeeper (server)</p> <ul style="list-style-type: none"> • Using ZooKeeper with Solr reduces the amount of manual configuration required to use Solr as a backend index for JanusGraph.

7.1.2. Data Proxy – Control Plane Communication Behavior

A query client connects to the data proxy using JDBC, a stateful protocol. The data proxy communicates with the data bridge in the control plane at the time of establishing the first query client connection in a session. Each session is assigned to a single data proxy pod.

If the data proxy pod assigned to a session fails in the middle of a query, the query will fail and the session connection fails. The query client must reestablish the connection and resubmit the query.

Some client tools may have this retry logic built in, so the impact on the platform user may be minimal.

7.1.3. Backup and Restore

You should formulate, document, and test a full platform backup and restore process. A simple, high-level backup and restore process might look like the following:

1. Deny access to the system during a maintenance period.
For example, disable inbound network traffic to the control plane by shutting down the Kubernetes pods that provide UI and API services.
2. [Back up each data store with an external tool](#) (for example, [Velero](#)), or use the data storage native backup system.

3. Block platform access, and [restore data storage from backups](#).
4. Restore access to the platform.
5. [Validate the restoration](#).

Back Up the Platform

To back up the platform, we recommend using a third-party tool designed for this purpose. These instructions use [Velero](#) as an example of such a tool.

1. Perform a one-time installation of the third-party tool (such as Velero):
 - a. Install the [Velero command line interface \(CLI\)](#).
 - b. Follow the instructions to install Velero in your cloud provider's container:
 - [Amazon Web Services \(AWS\)](#)
 - [Google Cloud Platform \(GCP\)](#)
 - [Microsoft Azure](#)
2. In the backup tool, create a backup of the platform:
 - a. Run the following command to create a backup of the platform:


```
velero backup create <backup_name> --include-namespaces <namespace_name>
```
 - b. Describe the backup (this also outputs the status of the backup):


```
velero backup describe <backup_name>
```
 - c. Verify that the backup successfully completed with no errors or warnings.
If any errors or warnings appear, check logs with the following command:


```
velero backup <backup_name>
```

Restore the Platform

To restore a backup of the the platform:

1. Delete the existing namespace:


```
kubectl delete namespace <namespace_name>
```



Warning

If you skip this step and attempt to restore to a new namespace while the original still exists, there may be several clashes, such as with hostnames.

2. Restore the namespace (this outputs the name of the restore: <restore_name>):


```
velero restore create --from-backup <backup_name>
```
3. Describe the restoration (this outputs the restoration status) and check it has succeeded:


```
velero restore describe <restore_name>
```

Validate the Restoration

To ensure that the backup of the platform restored correctly, perform the following steps:

1. Ensure that user management functions restored correctly:
 - a. Click **User Management** in the left navigation.
 - b. Check that users and user groups appear on the User Management tab.
 - c. If you had enabled LDAP, ensure that LDAP is still enabled.
2. Ensure that projects restored correctly in each data exchange:
 - a. Click **Data Exchanges** in the left navigation, and select **Open Data Exchange**.
 - b. Click the avatar symbol in the top right corner of the page and select **Manage Exchange**.
 - c. Ensure that business information (tags, terms, data classes), projects, datasets, assets, and so on display correctly.
 - d. Click the avatar symbol in the top right corner of the page, and select **View Exchange**.
 - e. Verify that the right groups and users are assigned to the data exchange.
 - f. Repeat these steps for each data exchange.
3. Ensure that all data planes restored correctly:
 - a. Run a simple query for each data plane to ensure that you can connect to each one. (Each data exchange has its own data plane.)
4. Ensure that connections restored correctly:
 - a. Click **Connections** in the left navigation.
 - b. Select a connection.
 - c. Click **Test**.

7.1.4. Disaster Recovery

You should formulate and document a disaster recovery approach and plan, including extensive disaster failover testing to ensure the plan works.

If you want to establish a secondary platform at a second physical location (to ensure against complete data center loss), consider establishing a hot/cold setup. The platform is not designed for a hot/hot configuration.

Test regular data snapshots, and establish a restore process to bring the secondary (cold) location in line with the hot location.

7.2. System Logs and Auditing

Data guardians use logs to ensure that all data is provisioned in compliance with company policies and to audit changes to demonstrate data compliance. System administrators also use them to query logs and check for errors.

The platform records all changes to any object (policies, projects, requests) in addition to other important events, such as user logins. See the [list of audit events](#).

You can forward audit records to your preferred security information and event management (SIEM) solution. For example Splunk or QRadar.

Logs:

- are JSON files
- are Splunk compliant
- record all changes to any object, such as policies, projects, and requests
- can only be appended to, never modified (Changes to any object do not delete or modify existing log information)
- can be exported and filtered on audit events, object type, dates, and user ID
- can be archived on an unscheduled or scheduled basis

7.2.1. Log Queries and Examples

The platform generates platform logs from the proxy. You need the proxy URL to set up your integrated development environment (IDE) when running a query on the proxy. You can get this from within the platform:

1. Click **View Exchange** to open the data exchange.
2. Open the **Data Plane** tab.
3. Open the **Data Proxy** tab to see the proxy host details.

Next, run a query from an IDE, such as VisualStudio or IntelliJ IDEA to produce a log. The proxy produces a two-audit event record; one of action type `user.authentication` and another of action type `policy.resolution`.

Note that the query runs against database metadata, not the platform metadata.

For the query audit event, you can set it to be verbose or non-verbose with an application property of: `privitar.jdbcproxy.queryengine.verboseAuditLog`. By default it is false. If you set it to true, the query audit events will also include information about the request query and the transformations applied to the field.

The audit event does not include information about the execution nor status of the request (query).

Audit Log Field Descriptions

Table 25. Audit Log Field Descriptions

Field	Description
<code>header.id</code>	A unique audit ID, generated for each event.
<code>header.event.action</code>	A system event which generates this audit record. The format of this field is <code>object type.action</code> . For example, <code>asset.create</code> on a create asset event, or <code>policy.resolution</code> on a policy resolution event.
<code>header.event.eventType</code>	Event type generating this event. For example, <code>ACTIVITY</code> .
<code>header.event.eventTime</code>	The time that the event generated the audit record.
<code>header.initiator.id</code>	The username, group name, or system name of the entity that triggered the event. For certain events, this can be NULL.

Field	Description
<code>header.initiator.typeURI</code>	Whether initiator is a user or <code>system_user</code> .
<code>header.initiator.tenant</code>	The tenant ID where the event takes place. For certain events, this can be NULL.
<code>header.observer.id</code>	The system observing the event.
<code>header.status.outcome</code>	Outcome of event is either <code>SUCCESS</code> or <code>FAILURE</code> .
<code>header.status.reason</code>	Optional error message when the outcome is <code>FAILURE</code> .
<code>header.target.id</code>	The target system ID observing the event. When running on Kubernetes this is the node ID. In other environments this is the ID of the data plane.
<code>header.target.typeURI</code>	The target system type.
<code>header.severity</code>	The impact of an event on the system. <ul style="list-style-type: none"> • <code>NORMAL</code> for a read action • <code>WARNING</code> for a create/update action • <code>CRITICAL</code> for a delete action or user login
<code>attachments.source.namespace</code> <code>attachments.source.pod</code>	Optional. When running in Kubernetes, this is the namespace and pod where the event takes place.
<code>attachments.correlationId</code>	The ID to link to the incoming request ID that generates this event. For workflow events, the <code>correlationId</code> can be used to link the different workflow events to a single workflow.
<code>attachments.responseData</code>	Optional. This is the response data that the event generated.
<code>attachments.requestData</code>	Optional. This is the request data for the event.

Policy Resolution Attachment Detail

The following explains common attributes that you're likely to see in the attachments section of the JSON view of a policy resolution log file:

- `correlationId`—A common identifier for the request or related group of requests.
- `requestData`—Provides context and result detail for the policy resolution event.
 - `context`—The metadata used for policy resolution.
 - `connection`—A configuration for connecting to and reading data from a data source, such as a JDBC connection string.
 - `id`—The unique ID of the data source connection used to access the requested data.
 - `name`—The name of the data source connection used to access the requested data.
 - `groupIds`—The identifier of the group who requested the data.
 - `project`—The consumption project used to generate the request.
 - `id`—The unique ID of the consumption project used to generate the request.
 - `name`—The name of the consumption project used to generate the request.

- `request`—Variable based on the type of request. Either `metadata` or `query`.
 - `metadata`—Issued when data owner requests metadata for a data source (always `get columns`).
 - `query`—Issued when the data consumer queries data (always a `select` statement).
- `userId`—The identifier of the user who requested the data.
- `userName`—The username of the user who requested the data.
- `fieldTriggers`—Describe how policies were triggered.
 - `fields`—Fields that were included in the query.
 - `assetId`—Unique identifier for the asset.
 - `assetName`—Name of the asset as described in the data exchange.
 - `catalog`—The dataset name.
 - `field`—The field that was included in the query.
 - `schema`—The name of the schema that the table and field belong to.
 - `table`—The name of the table that the field belongs to.
 - `trigger`—Describes which condition was triggered.
 - `policyId`—Unique identifier for the policy that was triggered.
 - `policyName`—The name of the policy that was triggered.
 - `ruleCondition`—The rule condition that caused the policy to trigger.
 - `ruleId`—The unique identifier of the rule that caused the policy to trigger.
 - `ruleName`—The name of the rule that caused the policy to trigger.
- `filters`—The record-level access control (RLAC) filters that were triggered.
 - `expression`—The RLAC rule that was triggered.
 - `field`—The field included in the RLAC rule.
 - `assetId`—Unique identifier for the asset.
 - `assetName`—Name of the asset as described in the data exchange.
 - `catalog`—The dataset name.
 - `field`—The field that was included in the query.
 - `schema`—The name of the schema that the table and field belong to.
 - `table`—The name of the table that the field belongs to.
- `request`—Variable based on the type of request. Either `metadata` or `query`.
 - `metadata`—Issued when data owner requests metadata for a data source (always `get columns`).
 - `query`—Issued when the data consumer queries data (always a `select` statement).
- `transformations`—The masking transformations that were applied based on the triggered policies.
- `field`—The field that was masked by the transformation logic.
 - `assetId`—Unique identifier for the asset.
 - `assetName`—Name of the asset as described in the data exchange.

- `catalog`—The dataset name.
- `field`—The field that was included in the query.
- `schema`—The name of the schema that the table and field belong to.
- `table`—The name of the table that the field belongs to.
- `singleFieldTransformation`—The type of transformation applied to the field.
 - `transformationName`—The name given when the transformation was created.
 - `constant`, `drop`, `generalizeDate`, `numericRegex`, `redactWithNull`, `regex`, `retain`, or `truncate`—The transformation type applied to the field.
 - `consistencyGroupId`—The unique identifier of the group of tokens used in consistent tokenization.
 - `consistent`—`true` means that consistent tokenization was applied. `false` means means that consistent tokenization was not applied.
 - `regex`—The regular expression that the platform applied a to the field.
 - `watermarkingRegex`—The watermarking regular expression that the the platform applied to the field.
 - `value`—The value applied to the field for the Constant Text Value (`constant`) transformation type.
 - `length`—The number of characters to preserve or delete when applying the Truncate transformation type.
 - `preserveSelectedCharacters`—`true` means the platform preserved the characters indicated by `length` when applying the Truncate transformation type. `false` means the platform deleted the characters indicated by `length` when applying the Truncate transformation type.
 - `truncateFromStart`—`true` means the platform applied the Truncate transformation type from the beginning of the string. `false` means the platform applied the Truncate transformation type from the end of the string.
- `cellLevelTransformation`—The type of transformation applied to the field.
 - `else`—The condition applied if none of the `ifAndElseIfs` conditions were true.
 - `singleFieldTransformation`—The type of transformation applied to the field.
 - `transformationName`—The name given when the transformation was created.
 - `constant`, `drop`, `generalizeDate`, `numericRegex`, `redactWithNull`, `regex`, `retain`, or `truncate`—The transformation type applied to the field.
 - `consistencyGroupId`—The unique identifier of the group of tokens used in consistent tokenization.
 - `consistent`—`true` means that consistent tokenization was applied. `false` means means that consistent tokenization was not applied.
 - `regex`—The regular expression that the platform applied a to the field.
 - `watermarkingRegex`—The watermarking regular expression that the the platform applied to the field.
 - `value`—The value applied to the field for the Constant Text Value (`constant`) transformation type.
 - `length`—The number of characters to preserve or delete when applying the Truncate transformation type.

- `preserveSelectedCharacters—true` means the platform preserved the characters indicated by `length` when applying the Truncate transformation type. `false` means the platform deleted the characters indicated by `length` when applying the Truncate transformation type.
 - `truncateFromStart—true` means the platform applied the Truncate transformation type from the beginning of the string. `false` means the platform applied the Truncate transformation type from the end of the string.
- `ifAndElseIfs`—When true, a condition for which the platform applies a transformation.
- `logicalExpression`—Contains the expression and the affected fields.
- `expression`—The logic for evaluating whether the platform applied a cell-level transformation.
- `field`—A field that was masked by a cell-level transformation.
 - `assetId`—Unique identifier for the asset.
 - `assetName`—Name of the asset as described in the data exchange.
 - `catalog`—The dataset name.
 - `field`—The field that was included in the query.
 - `schema`—The name of the schema that the table and field belong to.
 - `table`—The name of the table that the field belongs to.
- `transformation`—The masking transformation that was applied based on the triggered policies.
- `singleFieldTransformation`—The type of transformation applied to the field.
 - `transformationName`—The name given when the transformation was created.
 - `constant, drop, generalizeDate, numericRegex, redactWithNull, regex, retain, or truncate`—The transformation type applied to the field.
 - `consistencyGroupId`—The unique identifier of the group of tokens used in consistent tokenization.
 - `consistent—true` means that consistent tokenization was applied. `false` means means that consistent tokenization was not applied.
 - `regex`—The regular expression that the platform applied a to the field.
 - `watermarkingRegex`—The watermarking regular expression that the the platform applied to the field.
 - `value`—The value applied to the field for the Constant Text Value (`constant`) transformation type.
 - `length`—The number of characters to preserve or delete when applying the Truncate transformation type.
 - `preserveSelectedCharacters—true` means the platform preserved the characters indicated by `length` when applying the Truncate transformation type. `false` means the platform deleted the characters indicated by `length` when applying the Truncate transformation type.

- `truncateFromStart=true` means the platform applied the Truncate transformation type from the beginning of the string. `false` means the platform applied the Truncate transformation type from the end of the string.
- `dataPlane`—Provides context about the data plane on which the policy resolution event occurred.
 - `id`—The unique identifier of the data plane.
- `rlacFilterBehaviour`—The default behavior of the RLAC filter as set by the exchange administrator; either "DENY_ACCESS" or "ALLOW_ACCESS".
- `responseData`—Not currently used.
- `source`—The data plane through which the audit event was issued.
 - `namespace`—The Kubernetes namespace of the data plane through which the audit event was issued.
 - `pod`—The Kubernetes pod on which the audit event was issued.

Example of the User Info Audit

```
{
  "header": {
    "id": "00f5fc6b-a1d2-4db1-be44-9d7360a25176",
    "event": {
      "eventTime": "2022-11-04T12:36:49.439Z",
      "eventType": "ACTIVITY",
      "action": "user.authentication"
    },
    "initiator": {
      "id": "29bd6d2a-10a3-44aa-8de7-1f264e22e21c",
      "typeURI": "User",
      "tenant": "fff7a844-f732-4a65-8280-912eae8eb8d0"
    },
    "observer": {
      "id": "target"
    },
    "status": {
      "outcome": "SUCCESS"
    },
    "target": {
      "id": "01234567-89ab-cdef-0123-456789abcdef",
      "typeURI": "service/compute"
    },
    "severity": "WARNING"
  },
  "attachments": [
    {
      "source": {
        "namespace": "dpp-proxy",
        "pod": "dynamic-proxy-b48ad3d7-7b7c-431f-ad5d-633e5641c0cd-59c677blq52m"
      },
      "correlationId": "93429272-b9e7-4452-b799-dfeddf5e3a47",
      "requestData": {
        "context": {
```



```

    "project": {
      "id": "5a078d49-1bc3-4a6f-9caa-1d622f18e7cc"
    },
    "requestAttributes": {
      "request.project": "5a078d49-1bc3-4a6f-9caa-1d622f18e7cc"
    }
  }
}
]
}

```

You can see that "action" tells us what type of audit log it is.

Example of the Non-verbose Query Audit Event

```

{
  "header": {
    "id": "40012d8a-f002-4fcf-87cd-549e66d57634",
    "event": {
      "eventTime": "2022-11-07T06:46:13.024082Z",
      "eventType": "ACTIVITY",
      "action": "policy.resolution"
    },
    "initiator": {
      "id": "ba992d9f-2c0c-476e-9584-9e8ef923a5bf",
      "typeURI": "User",
      "tenant": "bc5bbfa3-e07c-4688-8644-a516afe40c1b"
    },
    "observer": {
      "id": "target"
    },
    "status": {
      "outcome": "SUCCESS",
      "reason": null
    },
    "target": {
      "id": "ip-10-18-122-83.eu-west-1.compute.internal",
      "typeURI": "service/compute"
    },
    "severity": "WARNING"
  },
  "attachments": [
    {
      "source": {
        "namespace": "dpp-8193-common-audit-dp",
        "pod": "dynamic-proxy-b48ad3d7-7b7c-431f-ad5d-633e5641c0cd-59c677blq52m"
      },
      "correlationId": "b93069e5-676b-41ce-9c2d-c69c2b5d9360",
      "responseData": null,
      "requestData": {
        "transformations": [
          {
            "field": [

```

```

        "catalog": "acmedb",
        "schema": "library",
        "table": "authors",
        "field": "firstname"
      }
    ],
    },
    {
      "field": [
        {
          "catalog": "acmedb",
          "schema": "library",
          "table": "authors",
          "field": "lastname"
        }
      ]
    },
    {
      "field": [
        {
          "catalog": "acmedb",
          "schema": "library",
          "table": "authors",
          "field": "id"
        }
      ]
    }
  ],
  "filters": [
    {
      "field": {
        "catalog": "acmedb",
        "schema": "library",
        "table": "authors",
        "field": "firstname"
      },
      "expression": "firstname <> 'AAAAA'"
    }
  ]
}

```

Example of the Verbose Query Audit Event for a SQL Query

Turn auditing on, is true by default: `privitar.audit.enabled=true`

Configure auditing to emit records to Fluent Bit:

`privitar.audit.auditServiceUrl=true`

```

{
  "header": {
    "id": "00f5fc6b-a1d2-4db1-be44-9d7360a25176",
    "event": {
      "eventTime": "2022-11-04T12:36:49.439Z",

```

```

    "eventType": "ACTIVITY",
    "action": "policy.resolution"
  },
  "initiator": {
    "id": "successfulUser",
    "typeURI": "User",
    "tenant": "fff7a844-f732-4a65-8280-912eae8eb8d0"
  },
  "observer": {
    "id": "target"
  },
  "status": {
    "outcome": "SUCCESS",
    "reason": null
  },
  "target": {
    "id": "01234567-89ab-cdef-0123-456789abcdef",
    "typeURI": "data_plane"
  },
  "severity": "WARNING"
},
"attachments": [
  "source": {
    "namespace": "dpp-proxy",
    "pod": "dynamic-proxy-b48ad3d7-7b7c-431f-ad5d-633e5641c0cd-59c677blq52m"
  },
  {
    "userId": "proxy-user",
    "request": {
      "query": "select * from library.authors"
    },
    "correlationId": "93429272-b9e7-4452-b799-dfeddf5e3a47",
    "responseData": null,
    "requestData": {
      "transformations": [
        {
          "field": [
            {
              "catalog": "acmedb",
              "schema": "library",
              "table": "authors",
              "field": "lastname"
            }
          ],
          "singleFieldTransformation": {
            "retain": {}
          }
        }
      ],
      {
        "field": [
          {
            "catalog": "acmedb",
            "schema": "library",
            "table": "authors",
            "field": "firstname"
          }
        ]
      }
    ]
  }
]

```

```

    }
  ],
  "singleFieldTransformation": {
    "regex": {
      "regex": "[A-Za-z]{1,15}"
    }
  }
},
{
  "field": [
    {
      "catalog": "acmedb",
      "schema": "library",
      "table": "authors",
      "field": "id"
    }
  ],
  "singleFieldTransformation": {
    "retain": {}
  }
}
],
"filters": [
  {
    "field": {
      "catalog": "acmedb",
      "schema": "library",
      "table": "authors",
      "field": "firstname"
    },
    "expression": "firstname <> 'AAAAA'"
  }
]
}
]
}
}

```

Example of the Verbose Query Audit Event for a Metadata Request

```

{
  "header": {
    "id": "00f5fc6b-a1d2-4db1-be44-9d7360a25176",
    "event": {
      "eventTime": "2022-11-04T12:36:49.439Z",
      "eventType": "ACTIVITY",
      "action": "policy.resolution"
    },
    "initiator": {
      "id": "proxy-user",
      "typeURI": "User",
      "tenant": "fff7a844-f732-4a65-8280-912eae8eb8d0"
    },
    "observer": {
      "id": "target"
    }
  },

```

```

"status": {
  "outcome": "SUCCESS"
},
"target": {
  "id": "01234567-89ab-cdef-0123-456789abcdef",
  "typeURI": "data_plane"
},
"severity": "WARNING"
},
"attachments": [
{
  "userId": "proxy-user",
  "request": {
    "metadata": "get tables"
  },
  "correlationId": "93429272-b9e7-4452-b799-dfeddf5e3a47",
  "requestData": {
    "transformations": [
      {
        "field": [
          {
            "catalog": "acmedb",
            "schema": "library",
            "table": "authors",
            "field": "firstname"
          }
        ],
        "singleFieldTransformation": {
          "regex": {
            "regex": "[A-Za-z]{1,15}"
          }
        }
      },
      {
        "field": [
          {
            "catalog": "acmedb",
            "schema": "library",
            "table": "authors",
            "field": "id"
          }
        ],
        "singleFieldTransformation": {
          "retain": {}
        }
      }
    ],
    {
      "field": [
        {
          "catalog": "acmedb",
          "schema": "library",
          "table": "authors",
          "field": "lastname"
        }
      ],
      "singleFieldTransformation": {

```

```

        "retain": {}
      }
    ],
    "filters": [
      {
        "field": {
          "catalog": "acmedb",
          "schema": "library",
          "table": "authors",
          "field": "firstname"
        },
        "expression": "firstname <> 'AAAAA'"
      }
    ]
  }
}

```

Logs for Possible Failures

There are also audit logs for some possible failures, such as when user authentication has failed:

```

{
  "header": {
    "id": "00f5fc6b-a1d2-4db1-be44-9d7360a25176",
    "event": {
      "eventTime": "2022-11-04T12:36:49.439Z",
      "eventType": "ACTIVITY",
      "action": "user.authentication"
    },
    "initiator": {
      "id": "user@gmail.com",
      "typeURI": "User",
      "tenant": "fff7a844-f732-4a65-8280-912eae8eb8d0",
      "sourceIp": null
    },
    "observer": {
      "id": "target"
    },
    "status": {
      "outcome": "FAILURE",
      "reason": {
        "message": "Received error while trying to authenticate username
user@gmail.com: Server Error"
      }
    },
    "target": {
      "id": "01234567-89ab-cdef-0123-456789abcdef",
      "typeURI": "data_plane"
    },
    "severity": "WARNING"
  },
  "attachments": [

```

```
]
}
```

This log shows a policy retrieval failure.

```
{
  "header": {
    "id": "00f5fc6b-a1d2-4db1-be44-9d7360a25176",
    "event": {
      "eventTime": "2022-11-04T12:36:49.439Z",
      "eventType": "ACTIVITY",
      "action": "policy.resolution"
    },
    "initiator": {
      "id": "proxy-user",
      "typeURI": "User",
      "tenant": "fff7a844-f732-4a65-8280-912eae8eb8d0",
      "sourceIp": null
    },
    "observer": {
      "id": "target"
    },
    "status": {
      "outcome": "FAILURE",
      "reason": {
        "message": "Failed to retrieve policies. Internal server error"
      }
    }
  },
  "target": {
    "id": "01234567-89ab-cdef-0123-456789abcdef",
    "typeURI": "data_plane"
  },
  "severity": "WARNING"
},
"attachments": []
}
```

7.2.2. Audit Events

This section lists all audit event types, organized by:

- [Administration Events](#)
- [Business Information Events](#)
- [Connection Events](#)
- [Data Proxy Events](#)
- [Dataset and Asset Events](#)
- [Policy Events](#)
- [Project Events](#)

Administration Events

Table 26. User Access Events

Event	Description
edit.rlac_filter_behaviour	Default behavior for access control rules changed
login	User logged in to data exchange
logout	User logged out of data exchange

Table 27. Data Plane Events

Event	Description
create.data_plane	New data plane created
create.data_agent	New data agent created
create.dynamic_proxy	New data proxy created
update.data_plane	Data plane updated
update.data_agent	Data agent updated
update.dynamic_proxy	Data proxy updated
delete.data_plane	Data plane deleted
delete.data_agent	Data agent deleted
delete.dynamic_proxy	Data proxy deleted

Business Information Events

Table 28. Attribute Events

Event	Description
create.attribute_type	New request type created
update.attribute_type	Request type edited
delete.attribute_type	Request type deleted

Table 29. Data Class Events

Event	Description
create.data_class	Data class created
update.data_class	Data class edited
delete.data_class	Data class deleted

Table 30. Tag Events

Event	Description
create.tag	Tag created
update.tag	Tag edited
delete.tag	Tag deleted

Table 31. Term Events

Event	Description
create.term	Term created
update.term	Term edited
delete.term	Term deleted

Connection Events

Table 32. Connection Events

Event	Description
create.connection	New connection created
update.connection	Connection edited
delete.connection	Connection deleted

Data Proxy Events

Table 33. Data Proxy Events

Event	Description
authentication.user	<p>Connection established to data proxy when data queried</p> <p>You can view this in the platform user interface (UI) by clicking Audit Log in the left navigation.</p> <p>To learn more, see "Viewing Audit Logs" in the DSP User Guide.</p>
policy.resolution	<p>The platform received a data request, validating the request and inferring which policies applied</p> <p>You can view this in the platform user interface (UI) by clicking Audit Log in the left navigation.</p> <p>To learn more, see "Viewing Audit Logs" in the DSP User Guide.</p>

Dataset and Asset Events

Table 34. Dataset Events

Event	Description
create.dataset	New dataset created
update.dataset	Dataset updated
delete.dataset	Dataset deleted

Table 35. Asset Registration Events

Event	Description
start.asset_creation_request	New asset registered

Event	Description
edit.asset_creation_request	Fields classified during asset registration Or unpublished, rejected asset updated
submit.asset_creation_request	Asset submitted for approval
approve.asset_creation_request	Asset request approved
publish.asset	Asset published to data exchange
reject.asset_creation_request	Asset request rejected

Table 36. Asset Update Events

Event	Description
start.asset_update_request	Published asset updated
edit.asset_update_request	Fields classified during asset update Or published asset updated
submit.asset_update_request	Asset submitted for approval
approve.asset_update_request	Asset update request approved
reject.asset_update_request	Update to published asset rejected

Table 37. Asset Deletion Events

Event	Description
submit.asset_deletion_request	Asset deleted
discard.asset_creation_request	Draft (unpublished) asset deleted
delete.asset	Asset deletion request approved
reject.asset_deletion_request	Asset deletion request rejected

Policy Events

Table 38. Transformation Events

Event	Description
create.transformation	New transformation created
update.transformation	Transformation updated
delete.transformation	Transformation deleted

Table 39. Default Transformation Policy Events

Event	Description
submit.default_transformation_policy_update_request	Default transformation policy changed and submitted for approval
approve.default_transformation_policy_update_request	Changed default transformation policy approved

Event	Description
<code>publish.default_transformation_policy_update_request</code>	Changed default transformation policy published

Table 40. Access Control Policy Creation Events

Event	Description
<code>start.access_control_policy_creation_request</code>	New access control policy created
<code>edit.access_control_policy_creation_request</code>	New draft access control policy edited
<code>create.access_control_rule.access_control_policy_creation_request</code>	New access control rule created
<code>delete.access_control_rule.access_control_policy_creation_request</code>	Access control rule deleted
<code>submit.access_control_policy_creation_request</code>	New access control policy submitted for approval
<code>approve.access_control_policy_creation_request</code>	New access control policy approved
<code>publish.access_control_policy</code>	Access control policy published
<code>reject.access_control_policy_creation_request</code>	Access control policy creation request rejected

Table 41. Access Control Policy Update Events

Event	Description
<code>start.access_control_policy_update_request</code>	Update to published access control policy initiated
<code>edit.access_control_policy_update_request</code>	Access control policy updated
<code>edit.access_control_rule.access_control_policy_update_request</code>	Access control rule updated
<code>delete.access_control_rule.access_control_policy_update_request</code>	Access control rule deleted
<code>submit.access_control_policy_update_request</code>	Edited access control policy submitted for approval
<code>approve.access_control_policy_update_request</code>	Edited access control policy approved

Table 42. Access Control Policy Deletion Events

Event	Description
<code>delete.access_control_policy</code>	Access control policy deleted
<code>discard.access_control_policy_creation_request</code>	Unpublished access control policy deleted
<code>discard.access_control_policy_update_request</code>	Draft of unpublished access control policy deleted

Table 43. Transformation Policy Creation Events

Event	Description
<code>start.transformation_policy_creation_request</code>	New transformation policy created
<code>edit.transformation_policy_creation_request</code>	Draft of transformation policy edited

Event	Description
<code>create.transformation_rule.transformation_policy_creation_request</code>	New transformation rule created Or transformation rule duplicated
<code>delete.transformation_rule.transformation_policy_creation_request</code>	Transformation rule deleted
<code>reorder.transformation_rule.transformation_policy_create_request</code>	Transformation rule reordered
<code>submit.transformation_policy_creation_request</code>	Transformation policy submitted for approval
<code>approve.transformation_policy_creation_request</code>	Transformation policy approved
<code>publish.transformation_policy</code>	Transformation policy published
<code>reject.transformation_policy_creation_request</code>	Transformation policy creation request rejected
<code>reorder.transformation_policy</code>	Published transformation policy reordered

Table 44. Transformation Policy Update Events

Event	Description
<code>start.transformation_policy_update_request</code>	Update to published transformation policy initiated
<code>edit.transformation_policy_update_request</code>	Transformation policy updated
<code>edit.transformation_rule.transformation_policy_update_request</code>	Transformation rule updated
<code>create.transformation_rule.transformation_policy_update_request</code>	Transformation rule duplicated
<code>create.transformation_rule.transformation_policy_update_request</code>	Transformation rule created
<code>delete.transformation_rule.transformation_policy_update_request</code>	Transformation rule deleted

Event	Description
<code>reorder.transformation_rule.transformation_policy_update_request</code>	Transformation rule reordered
<code>submit.transformation_policy_update_request</code>	Transformation policy submitted for approval
<code>approve.transformation_policy_update_request</code>	Transformation policy approved
<code>reject.transformation_policy_update_request</code>	Transformation policy update request declined

Table 45. Transformation Policy Deletion Events

Event	Description
<code>delete.transformation_policy</code>	Transformation policy deleted Or transformation policy with published rules deleted
<code>discard.transformation_policy_creation_request</code>	Unpublished transformation policy deleted
<code>discard.transformation_policy_update_request</code>	Draft of published transformation policy deleted

Project Events



Note

Audit event details for projects follow the pattern below for consumption projects, but may include extraneous information for migration projects.

Table 46. Project Creation Events

Event	Description
<code>start.project_creation_request</code>	New project created
<code>edit.project_creation_request.add_asset</code>	Asset added to project
<code>edit.project_creation_request.remove_asset</code>	Asset removed from project
<code>submit.project_creation_request</code>	Project submitted for approval
<code>approve.project_creation_request</code>	Project creation request approved
<code>publish.project</code>	Project published

Table 47. Project Update Events

Event	Description
start.project_update_request	Project update initiated
edit.project_update_request.details	Project's details updated
edit.project_update_request.add_asset	Asset added to project
edit.project_update_request.remove_asset	Asset removed from a project
submit.project_update_request	Project update request submitted
approve.project_update_request	Project update request approved
reject.project_update_request	Project update request rejected

Table 48. Project Deletion Events

Event	Description
delete.project	Published project deleted
discard.project_creation_request	Draft project deleted
discard.project_update_request	Draft project deleted

7.2.3. Privitar Query Engine and Data Proxy Metrics

Using a third-party event monitoring tool, such as [Prometheus](#), you can capture the following metrics from the Privitar Query Engine and data proxy:

Table 49. Privitar Query Engine Metrics

Metric	Description	Type
com_privitar_query_engine_jdbc_handler	Time spent executing the given action An equivalent metric exists in the data proxy as com_privitar_data_proxy_jdbc_handler	Time r
com_privitar_query_engine_policy_enforcement	Time spent: <ul style="list-style-type: none"> • SQL parsing • policy and metadata loading • policy enforcement • SQL generation 	Time r
com_privitar_query_engine_policy_load	Time spent loading policies from the control plane	Time r
com_privitar_query_engine_connection_details_load	Time spent loading connection details (such as JDBC URL, username, and password) from the control plane	Time r
com_privitar_query_engine_underlying_db_metadata_load	Time spent loading table metadata from the underlying database	Time r

Table 50. Data Proxy Metrics

Metric	Description	Type
time_of_last_request_since_epoch	The timestamp of the last request received by the data proxy. This is useful to see whether a proxy has been active recently	Gauge
com_privitar_data_proxy_jdbc_handler	Time spent executing the given action An equivalent metric exists in the Privitar Query Engine as com_privitar_query_engine_jdbc_handler	Timer

Some metrics have an action tag to discriminate between different actions covered by the same metric name. The following are explanations of each action tag:

Table 51. Action Tags

Action Tag	Description	Note
GET_DATABASE_PROPERTIES	Returns a map of static database properties	
GET_TABLES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_COLUMNS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_SCHEMAS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_CATALOGS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_TABLE_TYPES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_PROCEDURES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.

Action Tag	Description	Note
GET_PROCEDURE_COLUMNS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_COLUMN_PRIVILEGES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_TABLE_PRIVILEGES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_BEST_ROW_IDENTIFIER	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_VERSION_COLUMNS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_PRIMARY_KEYS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_IMPORTED_KEYS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_EXPORTED_KEYS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_CROSS_REFERENCE	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_TYPE_INFO	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.

Action Tag	Description	Note
GET_INDEX_INFO	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_UDTS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_SUPER_TYPES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_SUPER_TABLES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_ATTRIBUTES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_CLIENT_INFO_PROPERTIES	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_FUNCTIONS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_FUNCTION_COLUMNS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
GET_PSEUDO_COLUMNS	Executed when the corresponding method on the java.sql.DatabaseMetaData interface is executed	See the Javadoc of that method for more information.
CREATE_ITERABLE	Creates an iterable for a result set	
PREPARE	Prepares a statement	Executed when java.sql.Connection#prepareStatement is used

Action Tag	Description	Note
PREPARE_AND_EXECUTE	Prepares and executes a statement, and if the query has a result set then fetches the first frame of data	Changed from PREPARE_AND_EXECUTE_DEPRECATED for v1.1.1 of the platform
PREPARE_AND_EXECUTE_BATCH	Prepares a statement and then executes a number of SQL commands in one pass	Executed when java.sql.Statement#executeBatch is used
EXECUTE_BATCH	Executes a collection of bound parameter values on a prepared statement	Executed when java.sql.PreparedStatement#executeBatch is used
FETCH	Returns a frame of rows	
EXECUTE	Executes a prepared statement	Executes a prepared statement. Executed when java.sql.PreparedStatement#execute is used. Changed from EXECUTE_DEPRECATED for v1.1.1 of the platform
CREATE_STATEMENT	Called during the creation of a statement	
CLOSE_STATEMENT	Closes a statement	Executed when java.sql.Statement#close is used
OPEN_CONNECTION	Opens (creates) a connection	
CLOSE_CONNECTION	Closes a connection	
SYNC_RESULTS	Resets the ResultSet on a statement	For Privitar use only. Not a JDBC method. Used to manage internal state
COMMIT	Makes all changes since the last commit or rollback permanent	Executed when java.sql.Connection#commit is used
ROLLBACK	Undoes all changes since the last commit or rollback	Executed when java.sql.Connection#rollback is used
CONNECTION_SYNC	Synchronizes client and server view of connection properties	For Privitar use only. Not a JDBC method. Used to manage internal state

7.3. Monitoring Metrics

To monitor metrics on the platform and the control plane, you can capture metrics with [Kubernetes](#), extract metrics with [Bitnami Prometheus](#), and visualize the metrics on [Grafana](#) dashboards.

7.3.1. Monitoring the Privitar Data Security Platform

You can monitor DSP with either of two types of Prometheus Server:

- Bitnami Prometheus Server
- Prometheus–Community Server

Monitoring DSP: Before You Begin



Important

You must [install Bitnami Prometheus](#) before you [install the control plane](#) in order to enable the built-in metrics.

1. Create a namespace for storing all the monitoring components:

```
kubectl create namespace prometheus
kubectl label namespace prometheus istio-injection=enabled --overwrite
```

2. Create a file called `prometheus-global.yaml` that includes the following:

```
server:
  enabled: true
  extraFlags:
    - web.enable-lifecycle
    - web.enable-remote-write-receiver
```

3. Choose one of the following options to install Prometheus Server:

- [Option A: Install Prometheus–Community Server](#)

Choose this option if you wish to manually configure and export your own metrics.

- [Option B: Install Bitnami Prometheus Server](#)

Choose this option if you wish to use the built-in metrics, which you enabled during installation of the platform.

Option A: Install Prometheus–Community Server

Choose this option if you wish to manually configure and export your own metrics.

1. Add the Prometheus and Grafana helm chart repos:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
```

2. Install the Prometheus components:

```
helm install prometheus prometheus-community/prometheus -n prometheus \
  --set server.persistentVolume.storageClass="standard-rwo" \
  --set server.persistentVolume.size=8Gi \
  --set server.statefulSet.enabled=true \
```

```
--set alertmanager.persistence.storageClass="standard-rwo" \
-f prometheus-global.yaml
```

3. If necessary, expose the Prometheus port so that you can access the user interface (UI):

```
kubectl expose service prometheus-server \
--type=LoadBalancer --port=9090 --target-port=9090 \
--name=prometheus-server-ext \
-n prometheus
```

4. Check the logs to see whether there are any errors:

```
kubectl logs -f prometheus-server-0 -n prometheus
```

Option B: Install Bitnami Prometheus Server

Choose this option if you wish to use the built-in metrics, which you enabled during installation of the platform.

1. Install the Prometheus components:

```
helm install kube-prometheus bitnami/kube-prometheus -n prometheus \
--set server.persistentVolume.storageClass="standard-rwo" \
--set server.persistentVolume.size=8Gi \
--set server.statefulSet.enabled=true \
--set alertmanager.persistence.storageClass="standard-rwo" \
-f prometheus-global.yaml
```

2. If necessary, expose the Prometheus port so that you can access the user interface (UI):

```
kubectl expose service kube-prometheus-prometheus \
--type=LoadBalancer --port=9090 --target-port=9090 \
--name=prometheus-server-ext \
-n control-plane
```

3. Verify that CRDs have been installed:

```
$ kubectl get crds -A | grep prometheus
prometheuses.monitoring.coreos.com          2023-06-07T10:52:56Z
prometheusrules.monitoring.coreos.com       2023-06-07T10:52:57Z
$
```

4. If not already enabled, redeploy the control plane to enable built-in metrics:
 - a. Add the following to the `control-plane-values.yaml` file:

```
metrics:
  enabled: true
```

- b. For each deployment, set the replicas to 0:

```
kubectl scale deployment/xxxx --replicas=0 -n control-plane
```

- c. For each StatefulSet, set the replicas to 0:

```
kubectl scale deployment/xxxx --replicas=0 -n control-plane
```

- d. Redeploy the control plane, setting `clean: false`:

```
./dppctl install -f control-plane-values.yaml
```

- e. Verify that PrometheusRules have been defined:

```
$ kubectl get PrometheusRules -n control-plane
NAME                                     AGE
control-plane-postgres-postgresql      25h
control-plane-rabbitmq                  25h
$
```

5. If not already enabled, redeploy the data plane to enable built-in metrics:

- a. Add the following to the `data-plane-values.yaml` file:

```
metrics:
  enabled: true
```

- b. For each deployment, set the replicas to 0:

```
kubectl scale deployment/xxxx --replicas=0 -n control-plane
```

- c. For each StatefulSet, set the replicas to 0:

```
kubectl scale deployment/xxxx --replicas=0 -n control-plane
```

- d. Redeploy the data plane, setting `clean: false`:

```
./dppctl install -f data-plane-values.yaml
```

Monitoring DSP: Configure Prometheus Server

1. If you have exposed the UI port, load up the UI (<http://prometheus-server:9090>), and go to Status > Targets to check that the Prometheus server is able to identify the targets.
2. Edit the `configmap/prometheus-server` to ensure the following is seen in `scrape_configs`:

```
- job_name: prometheus
  static_configs:
    - targets:
      - localhost:9090
```

3. Restart the `prometheus-server` StatefulSet:

```
kubectl rollout restart statefulset/prometheus-server -n prometheus
```

Monitoring DSP: Install Grafana

1. Install the Grafana components:

```
helm install grafana grafana/grafana -n prometheus \
--set persistence.storageClassName="standard-rwo" \
--set persistence.type=statefulset \
--set persistence.size=8Gi \
```

```
--set persistence.enabled=true \
--set adminPassword="myGrafana@adminPassword" \
--set service.type=LoadBalancer
```

2. Expose the Grafana port for the UI:

```
kubectl expose service grafana \
--type=LoadBalancer --port=3000 --target-port=3000 \
--name=grafana-ext \
-n prometheus
```

Monitoring DSP: Enable Metrics Collection from Control Plane Services

By default, installing Prometheus and Grafana allows you to collect metrics from Kubernetes components, including the cluster, nodes, pods, services, and more.

If you require additional metrics from the individual components in the control plane, set up metrics exporters for each component.

Monitoring DSP: Enable Redis

1. Install the exporter for Redis:

```
redis_password=`kubectl get -o yaml secret/dpp-ui-redis-secret -n
control-plane | grep redis-password | sed "s/.*/ /" | base64 --decode`
helm upgrade -i prometheus-redis-exporter \
prometheus-community/prometheus-redis-exporter \
--set redisAddress="" \
--set auth.enabled=true \
--set auth.redisPassword=${redis_password} \
-n prometheus
```

2. Edit the `configmap/prometheus-server` to ensure the following is seen in `scrape_configs` (you may need to create the metrics endpoints for the `redis-master` and `redis-replicas`):

```
- job_name: redis_exporter_targets
static_configs:
- targets:
- redis://redis-master-metrics.control-
plane.svc.cluster.local:6379
- redis://redis-replica-smetrics.control-
plane.svc.cluster.local:6379
metrics_path: /scrape
relabel_configs:
- source_labels: [__address__]
target_label: __param_target
- source_labels: [__param_target]
target_label: instance
- target_label: __address__
replacement: prometheus-redis-exporter:9121
- job_name: 'redis_exporter'
static_configs:
- targets:
- prometheus-redis-exporter:9121
```

- Restart the prometheus-server StatefulSet:

```
kubectl rollout restart statefulset/prometheus-server -n prometheus
```

For more information on detailed configuration for the Redis Exporter, see: https://github.com/oliver006/redis_exporter#prometheus-redis-metrics-exporter

Monitoring DSP: Enable PostgreSQL



Note

DSP does not fully support the command `pg_stat_statements` and does not load the extension on startup.

- Determine the IP for the control-plane-postgres-postgresql service:

```
kubectl get services -A | grep control-plane-postgres-postgresql
```

- Install the exporter for PostgreSQL:

```
postgresql_password=`kubectl get -o yaml secret/dpp-control-plane-postgres-secret -n ${control_plane_namespace} | grep postgres-password | sed "s/.* // " | base64 --decode`
helm upgrade -i prometheus-postgres-exporter prometheus-community/prometheus-postgres-exporter -n ${namespace} \
--set config.datasource.host="ip.of.controlplanepostgres.above" \
--set config.datasource.user="postgres" \
--set config.datasource.password=${postgresql_password}
```

- Log in to the PostgreSQL instance:

- Log in to the pod:

```
kubectl exec -it control-plane-postgres-postgresql-0 -n control-plane -- /bin/bash
```

- Log in to PostgreSQL with the password from step 2.

```
psql -U postgres
```

- Determine whether the extension `pg_stat_statements` is installed:

```
select * from pg_available_extensions where
name='pg_stat_statements';
```

- Go into the Flowable database and create the extension:

```
\c flowable
create extension pg_stat_statements;
\q
```

- Exit the instance:

```
exit
```


4. Determine the IP for the prometheus-postgres-exporter service:

```
kubectl get services -A | prometheus-postgres-exporter
```

5. Check the logs for the PostgreSQL exporter to check if there are any errors.
6. Edit the configmap/prometheus-server to ensure the following is seen in scrape_configs:

```
- job_name: postgresql-cp
  static_configs:
  - targets:
    - control-plane-postgres-postgresql-metrics.control-
      plane.svc.cluster.local:9187
```

7. Restart the prometheus-server StatefulSet:

```
kubectl rollout restart statefulset/prometheus-server -n prometheus
```

Monitoring DSP: Enable Keycloak PostgreSQL

1. Determine the IP for the postgresql-ha-postgresql service:

```
kubectl get services -A | postgresql-ha-postgresql
```

2. Install the exporter for PostgreSQL:

```
postgresql_kc_password=`kubectl get -o yaml secret/dpp-keycloak-db-
secret -n ${control_plane_namespace} | grep postgresql-postgres-password
| sed "s/.* // " | base64 --decode`
helm upgrade -i prometheus-postgres-exporter-kc prometheus-community/
prometheus-postgres-exporter -n ${namespace} \
  --set config.datasource.host="ip.of.keycloakpostgres.above" \
  --set config.datasource.user="postgres" \
  --set config.datasource.password=${postgresql_kc_password}
```

3. Log in to the Keycloak PostgreSQL instance:

- a. Log into the pod:

```
kubectl exec -it postgresql-ha-postgresql-0 -n control-plane -- /bin/
bash
```

- b. Log in to PostgreSQL with the password from step 2.

```
psql -U postgres
```

- c. Determine whether the extension pg_stat_statements is installed:

```
select * from pg_available_extensions where
name='pg_stat_statements';
```

- d. Go into the Flowable database and create the extension:

```
\c keycloak
create extension pg_stat_statements;
\c repmgr
create extension pg_stat_statements;
\q
```

- e. Exit the instance:

```
exit
```

4. Determine the IP for the `prometheus-postgres-exporter-kc` service:

```
kubectl get services -A | prometheus-postgres-exporter-kc
```

5. Check the logs for the Keycloak PostgreSQL exporter for any errors.
6. Edit the configmap for Prometheus to include the job for Keycloak:

```
- job_name: postgresql-kc
  static_configs:
    - targets:
      - control-plane-postgres-postgresql-metrics.control-
        plane.svc.cluster.local:9187
```

7. Restart the `prometheus-server` StatefulSet:

```
kubectl rollout restart statefulset/prometheus-server -n prometheus
```

Monitoring DSP: Enable Metrics Collection from Data Plane Services

To collect the metrics for data plane services, such as the data agent and the data proxy, expose the metrics ports. No exporters are required.

1. Obtain the names of the pods:

```
data_agent_pod=`kubectl get pods -n ${data_plane_namespace} | grep data-
agent | cut -f1 -d' '`
dynamic_proxy_pod=`kubectl get pods -n ${data_plane_namespace} | grep
dynamic-proxy | cut -f1 -d' '`
```

2. Expose the pods:

```
kubectl expose pod ${data_agent_pod} --type=ClusterIP --port=9095 \
--target-port=9095 --name=data-agent-metrics -n data-plane
kubectl expose pod ${dynamic_proxy_pod} --type=ClusterIP --port=9095 \
--target-port=9095 --name=dynamic-proxy-metrics -n data-plane
```

3. Edit the configmap for Prometheus to include the job for the data agent and the data proxy:

```
- job_name: data-agent
  static_configs:
    - targets:
      - data-agent-metrics.data-plane.svc.cluster.local:9095
  metrics_path: /metrics
- job_name: dynamic-proxy
  static_configs:
    - targets:
      - dynamic-proxy-metrics.data-plane.svc.cluster.local:9095
```

4. Restart the `prometheus-server` StatefulSet:

```
kubectl rollout restart statefulset/prometheus-server -n prometheus
```

Monitoring DSP: Collect Metrics from Other Clusters



Note

You only need agents if you wish to collect metrics from outside the cluster where Prometheus Server is installed.

If you have other clusters that you also wish to collect the metrics from, you can make use of Prometheus' `remote-write` feature. Set up Prometheus in agent mode in the other clusters, and configure them to send the metrics by `remote-write` to the Prometheus global server.

To do this, you enable the `remote-write` feature by configuring the Prometheus Server to receive remote-writes, and configuring one or more remote agents to push metrics to the Prometheus Server through `remote-write`. You deploy the agents as Kubernetes deployments. This is in contrast to the global Prometheus server, which you deploy as a `StatefulSet`.

To learn more, see: <https://medium.com/@ehsan-khodadadi/prometheus-multi-cluster-monitoring-using-prometheus-agent-mode-cab2cdb20c11>

1. Create the following `prometheus-agent.yaml` file:

```
server:
  enabled: true
  defaultFlagsOverride:
    - --enable-feature=agent
    - --web.enable-lifecycle
    - --storage.agent.retention.max-time=30m
    - --config.file=/etc/config/prometheus.yml
  configPath: /etc/config/prometheus.yml

serverFiles:
  prometheus.yml:
    remote_write:
      - url: http://prometheus-server:9090/api/v1/write
    rule_files:

pushgateway:
  enabled: false

alertmanager:
  enabled: false

grafana:
  enabled: false

defaultRules:
  create: false
```

2. Install the Prometheus components:

```
helm install prometheus prometheus-community/prometheus \
  -n prometheus -f prometheus-agent.yaml
```

3. If necessary, expose the Prometheus port so that you can access the user interface (UI):

```
kubectl expose service kube-prometheus-prometheus \
  --type=LoadBalancer --port=9090 --target-port=9090 \
  --name=prometheus-server-ext \
  -n control-plane
```

4. Check the logs to see whether there are any errors:

```
kubectl logs -f prometheus-server-0 -n prometheus
```

5. If you have exposed the UI port (step 3), load up the UI (<http://prometheus-server:9090>) and verify that it is configured in agent mode.
6. Go to Status > Targets to check that the agent server is able to identify the targets.
7. On the Prometheus server, verify that the `web.enable-remote-write-receiver` has been enabled:

```
kubectl get -o yaml statefulset/prometheus-server -n prometheus | grep web
```

For example:

```
$ kubectl get -o yaml statefulset/prometheus-server -n prometheus | grep web
      --web.console.libraries=/etc/prometheus/console_libraries
      --web.console.templates=/etc/prometheus/consoles
      --web.enable-lifecycle
      --web.enable-remote-write-receiver
$
```

8. If you have exposed the Prometheus server UI port, run the following query to see whether the namespaces of the remote cluster appear:

```
kube_namespace_labels{namespace="my-target-cluster-namespace"}
```

Monitoring DSP: Download Grafana Dashboards

Once you have installed Grafana, download dashboards from <https://grafana.com/grafana/dashboards/> and configure them to use with your Prometheus data source.

7.3.2. Viewing Data Plane Metrics

As a system administrator, once you perform all of the setup steps in [Monitoring the Privitar Data Security Platform](#), you can use tools such as [Bitnami Prometheus](#) to view data plane metrics.

1. Determine the pods for the data plane namespace:

```
kubectl get pods -n data-plane
```

2. Install Prometheus into the same namespace:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo add stable https://kubernetes-charts.storage.googleapis.com/
helm repo update
helm install prometheus prometheus-community/prometheus -n data-plane
```

3. View the pods that are now in the data plane namespace:

```
kubectl get pods -n data-plane
```

4. Perform a port forward of the Prometheus server pod to your local machine:

```
kubectl --namespace data-plane port-forward prometheus-server-xxxx
9090:9090
```

5. In your web browser, navigate to the Prometheus user interface (UI):

```
http://localhost:9090
```

6. Run some queries on the data plane.
7. From the query bar of the Prometheus UI, click to reveal the available metrics.
8. Enter the following sample query in the query bar:

```
com_privitar_data_proxy_jdbc_handler_seconds_max{action="prepare"}
```

9. Click **Execute**.
10. Review the values on the right side of the page.

Table 52. Data Plane Metrics

Group	Metric Name	Description	Action Filter	Action Filter Description
Query Engine Database Connections	<ul style="list-style-type: none"> com_privitar_query_engine_db_connections_idle com_privitar_query_engine_db_connections_total com_privitar_query_engine_db_connections_active 	<ul style="list-style-type: none"> Number of idle connections to underlying databases Total number of connections to underlying databases Number of active connections to underlying databases 	N/A	N/A

Group	Metric Name	Description	Action Filter	Action Filter Description
Data Proxy JDBC Handler	<ul style="list-style-type: none"> com_privitar_data_proxy_jdbc_handler_seconds_max com_privitar_data_proxy_jdbc_handler_seconds_sum com_privitar_data_proxy_jdbc_handler_seconds_count 	<ul style="list-style-type: none"> The maximum number of seconds taken for a specified action on the data proxy The sum of the number of seconds taken for specified actions on the data proxy The count of specified actions on the data proxy <p>The sum divided by the count equals the average.</p>	<ul style="list-style-type: none"> prepare prepare_and_execute open_connection get_database_properties create_statement connection_sync close_statement close_connection 	<ul style="list-style-type: none"> Time taken to prepare a prepared statement (parse the query, fetch policies and metadata, and rewrite query). This event is triggered for prepared SQL statements that you typically run from a program, with variables used to replace parts of the query. Time taken to prepare a prepared statement (parse the query, fetch policies and metadata, and rewrite query) and execute it. This event is typically triggered when you are using a normal statement (such as through an interactive session where you issue a SQL query on the fly), not a prepared statement. It includes all the prepare and

Group	Metric Name	Description	Action Filter	Action Filter Description
				<p>execution time. For this metric, as it pertains to the data proxy, it also includes fetching the first 100 rows of data.</p> <p>For this metric, as it pertains to the data proxy, it also includes fetching the first 100 rows of data.</p>

Group	Metric Name	Description	Action Filter	Action Filter Description
Privitar Query Engine JDBC Handler	<ul style="list-style-type: none"> com_privitar_query_engine_jdbc_handler_seconds_max com_privitar_query_engine_jdbc_handler_seconds_sum com_privitar_query_engine_jdbc_handler_seconds_count 	<ul style="list-style-type: none"> The maximum number of seconds taken for a specified action on the data proxy The sum of the number of seconds taken for specified actions on the data proxy The count of specified actions on the data proxy <p>The sum divided by the count equals the average.</p>	<ul style="list-style-type: none"> prepare prepare_and_execute open_connection get_database_properties create_statement connection_sync close_statement close_connection 	<ul style="list-style-type: none"> Time taken to prepare a prepared statement (parse the query, fetch policies and metadata, and rewrite query). This event is triggered for prepared SQL statements that you typically run from a program, with variables used to replace parts of the query. Time taken to prepare a prepared statement (parse the query, fetch policies and metadata, and rewrite query) and execute it. This event is typically triggered when you are using a normal statement (such as through an interactive session where you issue a SQL query on the fly), not a prepared statement. It includes all the prepare and

Group	Metric Name	Description	Action Filter	Action Filter Description
				execution time. For this metric, as it pertains to the data proxy, it also includes fetching the first 100 rows of data.

7.4. Watermarking

The Privitar Data Security Platform automatically embeds a unique stamp (called a watermark) on certain types of transformed datasets. These include datasets transformed with the following tokenization types:

- [Regular expression \(regex\) number generation](#)
- [Regular expression \(regex\) text generation](#)

Because of the way the watermark is embedded, the file must have a large enough number of rows to contain the watermark. As a rule, files larger than 1,000 rows, with enough unique values, can contain watermarks.

You can use this stamp to trace the origin of a file to the project for which it was originally produced. The presence of a watermark is a good incentive to a file's recipient to make sure that they are not careless or malicious with the data, and also gives traceability in the event of a data breach or if data turns up somewhere unexpected.

Given an arbitrary file produced by the the platform, you can investigate the content to identify to which project, if any, a file belongs. Identifying the project gives access to its metadata properties, such as the file's intended purpose, who had the ability to create or access the data, and the policy under which access was granted.

7.4.1. About Watermark Investigation

The Privitar Data Security Platform (DSP) comes with a standalone watermark investigation tool.

You can use this command-line tool to investigate a tokenized dataset to determine whether the dataset has a watermark. If a watermark is present, the tool can show:

- The project ID and project name associated with the dataset
- The user ID and username (or that user's user group ID and group name) who queried the dataset through the project at the time that the platform created the result set

In order to investigate a watermark, the dataset must:

- be a .csv file
- be downloaded to a data plane where the tool is available

- have originated from a data plane in that DSP instance

7.4.2. Configure Watermark Investigation

To configure watermark investigation:

1. Ensure that the watermarking investigation tool has access to the data proxy.
2. Ensure that the data plane has connectivity to HashiCorp vault.
3. Download or create a properties .json file.

You can download an example properties file at this location:

<https://docs.privitar.com/provisioning/resources/WatermarkInvProperties.json>

Alternatively, you can create your own file using this code:

```
{
  "controlPlaneTarget": "dpp0000water.company-name.cloud:443",
  "exchangeId": "0651b9ca-eae4-4e36-b92b-04e9e459c33b",
  "dataPlaneId": "376a5c9c-dc50-4c42-9c0d-5bc841803d1f",
  "username": "testuser@acme.com",
  "controlPlaneChannelUsePlaintext": false,
  "controlPlaneChannelTrustedCertificatePath": "controlplane.crt",
  "controlPlaneChannelTlsProtocols": ["TLSv1.2"],
  "jwtSigningKeyPath": "./jwt-signing-key.pem",
  "jwtSigningKeyPassword": null,
  "jwtSigningAlgorithm": "RS256",
  "kmsProperties": {
    "properties": {
      "authenticationToken":
        "hvs.kuzTOe7a6OTDLTZtCtBoRa5P", "url": "http://10.0.0.59:8200", "path":
        "secret"},
    "extraPropertiesFilePath": null,
    "adapterJarPath": "./HashiCorpVaultKeyReader_deploy.jar",
    "sharedClassesRegex": "(org.slf4j.*)|(com.fasterxml.jackson.*)",
  },
  "csvProperties": {
    "filePath": "./watermark2__clients__20230602.csv",
    "fileContainsHeaders": true,
    "recordSeparator": "\r\n"
  }
}
```

4. For easier execution, copy the following files to the same directory of your choosing:
 - the dataset .csv file that you want to investigate
 - the properties .json file that you created (see the previous step)
 - the private key .pem file that you generated as part of [platform installation](#)
 - watermarking investigation tool .jar file (in [the data plane installer](#) at /jars/dsp_watermarking_investigation_unshaded_deploy.jar)
 - the [KMS adapter .jar](#) file (in [the data plane installer](#) at /jars/HashiCorpVaultKeyReader_deploy.jar)
 - the [control plane .crt certificate](#) (Run the following command in the control plane namespace:)

```
kubectl get secret dpp-control-plane-tls-ingress-cert -
o jsonpath='{.data.tls\.crt}' | base64 -D)
```

5. Edit the control plane section of the properties file to indicate the properties as follows:

```
{
  "controlPlaneTarget": "dpp0000water.company-name.cloud:443",
  "exchangeId": "0651b9ca-eae4-4e36-b92b-04e9e459c33b",
  "dataPlaneId": "376a5c9c-dc50-4c42-9c0d-5bc841803d1f",
  "username": "testuser@acme.com",
  "controlPlaneChannelUsePlaintext": false,
  "controlPlaneChannelTrustedCertificatePath": "controlplane.crt",
  "controlPlaneChannelTlsProtocols": ["TLSv1.2"],
  "jwtSigningKeyPath": "./jwt-signing-key.pem",
  "jwtSigningKeyPassword": null,
  "jwtSigningAlgorithm": "RS256",
}
```


To find the values for `controlPlaneTarget`, `exchangeId`, and `dataPlaneId`, download the data proxy configuration file as follows:

- a. Log in to the user interface of the platform.
- b. Click Data Exchanges in the left navigation.
- c. Click **Open Data Exchange**.
- d. Click the avatar symbol in the top right corner of the page and select **Manage Exchange**.
- e. Click the **Data Planes** tab.
- f. Click the arrow to the right of the associated data plane.
- g. Click the **Data Proxy** tab.
- h. Under Deployment Instructions, click **Download**.
The data proxy configuration file downloads.
- i. Open this JSON file.
This file includes the `dataPlaneId`, the `exchangeId`, and the `controlPlaneUrl`.
- j. Follow the instructions in the Watermarking Control Plane Properties table.

Table 53. Watermarking Control Plane Properties

Property	Description	Required?
<code>controlPlaneTarget</code>	<p>Indicates the URL of the control plane.</p> <p>Based on the data proxy configuration file, this is the <code>controlPlaneUrl</code> without the <code>https://</code> at the beginning and the port number of 443 added to the end.</p>	Yes

Property	Description	Required?
exchangeId	<p>Indicates the identifier for the associate data exchange.</p> <p>This is the exact same value as <code>exchangeId</code> in the data proxy configuration file.</p>	Yes
dataPlaneId	<p>Indicates the identifier for the associated data plane.</p> <p>This is the exact same value as <code>dataPlaneId</code> in the data proxy configuration file.</p>	Yes
username	<p>Indicates the username on the platform for the user who will perform the watermark investigation. This can be any existing user with any platform role, or you may create a new user in the platform for this purpose.</p>	Yes
controlPlaneChannelUsePlaintext	<p>Indicates whether to use plain text for communication between the control plane and the watermarking investigation tool. The default (<code>false</code>) indicates that TLS is used for communication.</p> <p>This is a property of the control plane trusted certificate. You can find this in the control plane Kubernetes log file.</p>	Yes
controlPlaneChannelTrustedCertificatePath	<p>This is the path to the certificate included in the installation bundle for the platform. See Install the Control Plane.</p> <p>Alternatively, you can fetch the path in Kubernetes, by running the following commands:</p> <pre>kubectl get secrets -n istio-system</pre> <pre>kubectl get secret <control plane certificate name> -n istio-system -o jsonpath='{.data.tls\.crt}' base64 --decode > ./certificate.crt</pre>	Yes

Property	Description	Required?
controlPlaneChannelTlsProtocols	<p>Indicates the supported TLS protocols to use for communication between the control plane and the watermarking investigation tool. The supported values are [TLSv1.2, TLSv1.3].</p> <p>You can find this in the control plane Kubernetes log file for the data proxy.</p>	Yes
jwtSigningKeyPath	<p>This is the path to the private key.</p> <div>  <p>Important</p> <p>In order to run a watermarking investigation on a data plane, you must have a private/public key pair with the public key registered on the the data plane and the private key in your local folder. Add the public key on the Data Proxy tab of the Data Planes page in the platform UI. See Create and Edit a Data Plane. Each user only needs to perform this action once per data plane.</p> </div>	Yes
jwtSigningKeyPassword	<p>This is the password for the JWT signing key, if applicable.</p>	Yes

Property	Description	Required?
jwtSigningAlgorithm	<p>Indicates the type of secure hash algorithm (SHA) used to create the JWT signing key.</p> <p>Keep the default of RS256 for RSASSA-PKCS1-v1_5 using SHA-256.</p> <p>Change to ES256 for ECDSA using P-256 (aka secp256r1 or prime256v1) and SHA-256.</p> <p>These are the only two SHA types that the platform currently supports.</p>	Yes

6. Edit the kmsProperties section (example below) of the properties file to indicate the properties in the table below.

```
"kmsProperties": {
  "properties": { "authenticationToken":
"hvs.kuzTOe7a60TDLTZtCtBoRa5P", "url": "http://10.0.0.59:8200", "path":
"secret"},
  "extraPropertiesFilePath": null,
  "adapterJarPath":  "./HashiCorpVaultKeyReader_deploy.jar",
  "sharedClassesRegex": "(org.slf4j.*)|(com.fasterxml.jackson.*)"
},
```

Set `kubernetesRole` and `kubernetesAuthPath` to the same value in `data-plane-values.yaml` used when you installed the platform. See [Install the Data Plane](#).

Table 54. KMS Properties

Property	Description	Required?	Example
authenticationToken	This is the value of the authentication token.	Yes	<code>hvs.kuzTOe7a60TDLTZtCtBoRa5P</code>
url	The URL to the HashiCorp Vault server, including the protocol.	Yes	http://vault:8200
path	The path in HashiCorp Vault used as the base path for all secrets to be read.	Yes	<code>secret</code>

Property	Description	Required?	Example
<code>extraPropertiesFilePath</code>	If not null, then read the file at this path in Java properties file format and merge it with the Properties field.	No	
<code>adapterJarPath</code>	If not null, then read the adapter.jar file from this path. If null, then the adapter.jar file must be on the JVM classpath.	Yes	<code>./HashiCorpVaultKeyReader_deploy</code>
<code>sharedClassesRegex</code>	Loaded classes and resources that match this regex will be shared between the adapter (plugin) and the rest of the application. If null, will default to sharing SLF4J and Jackson classes.	Yes	<code>(org.slf4j.*) (com.fasterxml.jackson.*)</code>

7. Edit the `csvProperties` section of the properties file to indicate the following about the .csv file:


```
"csvProperties": {
  "filePath": "./watermark2__clients__20230602.csv",
  "fileContainsHeaders": true,
  "recordSeparator": "\r\n"
}
```



Note

You only need to include optional properties that you are changing from the default. If you are using the default value, you can exclude that property from the properties file.

Table 55. Watermarking .csv Properties

Property	Description	Required?	Example
filePath	Indicates the path to the .csv file that you want to investigate.	Yes	./ 2022_12_31_retail_customer
fileContainsHeaders	Indicates whether the .csv file has a header row.	No	false
delimiter	Indicates the character used to mark the division between fields. Change this if your file uses a different delimiter character, such as a pipe (). Note that this cannot be a line break character.	No	
quoteCharacter	Change this if your file uses a different quote character, such as a single quote (').	No	'
escape	<p>Indicates the character used when there are delimiter or quote characters embedded in the data. To disable this setting, enter <code>null</code>.</p> <div>  Note Be sure to use the escape character before entering your escape character. </div>	No	
recordSeparator	Indicates the line feed. This varies by operating system. For Linux use the default (<code>\n</code>). For Windows, use <code>\r\n</code> .	No	<code>\r\n</code> .
charSet	Indicates which character set encoding the file uses. Change this if your file uses a different character set, such as ASCII.	No	ASCII.

You are now ready to [Investigate a Watermark](#).

7.4.3. Investigate a Watermark

As a system administrator, you can investigate a watermark.

To investigate a watermark:

1. Locate the file that you want to investigate.

**Note**

This file must be local to a data plane that has access to the tool, and it must be in .csv format.

2. Locate the properties JSON file that you configured (see [Configure Watermark Investigation](#)).
3. Run the watermark investigation tool, adding the properties JSON file as argument. For example:

```
java -jar dsp_watermarking_investigation_shaded.jar ./path/to/properties.json
```

The tool prompts you for a password.

4. Enter the password associated with the username in the configuration file.

The tool investigates the file.

If it finds a watermark, it returns the following:

- The project ID and project name associated with the dataset
- The user ID and username (or that user's user group ID and group name) who queried the dataset through the project at the time that the platform created the result set

If it does not find a watermark, it returns `No watermark was found`.

If the configuration file has an incorrect property, the tool returns the error message `Something went wrong reading the file`, followed by a description of the error. For example, if the delimiter in the properties file does not match the delimiter in the .csv file, the tool returns the error message `(line 1) invalid char between encapsulated token and delimiter`.

8. Integrations

You may already have a diverse and wide-ranging set of tools in your existing environments, for example, data catalogs, lineage tracking mechanisms, data discovery, or data virtualization tools. This section provides information on how the Privitar Data Security Platform (DSP) integrates with third-party tools and platforms.

8.1. Integration of Denodo Data Virtualization

To request data from the Privitar Data Security Platform (DSP) you need to configure a connection in Denodo. You must:

- Configure the data proxy to work in transparent mode (see [Configure the control-plane-values File](#)).

**Note**

This requires system administrator privileges.

- Install the DSP data proxy driver in Denodo.
- Set up a base view.
- Launch a query, either from within Denodo or with a business intelligence (BI) tool.

You need access to the platform, the data proxy, and Denodo Design Studio.

You need an approved project in the platform from which you can extract the URL of the data proxy driver. You must specify a proxy driver in Denodo in order to make requests to the platform's underlying database (in the data plane) and have the results passed back to Denodo. Denodo requests pass through the proxy and therefore apply all the policies that apply to that request.

To configure the connection in Denodo, you need the URL to the data plane, which contains both the connection ID and the project ID. Within the platform, you find the URL in the data exchange.

You must also provide the user ID and password of someone who can access the project.

**Note**

The Denodo configuration contains only a connection ID and the proxy ID. Although it uses a user ID to log in to the platform, user information is not passed from the platform back to Denodo.

8.1.1. Configure the Proxy Driver Connection in Denodo

To allow Denodo to communicate with the platform's data proxy, install the platform's driver into Denodo.

The Denodo driver and the platform JDBC driver may have clashing instances of the SL4J library. To mitigate this, follow the steps below.

1. Unzip the dynamic driver using these commands to unzip the JDBC JAR, remove Privitar's instance of the SL4J libraries, and reconstitute the JAR file:
 - a. `mkdir jdbc-modifications`
 - b. `unzip data_proxy_driver-[version].jar -d jdbc-modifications`
 - c. `rm -r jdbc-modifications/org/slf4j`
 - d. `cd jdbc-modifications`
 - e. `jar cf ../data_proxy_driver-[version]-modified.jar *`
2. Copy the modified JAR file into Denodo's installation directory to register the DSP Platform as a data source:
 - a. `mkdir [denodo platform installation directory]/lib/extensions/jdbc-drivers-other/privitar`
 - b. `cp data_proxy_driver-[version]-modified.jar [denodo platform installation directory]/lib/extensions/jdbc-drivers-other/privitar/`

You may need to reboot the VirtualData Port server/daemon.

8.1.2. Set Up a Denodo Base View and Run a Query

Denodo base views provide [virtual query language](#) (VQL) overlays to raw data. You can then use base views from multiple data sources (potentially from different storage technologies) to create "virtualized," derived views. End users can then query Denodo to access data from derived or base views using VQL, without any regard for the underlying storage technology hosting the data.

To set up a Denodo base view:

1. Open Denodo Design Studio and log in.
2. If your control plane has self-signed certificates (the default for self-managed installations), you must create a trust store to allow Denodo to connect. Pull your data plane certificate from Kubernetes using the following `kubectl` command:

```
kubectl get secret dpp-data-plane-tls-ingress-cert -n istio-system -o
jsonpath='{.data.tls\.cert}' | base64 --decode > ~/workspace/airbenders-
deployment/dpp-data-plane-[version]/dataplane.crt
```

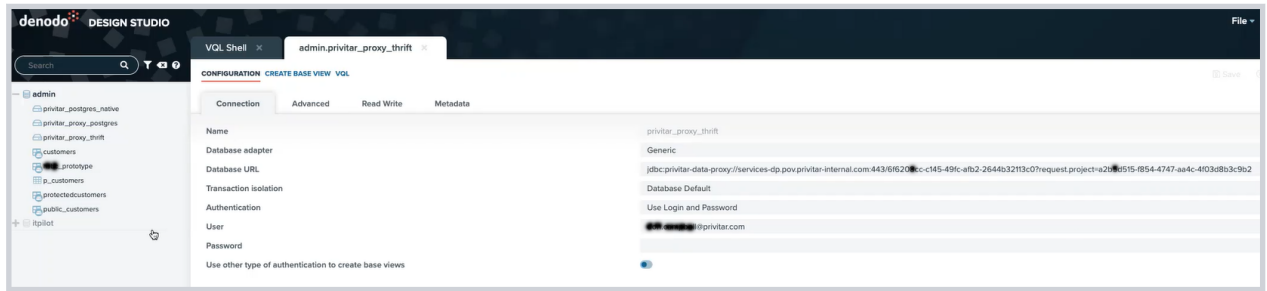
3. Create a trust store with key tool:

```
keytool -import -file dataplane.crt -keystore truststore-local.jks
```

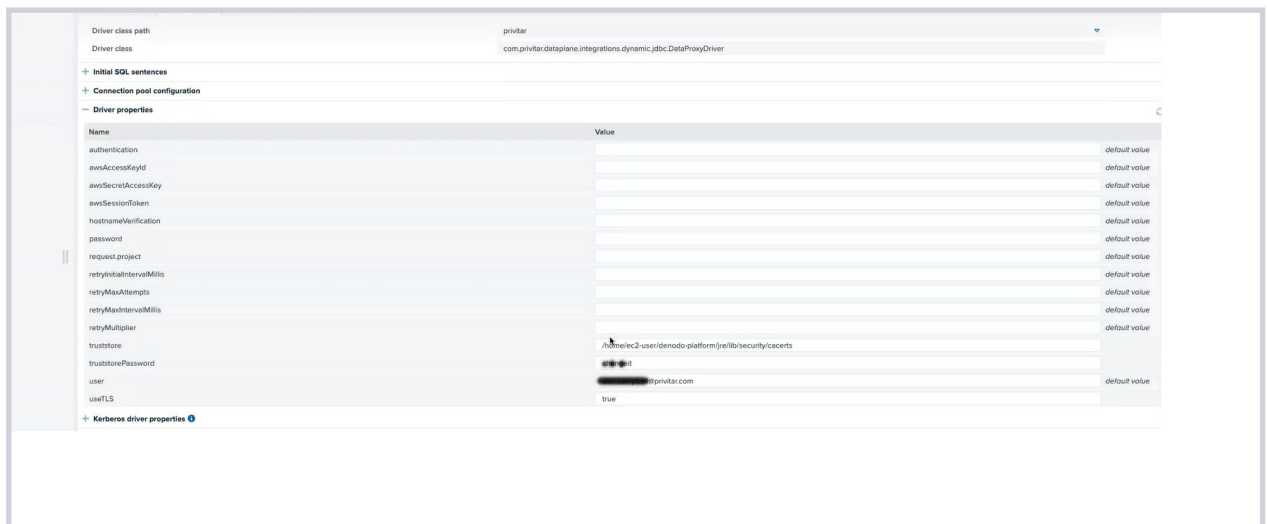
4. Set a trust store password, and answer **yes** when prompted to trust the certificate.
5. Click **File** → **New** → **Data Source** → **JDBC**

The Denodo instance above already has the DSP platform driver installed (see [Configure the Proxy Driver Connection in Denodo](#)).

6. **Database adapter**—Select **Google BigQuery**, **Hive 3.1.2 (HiveServer2)**, or **Spark SQL 2.x**.



7. **Database URL**—Enter the JDBC string provided by the platform control plane.
8. **User**—Enter a platform username.
9. **Password**—Enter a platform password.
10. On the **Advanced** tab, select *Privitar* from the **Driver class path** configuration in the **Other JDBC drivers** section.
11. **Driver class**—Enter `com.privitar.dataplane.integrations.dynamic.jdbc.DataProxyDriver`.
12. In **Driver properties**, set the following:



- **truststore**—absolute path to the trust store . jks file created above
 - **truststorePassword**—the password you set for the trust store when creating it
 - **useTLS**—set to *true*
13. On the Source Configuration tab, set **Allow literal as parameter** to *no*.
 14. Click **Save** to save your view., and test to make sure you can connect.
 15. Click **Test connection** to ensure that you can connect.

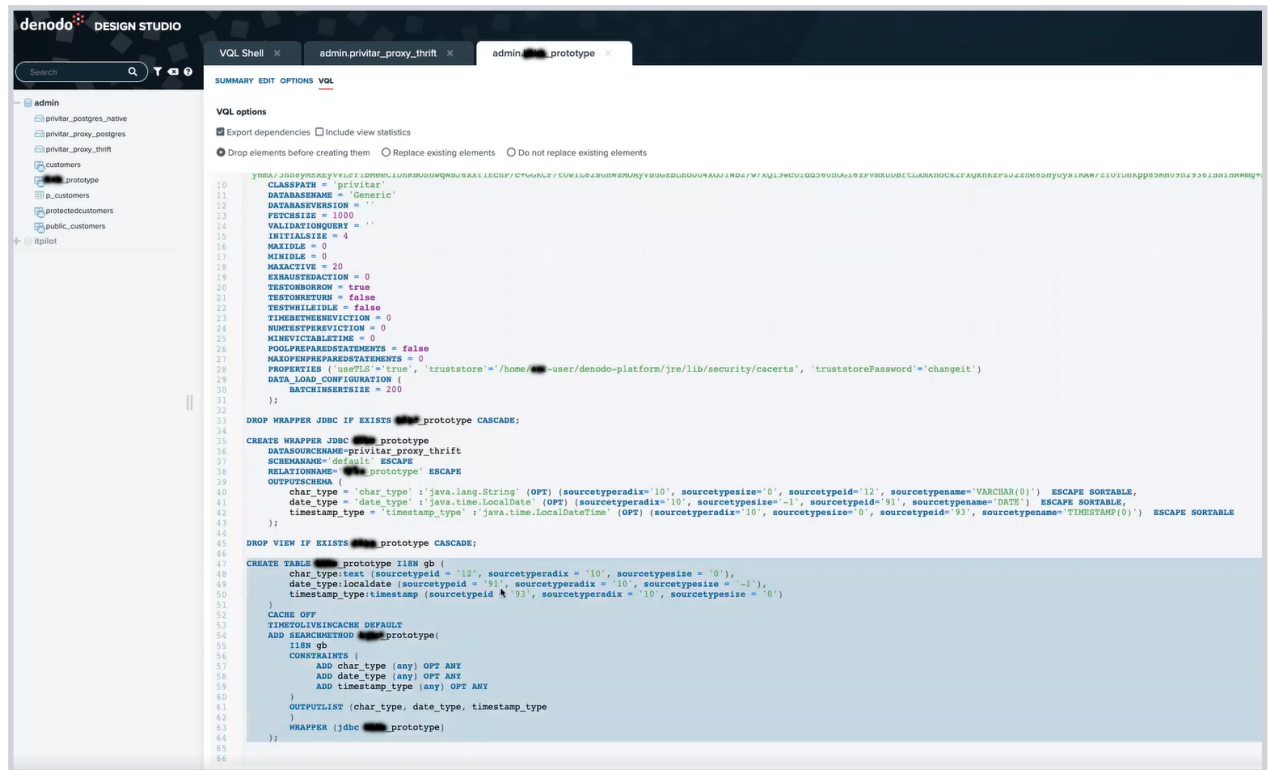


Warning

Our testing revealed occasional false negatives when using **Test connection**. If a test fails with the error "Driver not found," attempt to connect to the database to double-check the connection.

16. Click **Create Base View**.
17. Navigate through your database hierarchy until you come to the table from which you would like to create your base view.

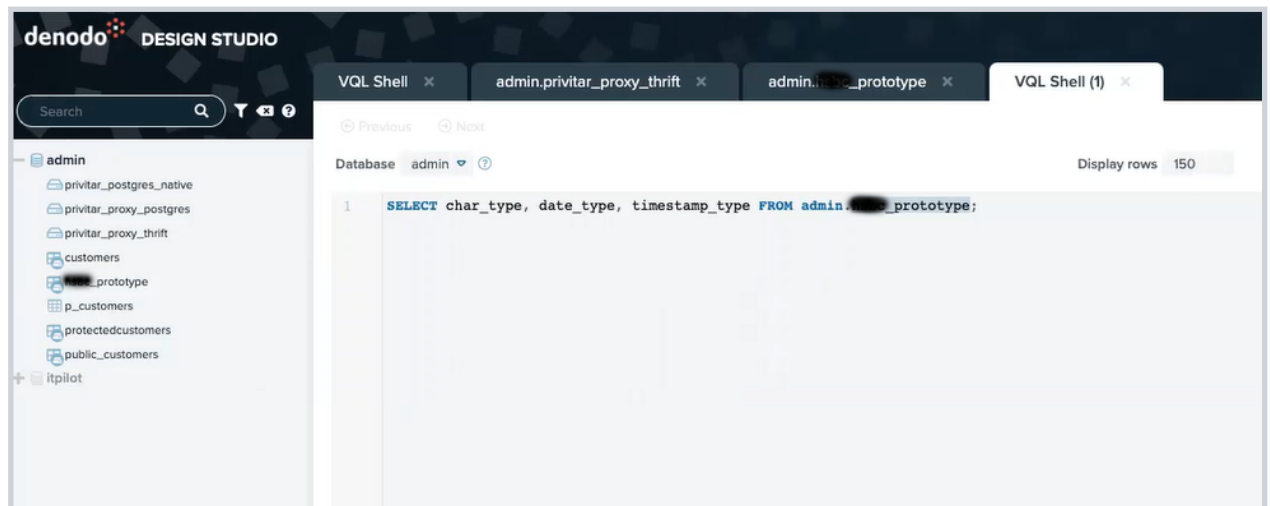
Check the table, and click **Create Selected**.



If you want to create the base view with a prefix in the name (for example, `protected_`), enter the prefix in the **Preview view names with** box before you click **Create Selected**.

18. Create a table in Denodo.

The table feature lets you move data from its source into a target database table. Use the table to run a query in the platform database.



8.1.3. Query Denodo from a Business Intelligence Tool

You can run queries from within Denodo or from a Business Intelligence (BI) tool that talks to Denodo.

To query Denodo from a BI tool:

1. Download the Denodo JDBC driver from Denodo's website.
For DBeaver, this is not required, as DBeaver comes bundled with it.
2. Connect to your Denodo instance with the following JDBC endpoint:

```
jdbc:denodo://[VirtualData Port Server Hostname]:[VirtualData Port
Server Port]/[database]
```

3. For the Denodo instance above, you can connect with: `jdbc:denodo://172.31.1.107:9999/admin`

8.1.4. Data Sources Supported with Denodo

The platform supports the following data sources with Denodo out of the box:

- [Apache Hive](#)
- [Apache Spark](#)
- [Google BigQuery](#)
- [PrestoSQL \(Trino\)](#)

8.2. Privitar Query Engine Properties

The Privitar Query Engine is designed to work within Privitar's data proxy. If you prefer to create your own application or use the Query Engine in some other way, here are the Privitar Query Engine properties that you will need to reference as you configure your data proxy.

8.2.1. Privitar Query Engine Example Implementation

The following is an example implementation of the Privitar Query Engine:

```
package com.privitar.dataplane.services.dynamic.jdbc.queryengine;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
/**
 * To run this put the query engine jar on the classpath, plus the jars of
 * all JDBC drivers required
 * to connect to underlying databases.
 */
public class QueryEngineExample {
    public static void main(String[] args) throws SQLException {
        Properties properties = new Properties();
        properties.put("remoteUser", "user.name");
        properties.put("remoteUserGroups", "group1,group2,group3");
        properties.put("remoteUserAreIdsExternal", "true");
        properties.put("controlPlaneTarget",
"your.controlplane.acme.com:443");
        properties.put("controlPlaneChannelTrustedCertificatePath", "/your/
path/config/cp-trusted-cert.crt");
```

```

        properties.put("controlPlaneChannelClientAuthJwtSigningKeyPath", "/" +
your/path/config/jwt-key");
        properties.put("dataPlaneId", "3cba2fde-aeaa-4fc5-8278-
dle95fd2efa4");
        properties.put("exchangeId", "6645b90f-b9bc-4ebe-b98d-c7eaa2ab89f7");
        properties.put("request.project", "38fd29ab-40c2-498c-a8b5-
ad3fad0c2b0a");
        String connectionId = "9b6c4577-b7cb-475f-98ce-9ed2a01a8d07";
        properties.put("datasource", connectionId);
        try (Connection conn = DriverManager.getConnection("jdbc:privitar-
query-engine:", properties);
            Statement statement = conn.createStatement();
            ResultSet resultSet = statement.executeQuery("select * from
bank_accounts limit 10")) {
            ResultSetMetaData metaData = resultSet.getMetaData();
            int columnCount = metaData.getColumnCount();
            StringBuilder sb = new StringBuilder();
            for (int i = 1; i <= columnCount; i++) {
                sb.append(metaData.getColumnName(i)).append(" | ");
            }
            sb.append("\n");
            while (resultSet.next()) {
                for (int i = 1; i <= columnCount; i++) {
                    sb.append(resultSet.getObject(i)).append(" | ");
                }
                sb.append("\n");
            }
            System.out.println(sb);
        }
    }
}

```

8.2.2. Privitar Query Engine JDBC Configuration Information

Classname: com.privitar.dataplane.services.dynamic.jdbc.queryengine

JDBC connection string: jdbc:privitar-query-engine:

All other configuration should be passed in the properties described in the following sections.

Any JDBC drivers required to connect to the datasources should be available as part of the classpath.

8.2.3. Privitar Query Engine JDBC Connection Properties

datasource

- **Required:** Yes
- **Accepted Values:** *String*
- **Example Value:** b27874c2-658f-4210-b7c6-3b3134de62d2
- **Description:** The ID of the datasource that will be queried via the Query Engine. The property **remoteUser** must be provided if no requesting user has been set on the QueryEngineDataSource object.

remoteUser

- **Required:** Yes
- **Accepted Values:** *String*
- **Example Value:** u123
- **Description:** The ID of the user that will be querying the underlying data source through the Privitar Query Engine.

remoteUserGroups

- **Required:** No
- **Accepted Values:** *String*
- **Example Values:**
 - If `remoteUserAreIdsExternal=FALSE`, an example value is `group1,group2,group3`.
 - If `remoteUserAreIdsExternal=TRUE`, an example value is `8d7d6732-a89b-11ed-afa1-0242ac120002,d7d6c1e-a89b-11ed-afa1-0242ac120002,8d7d6dae-a89b-11ed-afa1-0242ac120002`.
- **Description:** The group IDs to which the user who will be querying the underlying data source through the Privitar Query Engine belongs.

remoteUserAreIdsExternal

- **Required:** No
- **Accepted Values:** True, False
- **Default Value:** FALSE
- **Description:** Whether or not the remote user ID and group IDs are external.

secretManagerConfigPath

- **Required:** No
- **Accepted Values:** *String*
- **Example Value:** `file:///Users/me/secret-manager.json`
- **Description:** Path to a JSON file that contains the configuration for the secret manager used in connections. By configuring this, the query engine will expect that all usernames and passwords for the connections are passed as a reference. The format of this reference is adapter specific, but for HashiCorp® Vault it is passed in the form of `{version}|{name}` e.g. `0|postgres_username`.

The JSON file should conform to the following schema:

```
{
  "$schema" : "http://json-schema.org/draft/2019-09/schema#",
  "title" : "CUSTOM",
  "type" : "object",
  "additionalProperties" : false,
  "properties" : {
    "type" : {
```



```

    "type" : "string",
    "enum" : [ "CUSTOM" ],
    "default" : "CUSTOM"
  },
  "extraPropertiesFilePath" : {
    "type" : "string",
    "description" : "If not null then read the file at this path in Java
Properties file format and merge it with the properties field"
  },
  "adapterJarPath" : {
    "type" : "string",
    "description" : "If not null then read the adapter jar file from
this path. If null then the adapter jar must be on the JVM classpath"
  },
  "sharedClassesRegex" : {
    "type" : "string",
    "description" : "Loaded classes/resources which match this regex
will be shared between the adapter/plugin and the rest of the application.
If null, will default to sharing SLF4J and Jackson classes."
  },
  "properties" : {
    "type" : "object",
    "additionalProperties" : {
      "$ref" : "#/definitions/Object"
    },
    "description" : "Arbitrary set of key-value pairs to configure the
KMS"
  }
},
"required" : [ "type" ],
"definitions" : {
  "Object" : {
    "type" : "object",
    "additionalProperties" : false,
    "properties" : { }
  }
}
}

```

Properties for configuration of the key management system (KMS) are specific to the adapter used. For HashiCorp® Vault KMS the properties are as follows:

Table 56. KMS Properties

Property	Description	Required?	Example
url	The URL to the HashiCorp Vault server, including the protocol.	Yes	http://vault:8200

Property	Description	Required?	Example
authenticationToken	If authenticationMethod is set to TOKEN, then this is the value of the authentication token. Required only if authenticationMethod is set to TOKEN.	Variable	hvs.kuzTOe7a6OTDLTZtCtBoRa5E
path	The path in HashiCorp Vault used as the base path for all secrets to be read.	Yes	secret
authenticationMethod	The authentication method used to authenticate with HashiCorp Vault.	No	TOKEN, KUBERNETES
kubernetesAuthPath	If authenticationMethod is set to KUBERNETES, then this is the path where the Kubernetes auth method was installed. The default is to use "kubernetes," that is authenticate with auth/kubernetes/login.	No	
kubernetesRole	If authenticationMethod is set to KUBERNETES, then this is the name of the Kubernetes role to be used for auth. Required only if authenticationMethod is set to KUBERNETES.	Variable	

See [Deploy HashiCorp Vault as KMS or Secrets Manager](#) for more information on how to configure HashiCorp® Vault KMS with the platform.

kmsConnectionConfigRepositoryPath

- **Required:** No
- **Accepted Values:** *String*
- **Example Value:** file:///Users/me/kms.json
- **Description:** Path to a JSON file which contains the configuration for all the KMS available to the Query Engine. Only required if using NOVLT.

The JSON file should conform to the following schema:

```
{
  "$schema" : "http://json-schema.org/draft/2019-09/schema#",

```

```

"title" : "CUSTOM",
"type" : "object",
"additionalProperties" : false,
"properties" : {
  "type" : {
    "type" : "string",
    "enum" : [ "CUSTOM" ],
    "default" : "CUSTOM"
  },
  "extraPropertiesFilePath" : {
    "type" : "string",
    "description" : "If not null then read the file at this path in Java
Properties file format and merge it with the properties field"
  },
  "vaultlessKeyIdentifier" : {
    "type" : "string",
    "description" : "The key identifier for NOVLIT"
  },
  "watermarkingKeyIdentifier" : {
    "type" : "string",
    "description" : "The key identifier for watermarking"
  },
  "adapterJarPath" : {
    "type" : "string",
    "description" : "If not null then read the adapter jar file from
this path. If null then the adapter jar must be on the JVM classpath"
  },
  "sharedClassesRegex" : {
    "type" : "string",
    "description" : "Loaded classes/resources which match this regex
will be shared between the adapter/plugin and the rest of the application.
If null, will default to sharing SLF4J and Jackson classes."
  },
  "properties" : {
    "type" : "object",
    "additionalProperties" : {
      "$ref" : "#/definitions/Object"
    },
    "description" : "Arbitrary set of key-value pairs to configure the
KMS"
  }
},
"required" : [ "type" ],
"definitions" : {
  "Object" : {
    "type" : "object",
    "additionalProperties" : false,
    "properties" : { }
  }
}
}

```

Properties for configuration of the key management system (KMS) are specific to the adapter used. For HashiCorp® Vault KMS the properties are as follows:

Table 57. KMS Properties

Property	Description	Required?	Example
url	The URL to the HashiCorp Vault server, including the protocol.	Yes	http://vault:8200
authenticationToken	If authenticationMethod is set to TOKEN, then this is the value of the authentication token. Required only if authenticationMethod is set to TOKEN.	Variable	hvs.kuzTOe7a6OTDLTZtCtBoRa5E
path	The path in HashiCorp Vault used as the base path for all secrets to be read.	Yes	secret
authenticationMethod	The authentication method used to authenticate with HashiCorp Vault.	No	TOKEN, KUBERNETES
kubernetesAuthPath	If authenticationMethod is set to KUBERNETES, then this is the path where the Kubernetes auth method was installed. The default is to use "kubernetes," that is authenticate with auth/kubernetes/login.	No	
kubernetesRole	If authenticationMethod is set to KUBERNETES, then this is the name of the Kubernetes role to be used for auth. Required only if authenticationMethod is set to KUBERNETES.	Variable	

See [Deploy HashiCorp Vault as KMS or Secrets Manager](#) for more information on how to configure HashiCorp® Vault KMS with the platform.

controlPlaneTarget

- **Required:** Yes
- **Accepted Values:** *String*
- **Example Value:** localhost:8081

- **Description:** The URL of the data bridge. The Privitar Query Engine will set up a gRPC channel between itself and the data bridge in order to be able to fetch information from the control plane.

exchangeId

- **Required:** No
- **Accepted Values:** *String*
- **Example Value:** b27874c2-658f-4210-b7c6-3b3134de62d2
- **Description:** The UUID of the data exchange of which users of the Privitar Query Engine are members. The Privitar Query Engine will fetch the connection details of the data source and fetch policies setup in this exchange.

dataPlaneId

- **Required:** Yes
- **Accepted Values:** *String*
- **Example Value:** b27874c2-658f-4210-b7c6-3b3134de62d2
- **Description:** The UUID of the data plane that will be able to connect to the underlying data source.

policyExpiryInMilliseconds

- **Required:** No
- **Accepted Values:** *String*
- **Description:** The TTL for the applicable policy and connection details fetched from the control plane. Use -1 to disable caching of policies and connection details.

defaultFetchSize

- **Required:** No
- **Accepted Values:** *String*
- **Description:** The default number of rows that the Privitar Query Engine will fetch from the underlying database at a time. Setting this property will overwrite the default fetch size of the underlying JDBC driver that connects to the database.

defaultStatementQueryTimeoutSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 0
- **Description:** Number of seconds that the Privitar Query Engine will wait for the JDBC statement to execute. Setting this property to 0 will indicate that there is no time limit.

controlPlaneChannelUsePlaintext

- **Required:** No
- **Accepted Values:** *String*

- **Default Value:** FALSE
- **Description:** Whether the gRPC channel to the data bridge should be over plaintext.

controlPlaneChannelKeepAliveTimeSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 45
- **Description:** The time without read activity on the gRPC channel before the Privitar Query Engine sends a keepalive ping. Setting this option to a non-positive value disables the feature.

controlPlaneChannelTrustedCertificatePath

- **Required:** No
- **Accepted Values:** *String*
- **Description:** The path of the optional server TLS certificate that the Privitar Query Engine should trust before it begins communication with the data bridge. The file should contain an X.509 certificate collection in PEM format.



Tip

You may be able to retrieve this by running in the control plane namespace the following:

```
kubectl get secret dpp-control-plane-tls-ingress-cert -  
o jsonpath='{.data.tls\.crt}' | base64 -D
```

controlPlaneChannelClientAuthMode

- **Required:** No
- **Accepted Values:** JWT_BEARER_TOKEN, NONE
- **Default Value:** JWT_BEARER_TOKEN
- **Description:** The mechanism used by the Privitar Query Engine to authenticate itself to the data bridge.

controlPlaneChannelClientAuthJwtSigningAlgorithm

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** RS256
- **Description:** The JWT signing algorithm used (see RFC 7518) when controlPlaneChannelClientAuthMode is JWT_BEARER_TOKEN.

controlPlaneChannelClientAuthJwtExpirationSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 60
- **Description:** The JWT expiration period in seconds when `controlPlaneChannelClientAuthMode` is `JWT_BEARER_TOKEN`.

controlPlaneChannelClientAuthJwtSigningKeyPath

- **Required:** No
- **Accepted Values:** *String*
- **Description:** The path of the JWT signing key when `controlPlaneChannelClientAuthMode` is `JWT_BEARER_TOKEN`. This must be a PEM-encoded representation of a PKCS#8 key using the algorithm specified in `controlPlaneChannelClientAuthJwtSigningAlgorithm`.



Tip

You may be able to retrieve the signing key by running the following in the data plane namespace:

```
kubectl get secret data-agent-client-auth-secret
-o jsonpath='{.data.clientauthkey}' | base64 -D
```

controlPlaneChannelClientAuthJwtSigningKeyPassword

- **Required:** No
- **Accepted Values:** *String*
- **Description:** The password used to decrypt the private key (if it is encrypted) when `controlPlaneChannelClientAuthMode` is `JWT_BEARER_TOKEN`.

controlPlaneChannelTlsProtocols

- **Required:** No
- **Accepted Values:** *String*
- **Example Value:** `TLSv1.2,TLSv1.3`
- **Default Value:** `TLSv1.2`
- **Description:** A comma-separated list of TLS protocols that should be supported in the gRPC channel to control plane.

controlPlaneResolvedPoliciesRetryMaxAttempts

- **Required:** No
- **Accepted Values:** *String*

- **Default Value:** 5
- **Description:** Maximum number of attempts made by the Query Engine to fetch the applicable policies from the control plane.

controlPlaneResolvedPoliciesInitialBackoffSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 1
- **Description:** The number of seconds to wait before making another attempt at fetching applicable policies from the control plane, if the first attempt failed.

controlPlaneResolvedPoliciesBackoffMultiplier

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 3
- **Description:** Multiplier that increases the number of seconds to wait before making another attempt at fetching applicable policies from the control plane, if the previous attempt failed.

controlPlaneResolvedPoliciesMaxBackoffSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 10
- **Description:** The maximum number of seconds to wait before making another attempt at fetching applicable policies from the control plane, if the previous attempt failed.

controlPlaneResolvedPoliciesTimeoutSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 45
- **Description:** The number of seconds to wait for a response from the control plane when fetching applicable policies.

controlPlaneConnectionDetailsRetryMaxAttempts

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 5
- **Description:** Maximum number of attempts made by the Privitar Query Engine to fetch the connection details of the data source from the control plane.

controlPlaneConnectionDetailsInitialBackoffSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 1
- **Description:** The number of seconds to wait before making another attempt at fetching the connection details of the datasource from the control plane if the first attempt failed.

controlPlaneConnectionDetailsBackoffMultiplier

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 3
- **Description:** Multiplier that increases the number of seconds to wait before making another attempt at fetching the connection details of the datasource from the control plane if the previous attempt failed.

controlPlaneConnectionDetailsMaxBackoffSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 10
- **Description:** The maximum number of seconds to wait before making another attempt at fetching the connection details of the datasource from the control plane if the previous attempt failed.

controlPlaneConnectionDetailsTimeoutSeconds

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** 45
- **Description:** The number of seconds to wait for a response from the control plane when fetching the connection details of the datasource.

verboseAuditLog

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** FALSE
- **Description:** Whether the audit logs regarding the query and available policies should be verbose.

bypassPolicy

- **Required:** No
- **Accepted Values:** *String*

- **Default Value:** FALSE
- **Description:** Whether the Privitar Query Engine policy enforcement should be bypassed.

auditEnabled

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** FALSE
- **Description:** Whether to enable auditing.

auditStdout

- **Required:** No
- **Accepted Values:** *String*
- **Default Value:** FALSE
- **Description:** Whether to log audit records to standard output (STDOUT).

auditServiceUrl

- **Required:** No
- **Accepted Values:** *String*
- **Description:** The audit service URL to emit log records to.

request.project

- **Required:** Yes
- **Accepted Values:** *String*
- **Description:** The ID of the project set up in the data exchange.

request.*

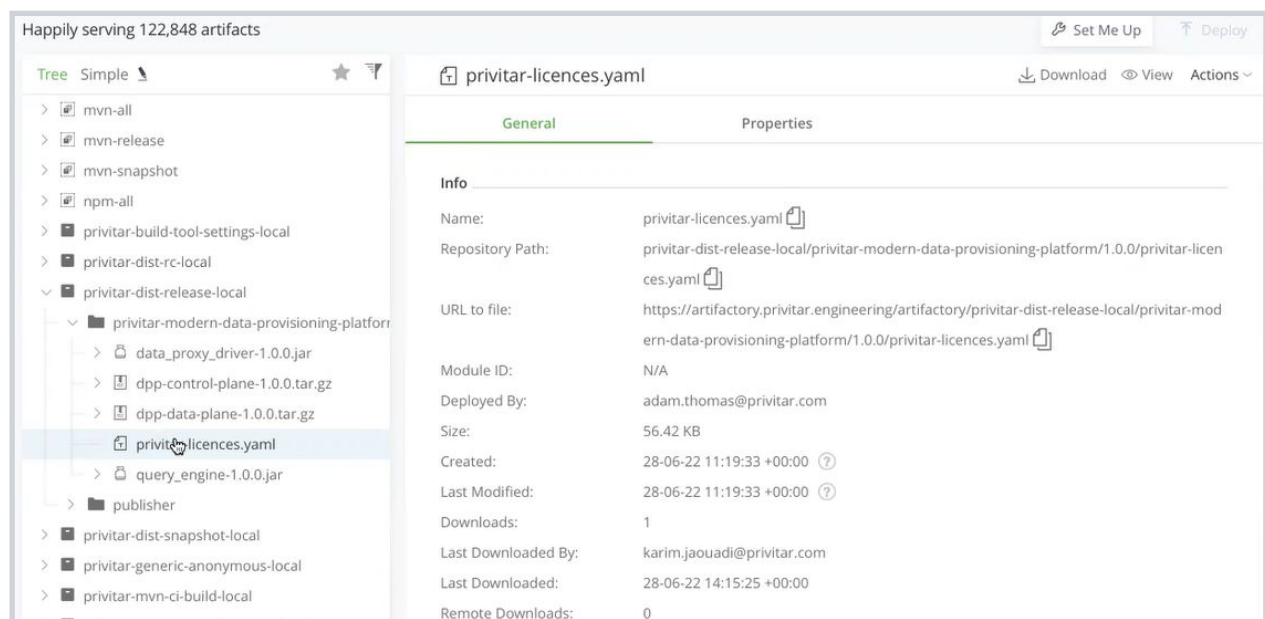
- **Required:** No
- **Accepted Values:** *String*
- **Description:** Any property starting with `request.` will be used as a custom user defined attribute to resolve policies. `request.project` is handled specially, see the description of that property. The ID of the project set up in the data exchange.

9. Third-Party Licensing Information

Your platform installation files include a YAML file that lists all third-party applications with their version numbers and licensing information. You will find `privitar-licenses.yaml` in the installation repository at:

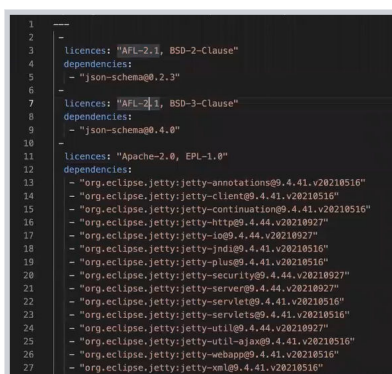
```
privitar-dist-release-local/privitar-modern-data-provisioning-platform/
[release number]/
```

This resource helps procurement and legal teams determine whether your installation complies with third-party licenses.



In the same folder, you can see the zipped files for the data plane, control plane, and data proxy driver. Download the `privitar-licenses.yaml` file, and open it with your preferred editor.

The file is very long. It contains license information and all dependencies.



10. Administration Troubleshooting

The following may help you troubleshoot issues that might arise during administration of the platform:

- [Privitar Support](#)
- [Reconnect a Node](#)

10.1. Privitar Support

Best Practices for Contacting Privitar Support

If you have any issues using any of our products, the Privitar Support team is here to help. In order to help us resolve your issue as quickly as possible, please take the following steps:

1. Check [the knowledge base](#) and [documentation](#).
2. Take note of the following:
 - A brief description of the issue
 - Any error messages associated with the problem
 - How frequently the issue occurs
 - Whether the problem is consistently reproducible
 - The steps that we could follow to reproduce the issue
 - Any troubleshooting that you have already undertaken
 - The version of the Privitar product
 - The versions of the components in your environment (operating system, databases, load balancers, service mesh, and so on)
 - Any recent changes to your environment, such as a new database
3. Collect any relevant screenshots, log files, and trace logs (when applicable).
4. Outline the business impact.
5. Go to <https://support.privitar.com> to log a ticket or reach out to support@privitar.com.

10.2. Reconnect a Node

Manually Reconnect a Node

You might need to manually reconnect a node if your organization uses the following configuration (typically in a non-production deployment of the platform):

- [Keycloak](#) for access management
- [PostgreSQL](#) high-availability (HA) deployment for a database
- [Pgpool](#) for load balancing
- [repmgr](#) for replication and failover

In the unlikely event that a primary node fails, repmgr will promote a standby instance to be the primary node. When the original primary node recovers, repmgr marks it as inactive, and Pgpool excludes this node from load balancing.

Example 1. See repmgr Status

To see the current status of repmgr within the cluster, run this command:

```
kubectl -n <namespace> exec -it postgresql-ha-postgresql-1 -- bash -c
"/opt/bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr -f /opt/bitnami/
repmgr/conf/repmgr.conf cluster show"
```

Example 2. Example repmgr Status

The output will look similar to the following:

```
Dev kubectl -n test1 exec -it postgresql-ha-postgresql-1 -- bash -c "/opt/
bitnami/scripts/postgresql-repmgr/entrypoint.sh repmgr -f /opt/bitnami/
repmgr/conf/repmgr.conf cluster show"
postgresql-repmgr 10:15:38.87
postgresql-repmgr 10:15:38.88 Welcome to the Bitnami postgresql-repmgr
container
postgresql-repmgr 10:15:38.88 Subscribe to project updates by watching
https://github.com/bitnami/bitnami-docker-postgresql-repmgr
postgresql-repmgr 10:15:38.88 Submit issues and feature requests at https://
github.com/bitnami/bitnami-docker-postgresql-repmgr/issues
postgresql-repmgr 10:15:38.88
postgresql-repmgr 10:15:38.89 DEBUG ==> Configuring libnss_wrapper...

ID      | Name                                | Role      | Status      |
Upstream                                | Location  | Priority   | Timeline    | Connection
string

-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
-----

1000 | postgresql-ha-postgresql-0 | primary | - failed
| ?                                | default | 100        |          | user=repmgr
password=Xz6ta4ncsJ host=postgresql-ha-postgresql-0.postgresql-ha-postgresql-
headless.test1.svc.cluster.local dbname=repmgr port=5432 connect_timeout=5
1001 | postgresql-ha-postgresql-1 | primary | * running
|                                | default | 100        | 2         | user=repmgr
password=Xz6ta4ncsJ host=postgresql-ha-postgresql-1.postgresql-ha-postgresql-
headless.test1.svc.cluster.local dbname=repmgr port=5432 connect_timeout=5
1002 | postgresql-ha-postgresql-2 | standby | running | postgresql-ha-
postgresql-1 | default | 100        | 1         | user=repmgr
password=Xz6ta4ncsJ host=postgresql-ha-postgresql-2.postgresql-ha-postgresql-
headless.test1.svc.cluster.local dbname=repmgr port=5432 connect_timeout=5

WARNING: following issues were detected
- unable to connect to node "postgresql-ha-postgresql-0" (ID: 1000)

HINT: execute with --verbose option to see connection error messages
command terminated with exit code 25
```

In this example, node postgresql-ha-postgresql-0 is unavailable.

Example 3. Reconnect an Unavailable Node

If a primary node becomes unavailable, run the following command against that node to manually reconnect it as a standby node to the cluster:

```
kubectl -n <namespace> exec -it postgresql-ha-postgresql-1 -- bash -c "rm  
-fr /bitnami/postgresql/data/"
```

11. Glossary of Data Security Terminology

This glossary defines terms that relate to the Privitar Data Security Platform.

A

access control policy	An access control policy is a reusable set of access control rules that serves a business context. An access control policy is a flexible construct that allows you to apply access control rules according to desired conditions. For example, you can write access control policies to define rules that examine and drop rows (records) according to the business condition and the actual data in those records.
access control rule	Access control rules act on the field level. Access control rules examine the actual data and discard each record being queried (requested) according to the rule's conditions.
access request	See project request .
asset	Assets are data structures; for example the tables in an Oracle® or PostgreSQL database.
asset registration request	An asset registration request is an inquiry made by a data owner to add a data asset (a database table, for example) to a dataset. A data guardian approves or denies asset registration requests.
attribute-based access control (ABAC)	<p>Attribute-based access controls (ABACs) are conditional policies and rules that regulate how users' access fields or rows, based on specific attributes, such as location, terms, and tags.</p> <p>ABACs determine how the platform applies policies and rules. In contrast, field-level access controls and record-level access controls determine where (on which assets, rows, or fields) the platform applies the policies and rules.</p>

B

business information	<p>Business information provides definition, structure, and clarity to data assets, consumption projects, policies, and rules by representing the context and semantics of an organization.</p> <p>Business information includes data classes, tags, terms, and purpose.</p> <p>Business information assists users to find and understand content on the platform and guides when to apply transformations based on attributes and conditions.</p>
----------------------	--

C

cell-level transformation	<p>Cell-level transformations allow you to select a different transformation for each distinct record of a specified field (column), that is, a cell, based on varying (logical) conditions.</p> <p>For example, you can instruct the platform to apply different transformations to an identity number or postal code in a given record based on the value of country of residence in a specific cell.</p>
connection	<p>A connection is a configuration for connecting to and reading data from a data source, such as a JDBC connection string. The platform uses this connection information to read metadata attributes from a data asset, to read the data itself, and to write the processed data to the target location.</p>
control plane	<p>The control plane is a logical perimeter that does not have direct access to data but may host components that drive operations in the data plane.</p> <p>The control plane is where policies, rules, projects, and assets are created and managed.</p> <p>The architectural split between the control plane and the data plane allows for configuration, orchestration, and administration (control) without the need to access data, but the ability to process data close to the source within a given jurisdiction. The control plane allows for this by using metadata, data classes, and other representations of the data.</p>

D

data agent	<p>The data agent provides access to the data plane whenever required by the control plane, for example to retrieve the schema for a data asset. It makes a long-lived connection to the data bridge on startup.</p>
data bridge	<p>The data bridge is the component in the control plane that handles communication with the data plane. It acts as a Google Remote Procedure Call (gRPC) server. It is replicated, and it sits behind an ingress with a load-balancer.</p>
data class (class)	<p>A data class is a categorization that data owners apply to fields within data assets to indicate the category of data. Within the Privitar Data Security Platform, data owners can apply a data class to identify the data's category and ensure that that kind of data is classified consistently throughout your organization. For example, data classes can classify birth dates, national identifiers, and postal codes.</p>

data consumer (consumer)	Data consumers are users on the Privitar Data Security Platform who request and consume data from the platform. Data consumers require direct access to data as part of their job responsibilities.
data exchange (exchange)	<p>A data exchange is a secure online portal where data owners can classify sensitive datasets, and data consumers can access them, without compromising data safety.</p> <p>Each data exchange is separate and different from other data exchanges, being a discrete entity within an enterprise.</p>
data guardian (guardian)	<p>Data guardians are users on the Privitar Data Security Platform who develop and maintain company policies and rules that govern data usage, including how the organization adheres to regulatory and compliance guidelines and requirements.</p> <p>Data guardians are responsible for approving all data requests, including requests to register data on the platform and requests to access data outside the platform.</p>
data owner (owner)	Data owners are users on the Privitar Data Security Platform who register and classify data on the platform. Data owners understand where the data comes from, its quality, its meaning, and for what purposes it can be used.
data plane	A data plane is a set of services used for the reading, writing, and processing of data. It contains a data agent and services capable of provisioning data, such as a data proxy or an integration using the Privitar SDK.
data proxy (proxy)	The data proxy is a Java Database Connectivity proxy (JDBC proxy) that allows data consumers to access sensitive data to which de-identification policies have been applied. It makes calls to the data bridge to fetch the information it needs, for example the details of how to connect to the sensitive data and the policies to be applied.
dataset	A dataset is a logical container of assets that is also known as a "data product." Its purpose is to group and facilitate an easier search experience. Data owners make datasets available to data consumers.
data type (type)	A data type is the data's categorization that is read from the source. Examples include: integer and string. The data type references how data is stored in a database, and each data type can have a different corresponding transformation. For example, you can store a person's age as an integer or a string.

E

encryption	<p>Encryption is the act of using a cryptographic algorithm to derive a value that is applied to a value in a dataset in such a way that only authorized parties can access the original value. In an encryption scheme, the original value, referred to as plaintext, is encrypted using an encryption algorithm to generate ciphertext that authorized parties can only read if it is decrypted. Encryption can be used as a de-identification technique.</p> <p>It is good practice to encrypt data at rest and in transit. However, while encryption can help protect against unauthorized access, it does not protect the privacy of individuals' data when it's used by people who are authorized. This is known as an insider attack.</p>
enterprise administrator (enterprise admin)	Enterprise administrators are users who perform operations within the Privitar Data Security Platform, such as creating a data exchange, creating a data plane, and configuring a data plane.
exchange	See data exchange .
exchange administrator (exchange admin)	Exchange administrators are users who perform tasks within a data exchange, such as creating and editing a data plane, managing users and groups, and performing everyday administration tasks.

F

field-level access control	<p>Field-level access controls are conditional policies and rules that regulate users' ability to access individual fields of a data asset. Field-level access controls determine which fields of the original dataset the platform retrieves prior to applying data transformation rules. Field-level access controls are implemented through drop field transformation, conditioned on attributes (ABAC), data consumer roles (RBAC), or purpose (PBAC).</p> <p>Field-level access controls determine where (on which fields) the platform applies policies and rules.</p>
field-level transformation	<p>Field-level transformations apply the same transformation to the entire field (column).</p> <p>The platform determines whether to apply a field-level transformation based on the data class of the column.</p>

H

HashiCorp® Vault Key Management System (HashiCorp® Vault KMS)

The HashiCorp® Vault KMS is a key management system (KMS) used to create and control encryption keys, which you use to encrypt data. A KMS is a system for the management (generation, distribution, storage, and more) of cryptographic keys and their metadata.

K

key format

The Privitar Data Security Platform uses "asymmetric" (or public key) encryption, which uses a pair of distinct, yet related keys. One key (the public key) is used for encryption, while the other in the pair (the private key) is used for decryption by an authenticated recipient (user).

L

linkability

"Linkability" is the probability of inferring the original value of transformed data by linking values from different datasets. Applying different tokens to the same value in different datasets reduces the ability to re-identify or de-anonymize data.

M

migration project

A migration project is a collection of raw data assets that a data owner wishes to mask and move to cloud storage or cross jurisdiction. A data consumer with appropriate authorization may consume the masked assets from the new location where, according to policy, masked data may be reversed.

P

consumption project

A consumption project is a collection of data assets that a team of data consumers wishes to provision safely. While data owners manage the data assets themselves, data consumers manage consumption projects, including linkability between assets. However, data consumers will not have access to the data within a consumption project until a data guardian approves their access.

policy

A policy is a reusable set of rules that serves a business context. Users of the platform can utilize the following types of policies:

- [access control policies](#)

- [transformation policies](#)

privacy enhancing technology (PET)	A privacy enhancing technology is a transformation type used to modify raw data to remove sensitive data elements. The Privitar Data Security Platform offers many PETs. These are the transformation types that data guardians select when building policies.
Privitar NOVL	Privitar NOVL is a feature of the Privitar Data Security Platform that applies consistent tokenization without a token vault. NOVL allows for data linkability across regions. NOVL also offers faster throughput and less latency than most vaulted solutions.
Privitar Query Engine	The Privitar Query Engine retrieves relevant policies and applies them to assets. The Query Engine transforms SQL queries, and the data retrieved with them, in compliance with the applicable policies.
project request (request)	A project request is an inquiry made either by a data consumer to use the assets in a data consumption project or by a data owner to create a migration project. Data guardians approve or deny all project requests.
provision	Provisioning is the act of making data available in a secure way to users and applications.
purpose	A purpose is the data consumer's intended use for the data in a consumption project. Data guardians use purposes as attributes in rules. Examples might include, "to find sources of bad loans" or "to build customer 360 profiles."
purpose-based access control (PBAC)	<p>Purpose-based access controls (PBACs) are conditional policies and rules that regulate how users' access fields, rows, or entire data assets, based on a consumption project purpose selected by a data consumer.</p> <p>PBACs determine how the platform applies policies and rules. In contrast, field-level access controls, and RLACs determine where (on which fields, rows, or assets) the platform applies policies and rules.</p>

R

record-level access control (RLAC)	Record-level access controls (RLACs) are conditional policies and rules that regulate users' ability to access individual records of an asset based on the values of selected fields of the same record. Record-level access controls determine which records of the original dataset the platform retrieves prior to applying transformation rules. Unlike data transformation rules, which are based solely on metadata,
------------------------------------	--

record-level access control rules are based on a combination of the data itself and metadata.

Record-level access controls (RLACs) determine where (on which records) the platform applies policies and rules. Attribute-based access controls (ABACs), purpose-based access controls (PBACs), and role-based access controls (RBACs) determine how the platform applies those policies and rules.

region

1. In the Privitar Data Security Platform, a region is a name for the geographical location, such as the location of a data exchange or a data agent. This is closely tied to jurisdiction. Some regulations require that data must remain within certain jurisdictions.
2. In cloud computing a region, (aka "geography"), is a named set of cloud resources in the same geographical area. A region is comprised of availability zones.

regular expression
(regex)

A regular expression is a series of characters that specifies a pattern to match text and numeric data formats. The Privitar Data Security Platform uses regular expressions to replace text strings and numbers with random characters.

For example, for an initial value of `abcdef`, you could use the following regular expression `[a-z]{6}` to produce an output such as `mvskyc`.

request

See [project request](#) and [asset registration request](#).

role-based access
control (RBAC)

Role-based access controls (RBACs) are conditional policies and rules that regulate how users access fields or rows, based on specific roles provided as user groups.

RBACs determine how the platform applies policies and rules. In contrast, field-level access controls, and record-level access controls determine where (on which fields, rows, or assets) the platform applies policies and rules.

rule

Rules are building blocks of policies. Rules are conditional based on attributes, such as user groups, terms, tags, locations, and so on. Rules also take actions specific to data classes and transformations.

Users of the platform can utilize the following types of rules:

- [access control rules](#)
- [transformation rules](#)

S

source connection A source connection is from where a data owner reads data.

system administrator (SysAdmin) System administrators are users who perform activities to install and set up the Privitar Data Security Platform. Most of these activities are external to the platform, such as deploying the platform, managing secrets required for installation, performing backup and restore activities, and performing updates to the platform.

T

tag A tag is a keyword that you can define to describe objects, such as when you want to group objects together or add context to those objects. For example, you might want to define tags that correspond to geography, line of business, project names, or applications. Tags help facilitate search and filtering.

target connection A target connection is to where a data consumer provisions data.

term Terms are words used within your organization to describe business concepts in plain language. Adding them to the platform ensures consistent use of those words throughout your organization. Terms also lend meaning to physical assets and their fields and give them context. When data consumers are browsing assets, terms allow them to understand the business meaning and semantics of the physical asset. Examples of terms could be "account type," "customer level," or "credit risk rating."

tokenization Tokenization is a form of fine-grained data protection that replaces a clear value with a randomly generated synthetic value that stands in for the original as a "token." The pattern for the tokenized value is configurable and can retain the same format as the original, which means fewer downstream application changes, enhanced data sharing, and more meaningful testing and development with the protected data.

token vault A token vault is a secure database (for example, PostgreSQL or Amazon DynamoDB) where you can store tokens generated during the de-identification of a dataset. Token vaults are only used for consistent tokenization (always returning the same token for the input value). Each token in a token vault is unique. That is, each token is only returned for one value. Token vaults allow for re-identification. That is, you are able to take a token from a de-identified dataset and look up the original input value.

transformation	A transformation defines a set of behaviors (privacy enhancing technologies) for the platform to execute on a field in a dataset to de-identify it, while still preserving data utility.
transformation policy	<p>A transformation policy is a reusable set of transformation rules that serves a business context. A transformation policy is a flexible construct that allows you to apply transformation rules in the way that best meets your needs. For example, you can write a policy around a regulation (such as HIPAA or GDPR) or around a business context (such as provisioning data for marketing analytics).</p> <p>The order of transformation policies matters. The platform applies them in the order that they are arranged by the data guardian.</p>
transformation rule	Transformation rules are conditional based on attributes, such as user group, terms, tags, location, and so on. Transformation rules apply pre-defined transformations to data classes.

W

watermark	<p>A watermark is a unique digital pattern created by the Privitar platform that is added into the records of de-identified datasets for traceability reasons. The platform adds watermarks to the data during the process of de-identification. They are invisibly embedded and distributed throughout the data, and as a result are robust against tampering and operations, such as filtering or reorganizing of the data.</p> <p>In the event of a leak or data breach, watermarks can be used to identify the data and plug potential security holes faster. Watermarks can also act as a deterrent to anyone handling the data, encouraging them to take the security of the dataset seriously when they know that the data can be traced.</p>
-----------	--