appviewx

KUBE+ Guides

Version: 2023.1.0 FP3

Copyright AppViewX, Inc.

External Reference Links

This product includes software developed by the CentOS Project (www.centos.org).

This product includes software developed by Red Hat, Inc. (www.redhat.com).

This product includes software developed by VMware, Inc. (www.vmware.com).

Contact Information

AppViewX, Inc.

222 Broadway, FL 19

New York, NY 10038

Email: info@appviewx.com

Web: www.appviewx.com

Contents

Preface

Revision History

Revision	Description	Date
1.4	Updated documents for Release 2023.1.0 FP3 HF3.	August 2024
1.3	Updated documents for Release 2023.1.0 FP3.	June 2024
1.2	Updated documents for Release 2023.1.0 FP2.	February 2024
1.1	Updated documents for Release 2023.1.0 FP1.	November 2023
1.0	Initial release of document for Release 2023.1.0.	September 2023

About this Guide

Welcome to the KUBE+ guide, introducing AppViewX KUBE+, a cutting-edge K8s Crypto Mesh designed for cloud and Kubernetes environments. This platform ensures top-notch security for containerized workloads by seamlessly handling TLS certificate issuance and management through automation. With KUBE+, administrators gain full control over the lifecycle of X.509 certificates within their Kubernetes cluster, empowering them with comprehensive certificate management capabilities.

Audience

The document is intended for internal users and customers of AppViewX.

Text Conventions

The following text conventions are used in this document:

Convention	Description
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`codeblock`	Indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Chapter 1: KUBE+ User Guide

KUBE+ Overview

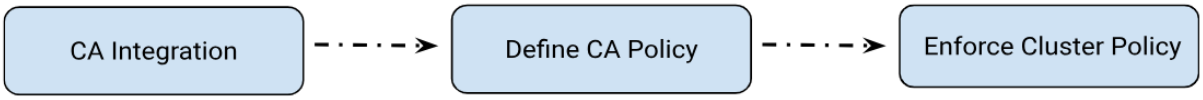
Simplify and Modernize Certificate Lifecycle Management (CLM) in Kubernetes

AppViewX KUBE+ is a comprehensive certificate lifecycle management solution for Kubernetes environments. It provides a central solution to discover, manage, automate, and govern certificates (or machine identities) across cloud-native containerized workloads and Kubernetes infrastructure. By bringing together visibility, automation, and policy-driven control, KUBE+ bakes security into the core of DevOps pipelines and Kubernetes management.

3 Steps to Bridge the Gap between Agility and Security in Kubernetes

1. Get Visibility into all SSL/TLS certificates across your Kubernetes environments.

2. Enforce PKI policies to ensure the use of compliant CAs and strong crypto-standards.



3. Automate end-to-end Certificate Lifecycle Management.

KUBE+ Features

AppViewX KUBE+ offers comprehensive lifecycle management of x.509 digital certificates across K8s clusters and container workloads. It comes equipped with a variety of features, including the following:

- **Visibility & Control**: This platform provides a single solution for discovering, ensuring compliance, and maintaining control over TLS certificates across K8s and multi-cloud infrastructures.
- **Full TLS coverage on K8s**: Pod-Pod Communication (mTLS), Ingress Traffic, Ephemeral Pods, K8s Infra.
- **Enterprise PKI**: Automated certificate lifecycle management integrated with Public and Private CA (Issue, renew, and revoke).
- **High Availability & Resiliency**: Always available PKI with Offline CA issuance.
- **Business Continuity**: Simplify and streamline the Enterprise PKI certificate issuance process via Workflow Orchestration.
- **Ecosystem**: One single integrated platform (CA, Vault, TLS keystores, ITSM, K8s platform, & DevOps Tools).
- **Governance**: Audit & Control of certificates across Hybrid Cloud Infra.

KUBE+ Deployment Architecture

Overview

AppViewX KUBE+ consists of a control plane and a set of in-cluster components, designed on a microservice architecture and deployed as a container workload. The control plane is equipped with the ability to gain valuable insights into SSL/TLS certificates throughout K8s clusters and enforce PKI policies for these clusters. Meanwhile, the in-cluster components handle the automated lifecycle management of x.509 digital certificates for K8s clusters and container workloads.

- **KUBE+**: As a control plane feature, KUBE+ serves as a centralized hub, empowering users to effortlessly discover, manage, and automate certificate issuance via a robust policy-driven approach.
- **Cert-Orchestrator**: The primary component of KUBE+ which is deployed within a Kubernetes cluster and is responsible for managing the lifecycle of certificates.

KUBE+

AppViewX KUBE+ is a powerful control plane that offers centralized visibility and management for certificates across Kubernetes and container workloads through its integrated components. With KUBE +, you can maintain up-to-date certificate inventories, track certificate chain of trust, locations, expiration dates, and crypto standards.

The platform enables you to establish and enforce enterprise-wide PKI policies and role-based access controls, providing enhanced security. KUBE+ also facilitates easy audits and compliance validation with comprehensive reporting and logging functionalities.

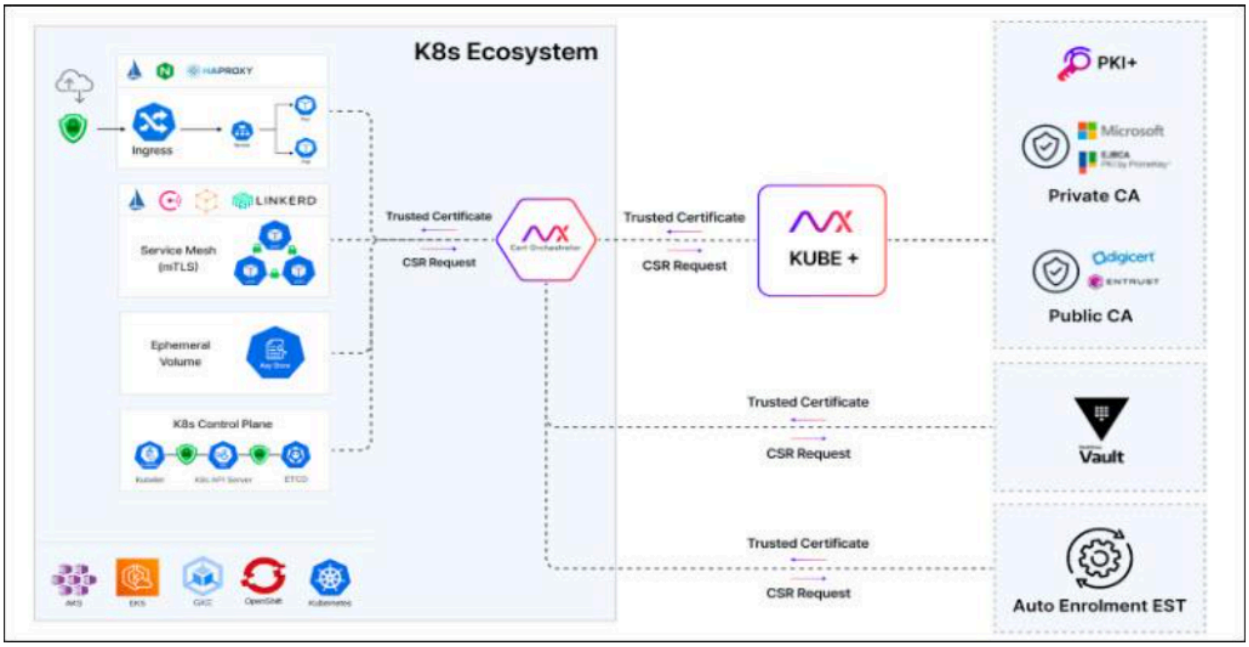
Moreover, the seamless integration with both public trust and private trust Certificate Authorities (CAs) simplifies the management of the entire certificate lifecycle, ensuring a smooth and efficient process.

Cert Orchestrator

Cert-Orchestrator is a Kubernetes cryptomesh that utilizes a microservice architecture and is deployed as a container workload. Its purpose is to facilitate the implementation of KUBE+ across workloads and hosted clusters.

Users have the option to deploy and integrate the cert orchestrator with AppViewX KUBE+. This enables the full range of Crypto Mesh features, or specific features can be selectively implemented depending on the use case. The cert-orchestrator comprises several sub-components that enable the KUBE+ solution:

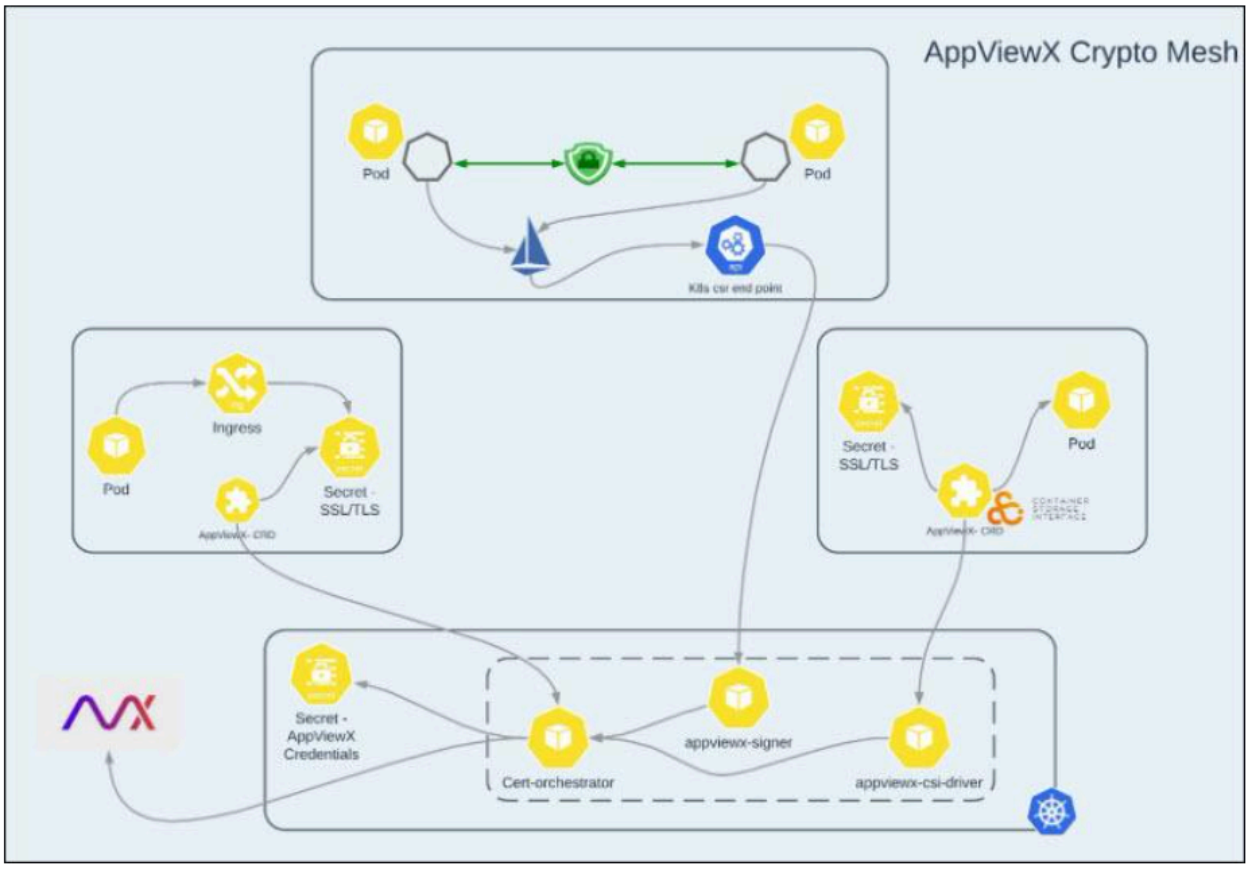
- **Cert-orchestrator controller**: The primary component of KUBE+ is deployed to enable end-to-end certificate lifecycle management, including discovery, enrollment, and renewal.
- **AppViewX-Signer**: The KUBE+ component is deployed together with Cert-orchestrator to manage certificates within Istio Service Mesh.
- **AppViewX-CSI**: The KUBE+ component is deployed along with Cert-orchestrator to manage certificates within ephemeral volumes of pods.
- **AppViewX-Infra-orchestrator**: The KUBE+ component, deployed as a daemon set along with Cert-orchestrator, enables certificate lifecycle management for the Kubernetes infrastructure or control plane, including certificate discovery and enrollment from external public or private CAs.



Communication Flow

Cert-orchestrator and AppViewX KUBE+ primarily communicate through REST API-based communication to facilitate end-to-end certificate lifecycle management operations.

The communication between the subcomponents of the cert-orchestrator and AppViewX KUBE+ for their respective business logic is enabled and routed through the cert-orchestrator controller. The cert-orchestrator serves as the control plane for the end-to-end certificate lifecycle management operations, which means that any outgoing communication from the Kubernetes cluster to AppViewX KUBE+ is only routed through the primary controller.



Authentication and Access Control

The communication between the K8s cluster and AppViewX KUBE+ takes place through a REST API, and it is authenticated. AppViewX provides several authentication modes to access its API.

Authentication

- **Basic Authentication:** AppViewX offers multiple modes of authentication for accessing its API. Users can create a dedicated user for authentication, either an external user (LDAP, RADIUS and TACACS) or an internal user created in AppViewX. For more information on Basic Authentication, refer to Platform User guide.
- **OAuth 2.0 (Service Account):** Users have the option to create a service account that is enabled through AppViewX OAuth 2.0. They can then obtain the client ID and client secret from the service account, which can be used for authentication purposes. For more information on OAuth 2.0, refer to Platform User guide.

Access Control

Each role assigns a specific set of permissions relating to the modules that can be accessed and the tasks that can be performed in each AppViewX module. The roles can be assigned only to a user group. The user groups that are assigned with a role will automatically inherit all the associated permissions. User groups can be assigned with more than a role.

The Roles management of Inventory comprises some of the Out of the Box (OOB) roles available for KUBE+ features via cert-orchestrator. The OOB roles can be cloned, enabled, and disabled. It can not be updated or deleted. Administrators can also create custom roles. Custom roles can be updated, deleted, enabled, and disabled. Users can either use OOB roles (if they match their needs) or custom roles to map to user groups. KUBE+ is enabled with a list of Out Of the Box roles to ease the process of defining access and role permissions for different personas accessing KUBE+.

KUBE-Application-User: For DevOps/Application users/CloudOps teams to perform Certificate Lifecycle Management for their applications (or) business units.

KUBE-PKI-Administrator: For Infosec and PKI teams to define and enforce PKI policies for their Kubernetes environments.

KUBE-cert-orchestrator: Role to be mapped to the service account or user used for deployment of Cert-Orchestrator for performing CLM operations on individual Kubernetes clusters.

As previously explained, when deploying Cert-Orchestrator within the Kubernetes cluster for access control, it's necessary to follow these steps:

1. Create a user or service account and associate it with the User Group.
2. Assign the KUBE-cert-orchestrator role to the User Group.
3. Additionally, ensure that the super access resource is linked to the User Group.
4. Ensure MFA (Multi Factor Authentication) is disabled for this user since it is used for API access.

For detailed instructions to perform any actions on role, see Platform User Guide.

Visibility & Control

Overview

To eliminate outages and ensure security, Infosec and PKI teams need to prevent security blindspots by getting complete visibility into all certificates across Kubernetes environments.

To achieve visibility into all certificates using KUBE+, the DevOps/CloudOps teams must adhere to the following process.

1. **Onboard Cluster** - Deploy AppViewX KUBE+ component in the desired cluster.
2. **Manage Cluster** - Deploy cert-orchestrator with the feature gate to discover certificates and manage the cluster in the inventory.
3. **Visibility & Insights** - The certificates discovered from these managed clusters can be visualized in the inventory and dashboards for assessing the security risks.

Prerequisites to Onboard a Cluster

The cert-orchestrator is deployed as a container workload on the same Kubernetes infrastructure where the workloads requiring certificates are running.

To begin the deployment process, the following prerequisites must be met and verified:

- Cluster Running Kubernetes version 1.22 and above; OpenShift version 4.10, 4.11 or 4.12.
- Docker version 20.x installed and running as the container runtime.
- Containerd running if the cluster does not support docker runtime.
- Access to AppViewX KUBE+ instance deployed on-prem or an AppViewX SaaS instance subscribed.
- Helm and kubectl packages installed in the Kubernetes cluster.

- In the case of a bastion host, the bastion host should be connected to the cluster.
- Access to the helm and docker image repositories hosted at AppViewX.

Note: Contact AppViewX support to obtain the authentication credentials for the Docker and Helm repository.

Cluster Inventory

To deploy the cert-orchestrator in the Kubernetes cluster the deployment configuration for the cluster should be obtained from AppViewX by onboarding the cluster in the Cluster Inventory and the cluster to be set in Managed status.

Cluster Inventory displays the list of clusters within a cert-orchestrator. On the Cluster Inventory page, you can:

- refresh the list, click the (**refresh**) icon.
- go to the pages, click the (**navigation**) icon.

The cluster inventory list includes the following information:

Column and Description Table

Column Name	Description
Cluster Name	Name of the Kubernetes cluster and metadata about the cluster retrieved on clicking the name of the cluster.
Vendor	The Kubernetes Cluster Provider.
No. of Nodes	Count of nodes in the cluster.
State	State of the Cluster for CLM operations. Possible Options: Managed (CLM operations can be performed) and Unmanaged (CLM operations cannot be performed).
Status	Health of the cert-orchestrator in the cluster. An event can be configured to send a notification when the cluster status changes to unhealthy. To

Column and Description Table (continued)

Column Name	Description
	configure an event, see Configuring Notification Event.
Last Updated Time	Timestamp of the most recent health check update.
Created By	The user who generated the deployment configuration.
Actions	Actions icon allows updating the deployment configuration for the Cluster.

- Onboarding a Cluster
- Managing a Cluster
- Unmanaging a Cluster
- Modifying Feature Gates of a Cluster
- Deleting a Cluster

Onboarding a Cluster

Connect Cluster enables you to obtain the deployment configuration of the cluster for deploying and managing the cert-orchestrator from AppViewX KUBE+.

Generating a deployment configuration for the cluster involves initiating the generation of deployment templates or configuration steps. KUBE+ enables DevOps teams or cluster administrators to generate deployment configurations for their clusters using various methods.

- **Easy onboarding**: - Enables DevOps team to generate a common deployment template for onboarding multiple clusters.
- **Advanced onboarding** - Enables individual cluster administrators to generate cluster specific deployment configuration.

- Onboarding a Cluster - Easy Method
- Onboarding a Cluster - Advanced

Onboarding a Cluster - Easy Method

Deploy and manage cert-orchestrator on multiple Kubernetes clusters using a common deployment configuration.

The common deployment template enables the DevOps team to generate a generic command that can be executed at multiple clusters where these commands just need Cluster Name and Vendor name to be changed accordingly while executed in the respective clusters.

A common credential will be generated with a default user group associated to the OOB role and resource which is downloadable as a YAML and the object type is a Kubernetes secret. The credential can be changed by editing the cluster once it's successfully managed in the inventory.

Additionally, the common deployment configuration enables default services, including discovery of certificates and provisioning of certificates from the Kubernetes cluster. Administrator can modify the services after managing the cluster in the cluster inventory.

If the on-boarding policy is enabled, new clusters will be onboarded with automated policy and PKI configuration. To configure on-boarding policy, follow the Configuring Policy Settings steps.

To obtain the common deployment template for Kubernetes cluster:

1. Go to **menu** > **KUBE+** > **INVENTORY** > **Cluster Inventory**.
2. Click **Connect Cluster** on the menu bar.
3. Select **Get Started** under **Easy Onboarding**.
4. On the **Cluster Advanced Onboard** page, enter values in the form fields to generate the deployment/ installation command. Details on the mapping of each field are provided in the table below:

Generating Helm Command - Fields and Description Table

Field	Description
Cluster Details	
Enter Cluster Name*	Enter a unique cluster name in the format of FQDN. Example: my-cluster.net.
Vendor*	Select the K8s vendor where the cert-orchestrator is deployed from the dropdown list. The options are: • EKS • AKS • GKE

Field	Description
	• OpenShift • Self-Managed
**: Mandatory fields*	

5. Click **Generate Installation Command** to get the Helm command in the Commands field.

> **Note:**
>
> • To see the commands in the full screen view, click the (**Expand**) icon.
>
> • To copy the command, click (**Copy**) icon.

6. Click **Download Credentials** to download the credentials of your AppViewX environment. Once deployed in the cluster, this credentials enables connectivity between the Kubernetes cluster and AppViewX environment from the cert-orchestrator deployed in the Kubernetes cluster.

> **Note:**
>
> - The downloaded credential creates a default OAuth svc account in AppViewX named **kube-svc-account** with a client ID and client secret.
> - The **kube-svc-account** is linked to a default user group named **kube-svc-usergroup**, which, in turn, is associated with the OOB **kube-cert-orchestrator** role and super access resource.
> - The downloaded credential includes the connectivity URL, serving as the AppViewX login URL.
> - After successfully onboarding and managing the cluster in the cluster inventory, the user or admin with access to the inventory can edit the cluster. This allows for modifications to the service account and connectivity URL as needed.

Onboarding a Cluster - Advanced

Deploy and manage cert-orchestrator on Kubernetes clusters by customizing installation commands and choosing specific services.

The advanced onboarding enables the DevOps team/cluster administrators to generate a specific install command that can be executed at the clusters, administrators or DevOps team can modify the namespace, connectivity URL and KUBE+ services to be deployed in the cluster.

To obtain cert-orchestrator deployment configuration for the respective Kubernetes cluster:

1. Go to **menu** > **KUBE+** > **Inventory** > **Cluster Inventory**.
2. Click **Connect Cluster** on the menu bar.
3. Select **Get Started** under **Advanced Onboarding**.
4. On the **Cluster Easy Onboard** page, enter values in the form fields to generate the deployment/ installation command. Details on the mapping of each field are provided in the table below:

Generating Helm Command - Fields and Description Table

Field	Description
Cluster and Connectivity Details	
Enter Cluster Name*	Enter a unique cluster name in the format of FQDN. Example: my-cluster.net.
Vendor*	Select the K8s vendor where the cert-orchestrator is deployed from the dropdown list. The options are: • EKS • AKS • GKE • OpenShift • Self-Managed
Connect To*	Select one of the following options to establish a connection between the cert-orchestrator and AppViewX across different AppViewX deployment scenarios: • **AppViewX URL** - For on-prem deployment, select this option. • **Cloud Connector URL** - For cloud SaaS deployment, select this option.

Field	Description
URL*	Enter the URL based on the Connect To type (onprem/cloud connector).
Credential Type*	Select one of the following credential types for integrating the cert-orchestrator with AppViewX: • **Basic Authentication** • **OAuth2.0**
Username*	This option is applicable for the Basic Authentication of the Credential Type and will auto populate the list of users from the user inventory. Select the required user to be used for authentication.  **Note:** In this mode, only users created within the AppViewX database or on boarded via AAA (LDAP, RADIUS, TACACS) are supported. SSO credentials cannot be used for API authentication. It is recommended to use OAuth 2.0 for authentication instead.
Crypto Mesh Details	
Namespace*	Enter the Namespace where the cert-orchestrator is to be deployed. It is recommended to install in the crypto-mesh namespace.
Features*	Select the list of feature gates to be enabled/ disabled in the cert-orchestrator deployment configuration for the cluster.
Certification Group*	Select the certificate group to onboard certificates:

Field	Description
	• **Auto Create Group** - This option enables Auto creation of Certificate Groups in AppViewX with the Group Name as the Namespace Name. • **Use Existing** - This option allows you to choose the existing certificate group. If you choose this option, select a group from the **Select Group** dropdown menu. .
Enable Private Key Discovery*	Set the value to “True”, if you want to discover the private keys from the Kubernetes secrets.

5. Click **Generate Installation Command** to get the Helm command in the Commands field.

> **Note:**
>
> • To see the commands in the full screen view, click the  (**Expand**) icon.
>
> • To copy the command, click  (**Copy**) icon.

6. Click **Finish**.
   Execute the copied installation commands sequentially on your Kubernetes cluster where the cert-orchestrator is to be deployed.

> **Note:** Upon clicking **Finish**, the deployment generated for the cluster will not appear in the cluster inventory until the cert-orchestrator is successfully deployed within the cluster.

To verify if the cert-orchestrator is deployed and functioning as expected, execute the following command:

```
kubectl get pods --all -n crypto-mesh
```

> **Note:** The initial status of the cert-orchestrator pod will be in 1/2 running state and the state will be changed to 2/2 upon approval/moving the cluster to managed state in the Cluster inventory.

- Expected status is for the pods to be in running status with 1/2 state.
- In case of any issues or logs to be collected or verified, execute **kubectl logs -f <cert-orchestrator-podname> -n crypto-mesh**.
- Ensure to review the deployment prerequisites for ephemeral volume if the Ephemeral Volume use case is selected.

Managing a Cluster

CLM operations on the Kubernetes cluster where the cert-orchestrator is deployed will be enabled only when the cluster is in **Managed State.**

Upon successful installation of cert-orchestrator in the desired cluster, the cluster information will be automatically populated in the Cluster Inventory in a **Pending Approval** status.

To change the cluster status to Managed:

1. Go to **menu** > **KUBE+** > **Inventory** > **Cluster Inventory**.
2. Select a preferred cluster(s), and then click **Manage**.

Note: To enable the automatic transition of clusters into a managed state, administrators can set the 'Cluster Auto Approval Policy' to **True** by navigating to **menu** > **KUBE+** > **System Administration** > **Cluster Settings**.

It takes a while to update the status of the cluster to Managed.

Unmanaging a Cluster

You can restrict additional CLM operations on a designated cluster that is already in a Managed State within the Cluster Inventory by unmanaging the cluster.

To unmanage a cluster:

1. Go to **menu** > **KUBE+** > **Inventory** > **Cluster Inventory**.
2. Select a managed cluster.
3. Click **Unmanage** on the menu bar.

   It takes a while to update the status of the cluster to Unmanaged.

Modifying Feature Gates of a Cluster

The feature gates configured while generating the deployment configuration can be modified post successful deployment of cert-orchestrator in the desired cluster.

To modify feature gates for a cluster, by simply clicking on the edit icon parallel to the cluster name.

1. Go to **menu** > **KUBE+** > **Inventory** > **Cluster Inventory**.
2. Click the icon on a cluster in the **Actions** column.
3. On the **Update Cert Orchestrator** page, cluster administrators can modify the feature gates, AppViewX URL, Enable/Disable Private Key Discovery, Modify the service account associated to the cluster.
4. If there are changes in the AppViewX URL or the service account (credential) associated with the cluster, the cluster administrator can download the credential as a YAML file and deploy it in the respective cluster.
5. Click **Generate Update Command** to get the updated deployment configuration.
6. Copy and execute the install command in the desired cluster.

> **Note:** To go to the previous page, click on **Cluster Inventory** in the breadcrumb trail.

7. Click **Finish** to update the modified changes (or) click **Reset** to restore the previous configuration changes associated to the cluster.

Deleting a Cluster

You can delete clusters from the inventory, if the user chooses to restrict and remove CLM operations on the designated cluster.

To delete a cluster:

1. Go to **menu** > **KUBE+** > **Inventory** > **Cluster Inventory**.
2. Select the designated cluster, regardless of whether it is in a Managed or Unmanaged state.
3. Click **Delete** on the menu bar.

> **Note:**

- Deleting clusters from the inventory will not remove the in-cluster KUBE+ components. Users are required to follow the Uninstall and Clean Up steps within the cluster to complete the removal process.
- If off-boarding policy is enabled, the cluster and its associated components will be removed from the AppViewX inventory as configured. To configure the off-boarding policy, follow the Configuring Policy Settings steps.

4. On the **Delete Cluster** pop-up window, enter the comment and click **Yes**.
   After confirming the deletion of the cluster, the status of the cluster is changed to **Delete in progress**. It takes a while for deleting a cluster from the inventory. When the deletion is progress, if you try to delete again, the failure message will be displayed the "Selected Clusters are recently offboard triggered".

Deployment Prerequisites for Ephemeral Volume Feature Gate

Enabling the feature gate "Provision Certificates to Ephemeral Volumes" for certificate provisioning into pod volumes will require the activation of a CSI driver feature within the cluster where the cert-orchestrator is deployed.

To enable the CSI provider:

1. Add the CSI provider helm repository to the Kubernetes cluster by using the command: `helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts`.
2. Install the CSI provider drive in the Kubernetes cluster by using the command `helm install csi secrets-store-csi-driver/secrets-store-csi-driver --namespace crypto-mesh --set syncSecret.enabled=true --set enableSecretRotation=true --set rotationPollInterval=5m`.

Note:

- The value of RotationPollInterval can be configured to set the time interval at which the CSI driver makes a call to appviewx-csi-provider for synchronizing the certificate content from secret to pod volume.
- If the CSI provider driver already existed prior to the installation of the crypto mesh, you can skip this step.
- For CSI feature, in case of any issues or logs to be collected or verified execute **kubectl logs -f <appviewx-csi-provider-podname> -n crypto-mesh**.

OpenShift Security Permissions

For deploying the AppViewX CSI feature in OpenShift environments, the following elevated privileges are required. To execute the prerequisites, Contact OpenShift Administrator.

1. After deploying the appviewx-csi-provider in an OpenShift cluster, patch the DaemonSet for the AppViewX CSI Provider to run with a privileged security context.`kubectl patch daemonset appviewx-csi-provider \--type='json' \-patch='[{"op": "add", "path": "/spec/template/spec/containers/0/securityContext", "value": {"privileged": true} }]'`.
2. Add the security account for the Secret Store CSI Driver and AppViewX CSI Provider to the privileged security context constraint. To obtain the name of the service account associated with the AppViewX CSI provider, run the command `oc get sa -n crypto-mesh`, and copy the relevant name to be used in the command below: `oc adm policy add-scc-to-user privileged system:serviceaccount:crypto-mesh:secrets-store-csi-driver oc adm policy add-scc-to-user privileged system:serviceaccount:crypto-mesh:<replace with service account name>`.
3. Give additional access to the application for retrieving secrets with the AppViewX CSI Provider. Create a **SecurityContextConstraint** to **allowHostDirVolumePlugin** and **allowHostPorts** for the service account of the application.

```
cat > application-scc.yaml << EOF
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
 name: appviewx-csi-provider
allowPrivilegedContainer: false
allowHostDirVolumePlugin: true
allowHostNetwork: true
allowHostPorts: true
allowHostIPC: false
allowHostPID: false
readOnlyRootFilesystem: false
defaultAddCapabilities:
- SYS_ADMIN
runAsUser:
 type: RunAsAny
seLinuxContext:
 type: RunAsAny
fsGroup:
```

```
 type: RunAsAny

users:

- system:serviceaccount:crypto-mesh:<replace with service account name>

EOF

Kubectl apply -f application-scc.yaml
```

Visibility

Visibility - Overview

After successfully installing cert-orchestrator on the desired cluster and enabling the feature gate for certificate discovery, it will automatically detect certificates within the Kubernetes environment and create an inventory, allowing you to proactively manage all certificates from a unified dashboard.

Certificates discovered from Kubernetes are further classified into:

1. **Ingress** - Certificates which are used by Ingress controllers.
2. **Infrastructure certificates** - Certificate discovered from Kubernetes control plane components.
3. **Service Mesh** - Certificates which are enrolled from AppViewX for Service Mesh for mTLS authentications.
4. **Others** - Certificates which are not of any of the above 3 classifications will be classified as Others.

Server

Server certificate inventory is where all the server certificates with the EKU (Extended/enhanced Key Usage) Server authentication will be present.

In this release, renewals and regenerations of the server certificates are only supported through the Cert-Orchestrator, which is part of the in-cluster component of KUBE+.

To go to Server certificate inventory page, go to **menu** > **KUBE+** > **VISIBILITY** > **Server**.

The following table describes the options available on the Server Certificate inventory page:

Options available on the Server Certificate page

Options	Description
	Allows to switch between the view by clicking the toggle button.
Groups	Expanding this dropdown displays the certificate groups and the number of certificates in each group. Selecting a group will display the filtered list of certificates.
Filter Summary	Displays the status of certificates according to expiry, compliance, validity, and so on.
Advanced Search	Allows you to perform a quick search for specific data. Clicking on the search bar dropdown opens the **Advanced Search** window. To find the preferred server certificate, perform any of the following: 1. Choose the CAs from the Certificate Authority dropdown menu. 2. Enter the desired search terms in the Common Name, Serial Number, or Issuer Common Name field. 3. Select a certificate attribute from the dropdown list and if required add more certificate attributes to the search criteria by clicking the Add Certificate Attribute. 4. Click Search. The matching server certificates are displayed on the Server Certificate page.
Actions	Displays the list of actions you can perform on the certificates.

Options available on the Server Certificate page (continued)

Options	Description
	• Export Certificates • Download Certificates • Delete • Change Status • Assign Group • Unassign Group • Add/Modify Comments • Certificate Attributes • Renew Certificate • Revoke Certificate • CA Switch • Revocation Check
Columns	Allows you to select the columns to be displayed on the Server Certificate inventory page.
Number of Rows per Page	Hover the mouse over the number of row displayed on the page, the Show popup opens and choose the no. of rows to be displayed on the page.
	Allows to switch between the certificate inventory pages.
	Allows to refresh the certificate inventory data.

The Server Certificate inventory list includes the following information:

Column and Description Table

Column Name	Description
Common Name	The common name of the certificate.

Column and Description Table (continued)

Column Name	Description
Discovery Source	The source from which a certificate management system discovers and retrieves information about certificates
Serial Number	A unique identifier assigned to the certificate by the CA during the issuance process.
Group	The certificate group name.
Issuer Common Name	Issuer name of the certificate.
Valid To (GMT)	The expiration date and time of a certificate, expressed in Greenwich Mean Time (GMT).
Status	The status of the certificate.
Certificate Authority	Name of the Certificate Authority (CA).
Kube Attributes	Attributes configured in CA.

All

This inventory displays all CLM-issued, discovered, or uploaded Server Authentication certificates.

Ingress Certificates

Certificates discovered from Kubernetes secrets and secrets associated with Kubernetes ingresses are classified as Ingress certificates.

Infrastructure Certificates

Certificates discovered from Kubernetes control plane components via feature gate **“Discover K8’s Infra Certificates”** are classified as Infrastructure certificates.

ServiceMesh

KUBE+ provides the feature gate to secure pod-pod communication in a Kubernetes service mesh infrastructure with mTLS certificates signed by Enterprise PKI. If the feature gate **“Enable mTLS certificates for Service Mesh”** is enabled the mTLS certificates signed by AppViewX will be classified as Service Mesh certificates.

Others

Certificates discovered from Kubernetes secrets and not associated with any ingress (or) which does not classify into any of the above categories will be classified as Other certificates.

Client

Client certificate inventory is where all the client certificates with the EKU (Extended/enhanced Key Usage) Client authentication will be present.

In this release, renewals and regenerations of the client certificates are only supported through the Cert-Orchestrator, which is part of the in-cluster component of KUBE+.

To go to Client certificate inventory page, go to **menu** > **KUBE+** > **VISIBILITY** > **Client**.

The following table describes the options available on the server certificate inventory page:

Options available on the Client Certificate page

Options	Description
	Allows to switch between the view by clicking the toggle button.
Groups	Expanding this dropdown displays the certificate groups and the number of certificates in each group. Selecting a group will display the filtered list of certificates.
Filter Summary	Displays the status of certificates according to expiry, compliance, validity, and so on.
Advanced Search	Allows you to perform a quick search for specific data. Clicking on the search bar dropdown opens the **Advanced Search** window. To find the preferred server certificate, perform any of the following: 1. Choose the CAs from the Certificate Authority dropdown menu. 2. Enter the desired search terms in the Common Name, Serial Number, or Issuer Common Name field.

Options available on the Client Certificate page (continued)

Options	Description
	3. Select a certificate attribute from the dropdown list and if required add more certificate attributes to the search criteria by clicking the Add Certificate Attribute. 4. Click Search. The matching server certificates are displayed on the Server Certificate page.
Actions	Displays the list of actions you can perform on the certificates. • Export Certificates • Download Certificates • Delete • Change Status • Assign Group • Unassign Group • Add/Modify Comments • Certificate Attributes • Renew Certificate • Revoke Certificate • CA Switch • Revocation Check
Columns	Allows you to select the columns to be displayed on the Server Certificate inventory page.
Number of Rows per Page	Hover the mouse over the number of row displayed on the page, the Show popup opens and choose the no. of rows to be displayed on the page.

Options available on the Client Certificate page (continued)

Options	Description
	Allows to switch between the certificate inventory pages.
	Allows to refresh the certificate inventory data.

The Client Certificate inventory list includes the following information:

Column and Description Table

Column Name	Description
Common Name	The common name of the certificate.
Discovery Source	The source from which a certificate management system discovers and retrieves information about certificates
Serial Number	A unique identifier assigned to the certificate by the CA during the issuance process.
Group	The certificate group name.
Issuer Common Name	Issuer name of the certificate.
Valid To (GMT)	The expiration date and time of a certificate, expressed in Greenwich Mean Time (GMT).
Status	The status of the certificate.
Certificate Authority	Name of the Certificate Authority (CA).
Kube Attributes	Attributes configured in CA.

All

This inventory displays all CLM-issued, discovered, or uploaded Client Authentication certificates.

Ingress Certificates

Certificates discovered from Kubernetes secrets and secrets associated with kubernetes ingresses are classified as Ingress certificates.

Infrastructure Certificates

Certificates discovered from Kubernetes control plane components via feature gate **“Discover K8’s Infra Certificates”** are classified as Infrastructure certificates.

ServiceMesh

KUBE+ provides the feature gate to secure pod-pod communication in a Kubernetes service mesh infrastructure with mTLS certificates signed by Enterprise PKI. If the feature gate **“Enable mTLS certificates for Service Mesh”** is enabled the mTLS certificates signed by AppViewX will be classified as Service Mesh certificates.

Others

Certificates discovered from Kubernetes secrets and not associated with any ingresses (or) which does not classify into any of the above categories will be classified as Other certificates.

Trust Store

The Trust Store certificate inventory consists of the following:

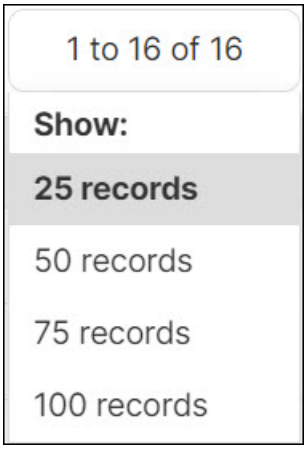
- Intermediate
- Root

To go to Trust Store certificate inventory page, go to **menu** > **KUBE+** > **VISIBILITY** > **Trust Store**.

Options available on the Intermediate and Root Certificate page

Options	Description
Advanced Search	Allows you to perform a quick search for specific data. Clicking on the search bar dropdown opens the **Advanced Search** window.
All	Allows you to filter the certificates by their Certificate Authority (CA).
Actions	Displays the list of actions you can perform on the certificates: • Export Certificates • Copy • Download • Delete

Options available on the Intermediate and Root Certificate page (continued)

Options	Description
	• Chain of Trust Preference • Certificate Attributes
Columns	Allows you to select the columns to be displayed on the Client Certificate inventory page.
Number of Rows per Page	hover the mouse over the number of row displayed on the page, the Show popup opens and choose the no. of rows to be displayed on the page. 
	Allows to switch between the certificate inventory pages.
	Allows to refresh the certificate inventory data.

The Trust Store certificate inventory list includes the following information:

Column and Description Table

Column Name	Description
Common Name	The common name of the certificate.
Certificate Authority	Name of the Certificate Authority (CA).
Serial Number	A unique identifier assigned to the certificate by the CA during the issuance process.

Column and Description Table (continued)

Column Name	Description
Valid From (GMT)	The start date and time of a certificate, expressed in Greenwich Mean Time (GMT).
Valid To (GMT)	The expiration date and time of a certificate, expressed in Greenwich Mean Time (GMT).
Issuer Organization	The name of the issuer organization of he certificate that acts as the CA.
Issuer Organization Unit	The name of the issuer organization unit of he certificate that acts as the CA.
Key Algorithm & Size	The key type of the certificate such as: 1. RSA 2. ECDSA and sizes such as: • 2048/4096/3072 bit length for RSA. 256 / 384 / 521 bit length for ECDSA. .
Signature	The private key of the certificate issuer.
Discovery Source	The source from which a certificate management system discovers and retrieves information about certificates.
Discovered File Name(s)	The name of the file discovered during a discovery process.
Application(s)	The applications that use the certificates.
Associated Object(s)	The objects that are linked to a particular certificate.

Intermediate

Intermediate certificate inventory is where the issuer certificates apart from the root certificate will be displayed. In the chain of trust, if there is more than one layer of the intermediate certificate (excluding root and end certificate), all those certificates will be shown in this inventory.

Root

Root certificate inventory is where the root certificate of all the issuer certificates will be maintained. For any chain of trust certificates, there can be only one root certificate.

Dashboard - Insights

Dashboard - Overview

AppViewX KUBE+ has a few in-built reports that are available under the dashboard. These reports can be used to categorize certificates based on specific parameters.

The reports can be used to enforce the DevOps/CloudOps team to follow standard PKI practices across your Kubernetes clusters.

On the Dashboard page,

- you can manually refresh the each report by clicking the (**Refresh**) icon
- Using the (**Options**) icon and select any of the following options:
  - **Collapse/Expand** - This option allows you to collapse or expand the report.
  - **Minimize** - This option allows you to minimize the report.
  - **Download as PDF** - This option allows you to download the report as .pdf to your system.
  - **Export as Excel** - This option allows you to export the report data as Excel to your system.

.

The reports displayed on the dashboard are:

- Kube Cluster: Insights
- Expiry Insights: Summary
- Expiry Insights: CA Based
- Cert-Manager: SSL Insights
- Certificate Status: K8s Secrets
- Certificate Expiry Timeline
- Expiry Remediation for Discovered Certificates
- KUBE Certificate Authority Switch

Kube Cluster: Insights

The Kube Cluster: Insights presents a concise overview of the clusters onboarded within AppViewX. Click on any widget to access detailed information specific to that particular summary on the respective widgets popup. Within the popup window, you can view all the entries and download the content in the desired format if needed. The Widgets within the Inventory Summary report are:

Widgets in the Kube Cluster: Insights

Widgets Name	Description
Clusters	The count of clusters onboarded within cluster inventory.
Worker Nodes	The summary of worker nodes.
Active Worker Nodes	The chart displays the count of active worker nodes.
K8s Secrets with Certificates	The count of certificates discovered from K8s Secrets.
Expired Certificates	The chart displays the count of expired certificates in Kubernetes cluster.
Managed Certificates	The chart displays the count of certificates managed within KUBE+ secure apps.
Certificate Authorities	The count of certificate authorities created.

Widgets in the Kube Cluster: Insights (continued)

Widgets Name	Description
Cluster Policies	The count of cluster policies created.
Certificate Issued	The count of certificate authorities issued.
Certificates Set to Auto-Renewal	The count of certificates set for the automatic renewal.

Expiry Insights: Summary

The Expiry Insights: Summary snapshot provides a visual representation of the summary of the certificates with their expiration dates in the form of a graphical display. This snapshot helps for a quick overview and analysis of certificate expiration information. Click on the corresponding area, and a popup window will appear with the relevant details. Within the popup window, you have the ability to view the list of certificate pertaining to the selected area and download the content in the desired format if needed. The Expiry Insights: Summary visual representation includes areas that display the total number of certificates and provides a breakdown based on their expiration details. This breakdown helps to understand the distribution of certificates according to their expiration dates, allowing for better visibility and analysis.

Expiry Insights: CA Based

The Expiry Insights: CA Based snapshot provides a visual representation of the summary of the certificates with their expiration dates across Kubernetes cluster in the form of a graphical display. This snapshot helps for a quick overview and analysis of certificate expiration information. Click on the corresponding area, and a popup window will appear with the relevant details. Within the popup window, you have the ability to view the list of certificate pertaining to the selected area and download the content in the desired format if needed. The Expiry Insights: CA Based visual representation includes areas that display the total number of certificates and provides a breakdown based on their expiration details. This breakdown helps to understand the distribution of certificates according to their expiration dates, allowing for better visibility and analysis.

Cert-Manager: SSL Insights

The Cert-Manager: SSL Insights presents a concise overview of the SSL certificates managed within a Kubernetes cluster using Cert-Manager.

Certificate Status: K8s Secrets

The Certificate Status: K8s Secrets snapshot provides a visual representation of the summary of the valid or deleted certificates in the Kubernetes secrets in the form of a graphical display. Click on the corresponding area, and a popup window will appear with the relevant details. Within the popup window, you have the ability to view the list of certificate pertaining to the selected area and download the content in the desired format if needed. The Certificate Status: K8s Secrets visual representation includes areas that display the total number of certificates and provides a breakdown based on their expiration details.

Certificate Expiry Timeline

Within this section, you can view the certificate expiry summary based on the expiry date in the form of graphs.

Expiry Remediation for Discovered Certificates

The Expiry Remediation for Discovered Certificates Snapshot provides a visual representation of the all the expired and to be expired certificates and allows the expiring certificates to valid certificates.

KUBE Certificate Authority Switch

The KUBE Certificate Authority Switch Snapshot provides a visual representation of the self-signed or any CA signed expiring certificates and allows to switch the expiring certificates to valid CA.

Viewing Dashboard Inventory

All the dashboards of AppViewX are listed under the Dashboard inventory. From KUBE+ dashboard you can go to AppViewX's Dashboard inventory.

To view dashboard inventory:

1. Go to **menu** > **KUBE+** > **DASHBOARD**.
2. Click  (**Dashboard Inventory**) on the command bar.
   The Dashboard inventory is displayed.

Aligning the Reports

You can drag and drop the reports within the dashboard to the new position and it can be aligned.

To align the reports within the dashboard:

1. Go to **menu** > **KUBE+** > **DASHBOARD**.
2. Drag and drop the reports to your preferred position.
3. Click the  (**Align**) icon.
   The reports are aligned to the new position.

Saving the Dashboard

When you drag and drop the widgets to organize them in the dashboard, complete the following steps to save the changes:

To save a dashboard:

1. Go to **menu** > **KUBE+** > **DASHBOARD**.
2. After making necessary changes in the dashboard, click the  (**Save Dashboard**) button in the Command bar.
3. A pop-up message appears at the top of the dashboard, "**Dashboard saved successfully**."

Downloading the Dashboard Page

You can download the dashboard page as a pdf.

To download the dashboard page as pdf:

1. Go to **menu** > **KUBE+** > **DASHBOARD**.
2. Click the  (**Download as PDF**) button in the Command bar.
   The dashboard is downloaded to your machine as a pdf file.

Scheduling and Emailing the Reports

You can schedule the reports and email them to the email IDs.

To schedule and email the reports:

1. Go to **menu** > **KUBE+** > **DASHBOARD**.
2. Click the  (**Schedule and Email Reports**) icon.
3. Enter the necessary details.
   The reports will be generated and sent to the email IDs as configured.

Refreshing the Dashboard

You have the option to refresh the data within the dashboard to display the most up-to-date reports..

To refresh the dashboard:

1. Go to **menu** > **KUBE+** > **DASHBOARD**.
2. Drag and drop the reports to your preferred position.
3. Click the  (**Refresh**) icon.
   The reports are refreshed within the dashboard.

Policy Enforcement

- Policy Enforcement Overview
- Certificate Authority (CA) Integration
- Groups
- Cluster Policy

Policy Enforcement Overview

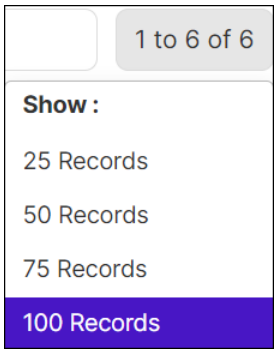
To meet compliance and to ensure strong crypto standards PKI policies should be defined and enforced across all your Kubernetes clusters onboarded and managed in KUBE+.

The process for defining and enforcing the policy definition for your managed clusters is as follows.

1. **CA Integration** - Integrate AppViewX KUBE+ with your Internal or External CA's for signing the certificates for your Kubernetes workloads.
2. **CA Policy** - Define CA Policy to enforce your organization crypto standards and map them to Certificate Groups (to categorize certificates based on business units).
3. **Enforce Cluster Policy** - Enforce dedicated CA Policy / PKI policy to one more cluster to promote secure and compliant certificate management practices.

On the CA Policy page,

- refresh the list, click the (refresh) icon.
- go to the pages, click the (navigation) icon.
- hover the mouse over the number of row displayed on the page, the Show popup opens and choose the no. of rows to be displayed on the page.



To view the list of policy, go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.

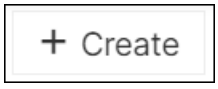
- Configuring Policy Details
- Configuring Policy for Amazon CA
- Configuring Policy for Amazon Private CA
- Configuring Policy for DigiCert CA
- Configuring Policy for EJBCA CA
- Configuring Policy for Entrust CA
- Configuring Policy for Entrust MPKI CA
- Configuring Policy for GlobalSign Atlas CA
- Configuring Policy for GlobalSign CA
- Configuring Policy for GlobalSign MSSL CA
- Configuring Policy for GoDaddy CA

Configuring Policy Details

To configure a policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click 

on the menu bar.
3. On the **CA Policy : Create** page, enter/select the field information for the **Policy Details** section.

Field and Description for the Policy Details Section

Name	Description
***Policy name**	Provide a unique name to identify the CA policy name. **Note:** No special characters other than '.', '-','_' are allowed. The name should not start with special characters.
Description	Provide a description of the policy.
***Policy Enforcement Type**	Select a policy enforcement type. The options are:

Name	Description
	• **Strict** (default) - This enforces the standards defined in the policy where a user cannot modify any parameters. • **Suggestive** - This suggests users with policy parameters. A user can modify suggested values if required.
Certificate Requests Need Approval	When enabled, it will enforce the peer approval process for any requests made for new/renew/regenerate/reissue or revocation of certificates. Peer approving the request is defined in the approval workflow.
Enable Access to Private Key	When enabled allows the user to download private keys from the holistic view.
Enable certificate push-bind access for a read-only user	Enabling the option might allow the user with the read-only user group to perform certificate push, bind, and rollback operations from the holistic view.
Validate issuer and root certificate for compliance	Enabling the option would validate if the Issuer and Root of the certificate are also compliant with the standard defined in the policy.
**: Mandatory fields*	

4. In the **CA details**, select a certificate authority, fill the necessary details, and then click **Save CA Details**.
5. In the **Group selection** section, select a group(s) to apply the policy to all the certificates for the selected group.

> **Note:** Additionally, you can find the preferred group(s) by entering the search key word in the **Search** field. The search key words can be added as Favorites for future use.

6. In the **Compliance Check** section, you can enable the **Perform Compliance check** option to perform an immediate compliance check.

> **Note:** Scheduled Compliance check will run periodically based on the Job scheduler settings.

7. Click **Create Policy**.

Configuring Policy for Amazon CA

To configure an Amazon CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.
3. Refer Configuring Policy Details to configure the following:
   - **Policy Details** section
   - **Group Selection** section
   - **Compliance Check** section
4. On the **CA Policy: Create** page, click **Amazon** under the **CA details** on the left side of the screen.

CA Details for Amazon Policy

Name	Description
***CA Accounts**	The Amazon CA accounts configured in the CA settings screen are listed. Select a CA account from the list to create the policy.
**: Mandatory fields*	

5. Click **Add**.
6. You can use the delete icon against the CA account to delete the configuration.

Certificate Parameter - Field and Description Table

Field	Description
Host Name	Enter the host name.
Allowed Domain Names	As you type the domain name, the matching domain names are displayed. Select the desired domain names.
Common Name	You can provide the common name**.** For example, *.domain.com It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, Regenerate.

Field	Description
	Note: Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.)
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@)

7. Click **Save CA Details** to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against the **Amazon** option to indicate the details are successfully stored.
8. Click **Create Policy**.
   The policy is created and a confirmation message displays.

Configuring Policy for Amazon Private CA

- You must configure the CA setting with Amazon Private CA credentials.
- You must have validated and fetched the Amazon Intermediate CAs along with the issuer region details in the CA settings page.

To configure an Amazon Private CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.
3. Refer Configuring Policy Details section to configure the following,
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
4. On the **CA Policy: Create** page, click **Amazon Private CA** in the **Certificate Authority** pane on the left side of the page.

CA Details - Field and Description Table

Field	Description
***CA Account**	Select the certificate authority account.
***Issuer Region**	Select the issuer region from the dropdown list.
***Issuer Name**	Select the issuer name from the dropdown list.
***Validity**	Enter the validity period for the certificate. The available options are: • **Days** - You can enter more than one validity period in days, to choose one in certificate enrolment. • **Month** - You can enter more than one validity period in Months, to choose one in certificate enrolment. • **Year** - You can enter more than one validity period in Year, to choose one in certificate enrolment.
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one or more than one **Bit Length - Key Type** from the dropdown list. The discovered certificate's Key Type and Bit length will be compared against the selected Bit Length - Key Type(s) to identify if they are complaint with the policy.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>Selected Bit Length - Key Type(s) is enforced while performing any certificate request operations such as New, Renew, Regenerate. The Amazon Private CA supports below Bit type and Length.
<table><tr><th>Type</th><th>Length</th></tr><tr><td>RSA</td><td>2048
4096</td></tr><tr><td>EC</td><td>prime256v1 sec384r1</td></tr></table></td></tr>
<tr><td>*Hash Function</td><td>Supported Hash Function(s) are listed. You can select one or more than one Hash Function from the dropdown list. The supported hash functions are:
• SHA256
• SHA384
• SHA512.</td></tr>
<tr><td>*Signature Algorithm</td><td>Select the SignAlgothirm from the dropdown list. The available options are:
• SHA256WITHECDSA
• SHA384WITHECDSA
• SHA512WITHECDSA
• SHA256WITHRSA
• SHA384WITHRSA
• SHA512WITHRSA.
Note: The Issuer will print the issuer algorithm that the users selected in the Signature Algorithm field.</td></tr>
<tr><td colspan="2">*: Mandatory fields</td></tr>
</table>

5. In the **Certificate parameters** section, enter/select the following details.

Certificate parameters - Field and Description Table

Field	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. Common Name is enforced while performing any certificate request operations such as New, Renew, Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.)
Organization	You can provide the organization's name. The discovered certificate's Subject Organization will be compared against the organization provided in the policy to identify if they are complaints. The organization is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's Subject Organization Unit will be compared against the organization unit provided in the policy to identify if they are Complaint. Organization Unit is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's Locality will be compared against locality provided in the policy to identify if they are complaints. The locality is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's State will be compared against the state provided in the policy to identify if they are complaints. The state is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's Country code will be compared against the country code provided in the policy to identify if they are complaints. Country code is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address.

Field	Description
	The discovered certificate's mail address will be compared against the email address provided in the policy to identify if they are complaints. Mail address is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. Subject Alternative Name is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@) .
**: Mandatory fields*	

6. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **the Amazon** option to indicate the details are successfully stored.
7. Click the **Create Policy** button to create a new policy.

   The policy is created and a confirmation message displays.

Configuring Policy for DigiCert CA

To configure a DigiCert CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.
3. Refer Configuring Policy Details section, to configure the following:
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**

4. On the **CA Policy: Create** page, click digicert in the **Certificate Authority** pane on the left side of the page.

CA Details - Field and Description Table

Field	Description
***CA Account**	The GlobalSign CA accounts configured in CA settings screen are listed. Select a CA account from the list to create the policy.
***Division**	Select the division from the dropdown list.
***Certificate Type**	The **Certificate Types** corresponding to the selected CA account are listed. Select one (or) more **Certificate Type** from the list to create the policy.
***Validity**	Enter the validity period for the certificate. The available options are: **Days** - You can enter more than one validity period in days, to choose one in certificate enrolment. **Month** - You can enter more than one validity period in Months, to choose one in certificate enrolment. **Year** - You can enter more than one validity period in Year, to choose one in certificate enrolment.
**: Mandatory fields*	

5. In the **Vendor Specific Details section**, select/enter the details as listed in the table.

Vendor Specific Details

Field	Description
***Server Type**	Select the server type from the dropdown list.
**: Mandatory fields*	

6. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
7. You can use the **Edit** option in the table to modify the configuration and the **Remove** option to delete the configuration.
8. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve,** and **Hash Function.**
9. You can fill the **Certificate parameters** section based on your organization's policies and standards.

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com

Name	Description
	It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.)
Organization	You can provide the organization's name. The discovered certificate's **SubjectOrganization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints.

Name	Description
	Country code is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN) It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@)
**: Mandatory fields*	

10. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against the **DigiCert** option to indicate the details are successfully stored.
11. Click the **Create Policy** button to create a new policy.
12. The policy is created and a confirmation message displays.

Configuring Policy for EJBCA CA

To configure an EJBCA CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.
3. Refer Configuring Policy Details section to configure the following,

- **Policy Details**
- **Group Selection**
- **Compliance Check**

4. On the **CA Policy: Create** page, click **EJBCA** in the **Certificate Authority** pane on the left side of the page.

Field	Description
***CA Accounts**	The EJBCA accounts configured in the CA settings screen are listed. Select a CA account from the list to create the policy.
**: Mandatory fields*	

5. In the **Vendor Specific Details** section, select/enter the details as listed in the table:

Vendor Specific Details Section - Field and Description Table

Field	Description
End entity user name	Enter the name of the end entity user.
***End entity Profile name**	Enter the name of the end entity profile.
***Issuer Common Name**	Enter the common user name.
***Certificate Profile Name**	Enter a certificate profile name.
**: Mandatory fields*	

6. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
7. You can use the **Remove** option to delete the configuration.
8. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve, and Hash Function**.

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description	Purpose
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down.	The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any

Name	Description	Purpose
		certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate.	The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend using P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported**Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down.	The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*		

9. You can fill the **Certificate parameters** section based on your organization's policies and standards.

Certificate parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.

Name	Description
	Note: Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they

Name	Description
	are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

10. Click the **Save CA Details** button to save the configuration. A green tick mark displays in the **Certificate Authority** pane against the **EJBCA** option to indicate the details are successfully stored.
11. Click **Create Policy** button to create a new policy.
12. The policy is created and a confirmation message displays.

Configuring Policy for Entrust CA

To configure an Entrust CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.
3. Refer Configuring Policy Details section to configure the following,
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
4. On the **CA Policy: Create** page, click Entrust in the **Certificate Authority** pane on the left side of the page.

CA Details - Field and Description Table

Field	Description
***CA Account**	The Entrust CA accounts configured in the CA settings screen are listed. Select a CA account from the list to create the policy.
***Certificate Type**	The **Certificate Types** corresponding to the selected CA account are listed. Select one (or) more **Certificate Type** from the list to create the policy.
***Validity**	Enter the validity period for the certificate. The available options are: **Days** - You can enter more than one validity period in days, to choose one in certificate enrolment. **Month** - You can enter more than one validity period in Months, to choose one in certificate enrolment. **Year** - You can enter more than one validity period in Year, to choose one in certificate enrolment.
**: Mandatory fields*	

5. In the **Vendor Specific Details section**, select/enter the details as listed in the table:

Field	Description
Additional Emails	Enter the valid email address in the field.

6. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
7. You can use the **Edit** option in the table to modify the configuration and **the Remove** option to delete the configuration.
8. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve,** and **Hash Function.**

Bit Length, ECDSA curve, and Hash Function - Field and Descriptions Table

Name	Description	Purpose
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down.	The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate.	The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend to use P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported **Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down.	The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*		

9. You can fill the **Certificate parameters** section based on your organization's policies and standards.

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state.

Name	Description
	The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

10. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **Entrust** option to indicate the details are successfully stored.
11. Click the **Create Policy** button to create a new policy.
12. The policy is created and a confirmation message displays.

Configuring Policy for Entrust MPKI CA

To configure an Entrust MPKI CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.
3. Refer Configuring Policy Details section to configure the following,
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
4. On the **CA Policy: Create** page, click Entrust MPKI in the **Certificate Authority** pane on the left side of the page.
   The following table provides the field description in the **CA Details** section:

CA Details - Field and Description Table

Field	Description
***CA Accounts**	The Entrust CA accounts configured in the CA settings screen are listed. • Select a CA account from the list to create the policy. • The selected CA account will be listed in the **CA Accounts** table.
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***Hash Function**	Supported **Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function** is enforced while performing any certificate request operations such as New, Renew, Regenerate.

Field	Description
*: Mandatory fields	

5. You can fill the **Certificate parameters** section based on your organization's policies and standards. The following table provides the field description under **the Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.) .
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state.

Name	Description
	The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN) It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

6. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **Entrust MPKI** option to indicate the details are successfully stored.

Note: The Pop-up message is displayed as **Entrust MPKI - CA details added.**

7. Refer **Configuring Policy Details** section to configure the following:

- **Group Selection** section
- **Compliance Check** section

8. Click the **Create Policy** button to create a new policy.
9. The policy is created and a confirmation message is displayed.

Configuring Policy for GlobalSign Atlas CA

Before You Begin: The prerequisites for configuring the policy are as follows:

- Certificate Group(s) must be available to map the policy to them.
- CA accounts (settings) must be available to which the policy is going to be created.
- AppViewX permission required (Accounts > Roles - Click here to check Accounts management).

To configure policy for GlobalSign Atlas CA:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** button to configure GlobalSign Atlas custom policy.
3. Refer Configuring Policy Details section to configure the following:
   - Policy Details section
   - Group Selection section
   - Compliance Check section
4. On the **CA Policy: Create** page, click **GlobalSign Atlas** in the **Certificate Authority** pane on the left side of the screen.

   The updated fields for the CA are displayed on the right side of page.

CA Details - Field and Description Table

Field Name	Description
***CA Accounts**	The GlobalSign Atlas CA accounts configured in the CA settings screen are listed here. Select a CA account from the list to create the policy.

Field Name	Description
***Validity**	Enter the validity period for the certificate. The available options are: • **Days** - You can enter more than one validity period in days, to choose one in certificate enrolment. • **Month** - You can enter more than one validity period in Months, to choose one in certificate enrolment. • **Year** - You can enter more than one validity period in Year, to choose one in certificate enrolment.
***Bit Length - Key Type**	All the Key Types are listed with corresponding Bit Length. You can select one (or) more than one Bit Length - Key Type(s) from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected Bit Length - Key Type(s) to identify if they are compliant with the policy. Selected Bit Length - Key Type(s) is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***Hash Function**	Supported Hash Function(s) are listed here. You can select one (or) more Hash Function(s) from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected Hash Function(s) to identify if they are compliant with the policy. Selected Hash Function(s) is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*	

5. Enter the desired values above and click **Add.**

The CA details are saved to the table and the confirmation message is displayed.

6. You can use the **Edit** (pencil)option in the table to modify the configuration and the **Remove** (bin) option to delete the configuration.
7. Enter values in the **Certificate parameters** section based on your organization's policies and standards.

Certificate Parameters - Field and Description Table

Field Name	Description
***Host Name**	Enter the hostname for the certificate. The hostname should not start and end with a dot/full stop (.)
***Allowed Domain Names**	Enter only the white-listed domain names. Press enter after adding the domain name. multiple domain names can be added.
Common Name	You can provide the common name. For example, *.domain.com It helps enforce domains for which a certificate can be requested. Common Name is enforced while performing any certificate request operations such as New, Renew, Regenerate. Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.)
**: Mandatory fields*	

8. Click the **Save CA Details** button.
9. Select groups from the **Group selection** section and indicate for **Compliance check** using the toggle button in the respective section.
10. Click the **Create Policy** button to create a new policy.

    The policy is created and a confirmation message is displayed.

Configuring Policy for GlobalSign CA

To configure a GlobalSign CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.
3. Refer Configuring Policy Details section to configure the following,
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
4. On the **CA Policy: Create** page, click **GlobalSign** in the **Certificate Authority** pane on the left side of the page.
5. In the **Vendor Specific Details** section, select/enter the details as listed in the table.

   Vendor Specific Details - Fields and Description Table

Field	Description
***Incorporating Agency Reg. No**	Enter the agency registration number.
***Designation**	Enter the designation.
***Business Category**	Select the business category from the dropdown list.
**: Mandatory fields*	

6. Click the **Add** button.

   The CA details are saved to the table and the confirmation message displays.

7. You can use the **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.
8. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve,** and **Hash Function**.
   The following table provides the field description in the **CA Details** section:

CA Details - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend using P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported **Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*	

9. You can fill the **Certificate parameters** section based on your organization's policies and standards.

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.)
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Country code	You can provide a country code.

Name	Description
	The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the mail address provided in the policy to identify if they are Complaint. **Mail address** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN) It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

10. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against the GlobalSign option to indicate the details are successfully stored.
11. Click the **Create Policy** button to create a new policy.
12. The policy is created and a confirmation message displays.

Configuring Policy for GlobalSign MSSL CA

To configure a GlobalSign MSSL CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** button to configure GlobalSign MSSL based policy.
3. Refer Configuring Policy Detailssection to configure the following,
   - **Policy Details** section
   - **Group Selection** section
   - **Compliance Check** section
4. On the **CA Policy: Create** page, click **GlobalSign MSSL**in the **Certificate Authority** pane on the left side of the screen.

   The following table provides the field description under **CA Details** section:

CA Details - Field and Description Table

Name	Description
* **CA Accounts**	The GlobalSign MSSL CA accounts configured on the CA settings screen are listed. Select a CA account from the list to create the policy.
* **Product Type**	All Managed SSL Product Types require that the Organization's information and at least one Domain be registered in the Managed SSL account prior to ordering.
* **Signature Algorithm**	Select the **Signature Algorithm** from the drop-down list.
* **MSSL Profile Allowed Domain Name**	Select the **MSSL Profile Allowed Domain Name** from the drop-down list.
* **Validity**	Provide the value and press **Enter.** Enforce **Certificate Validity** period for selected Certificate Type. The certificate validity of GlobalSign MSSL CA is represented in Day(s) and Year(s). One (or) more than one period can be added.
**: Mandatory fields*	

5. Select **CA accounts, Product Type, Signature Algorithm, MSSL Profile Allowed Domain Name, and Validity** under **CA details** section and provide **Validity** period.
6. Click **Add** button. The CA details are saved to the table and the confirmation message is displayed.
7. You can use **Edit** option in the table to modify the configuration and the **Remove** option to delete the configuration.

The following table provides the field description under the **CA Details** section:

Bit Length and ECDSA curve- Field and Description Table

Name	Description
* **Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are compliant with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
* **ECDSA Curve**	The ECDSA curve's values get auto-populated once the details are configured/selected and the **Add** button is clicked. A discovered certificate's key elliptic curves will be compared against the information to identify if there is a complaint. Additionally, the below details will also be enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend using the P256/ P384/ P521 ECDSA curve while enrolling for a certificate.
**: Mandatory fields*	

8. Select **Bit Length -Key Type, ECDSA curve, Hash Function** under **CA details** section.

The following table provides the field description under the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Host Name	You can provide the hostname. The hostname should not start and end with a dot/fullstop(.).
Common Name	You can provide the common name**.**For example, *.domain.com It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, Regenerate. Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide organization name.

Name	Description
	The discovered certificate's **SubjectOrganization** will be compared against the organization provided in the policy to identify if there are complaints. **Organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organizational unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if there is a Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are Complaint. **Locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if there is a complaint. **State** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if there are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide the organization unit mail address. The discovered certificate's **mail address** will be compared against the mail address provided in the policy to identify if there is a complaint. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. .
Subject Alternative Name	You can provide the subject's alternative name (SAN)

Name	Description
	It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

9. You can fill in the **Certificate parameters** section based on your organization's policies and standards.
10. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against the **GlobalSign MSSL**option to indicate the details are successfully stored.
11. Click **Create Policy** button to create a new policy.
12. The policy is created and a confirmation message is displayed.

Configuring Policy for GoDaddy CA

To configure a GoDaddy CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

3. Refer Configuring Policy Details section to configure the following,
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
4. On the **CA Policy: Create** page, click GoDaddy in the **Certificate Authority** pane on the left side of the page.

CA Details - Field and Description Table

Field	Description
***CA Account**	The GoDaddy CA accounts configured in CA settings screen are listed. Select a CA account from the list to create the policy.
***Certificate Type**	The **Certificate Types** corresponding to the selected CA account are listed. Select one (or) more **Certificate Type** from the list to create the policy.
***Validity**	Enter the validity period for the certificate. The available options are: **Days** - You can enter more than one validity period in days, to choose one in certificate enrolment. **Month** - You can enter more than one validity period in Months, to choose one in certificate enrolment. **Year** - You can enter more than one validity period in Year, to choose one in certificate enrolment.
**: Mandatory fields*	

5. In the **Vendor Specific Details section**, select/enter the details as listed in the table:

Vendor Specific Details section - Field and Descripion Table

Field	Description
***Slot Size**	Select the size of the slot from the dropdown list.
**: Mandatory fields*	

6. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
7. You can use **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.
8. Select **Bit Length -Key Type, ECDSA curve, Hash Function** in the **CA details** section.

CA Details - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down.

Name	Description
	The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend using P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported**Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*	

9. You can fill **Certificate parameters** section based on your organization's policies and standards.

   The following table provides the field description in the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name**.** For example, *.domain.com

Name	Description
	It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are

Name	Description
	complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

10. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against the **GoDaddy** option to indicate the details are successfully stored.
11. Click **Create Policy** button to create a new policy.
12. The policy is created and a confirmation message displays.

Configuring Policy for Google CA

To configure a Google CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Click **+ Create** on the top-right of the page.

3. Refer Configuring Policy Details section to configure the following,
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
4. On the **CA Policy: Create** page, click **Google** in the **Certificate Authority** pane on the left side of the page.

CA Details - Field and Description Table

Field	Description
***CA Accounts**	The Google CA accounts configured in the CA settings screen are listed. Select a CA account from the list to create the policy.
***Issuer Location**	The **Issuer Location** corresponding to the selected CA account is listed. Select an **Issuer Location** from the list to create the policy.
***Issuer Name**	The **Issuer Name** corresponding to the selected issuer location is listed. Select an **Issuer Name** from the list to create the policy.
***Validity**	Provide the value and press **Enter.** Enforce **Validity** period for selected Issuer Name. The validity for Google CA can be represented in Day(s)/ Month(s)/ Year(s). One (or) more than one **Validity** period can be added.
**: Mandatory fields*	

5. In the CA details section, select the **Bit Length -Key Type(s), ECDSA curve(s), and Hash Function(s)**.

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.

Name	Description
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend to use P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported **Hash Function(s)** are listed. You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*	

6. You can fill the **Certificate parameters** section based on your organization's policies and standards.

   For the Policy Enforcement Type = **Strict**

   For the Policy Enforcement Type = **Suggestive**

   The following table provides the field description in the **Certificate parameters** section:

   Certificate Parameters - Field and Description Table

Name	Description
Hostname	This text field is displayed if the **Policy Enforcement Type = Strict** or **Suggestive**. Enter the unique name or label for the host. The field is **mandatory** only when the **Policy Enforcement Type = Strict**.

Name	Description
	Note: The hostname should not start or end with a dot.
Allowed Domain Name	This text field is displayed if the **Policy Enforcement Type = Strict** or **Suggestive**. The field is **mandatory** only when the **Policy Enforcement Type = Strict**. Enter the valid domain name (two parts separated by a dot, such as example.com) .
Blocked Domain Name	This text field is displayed only if the **Policy Enforcement Type = Suggestive**. Enter the domain names (two parts separated by a dot, such as example.com) that need to be blocked. .
Common Name	This is a non-editable field. It has comma-separated values of possible domain names which are allowed. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality.

Name	Description
	The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, and Renew, Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are Complaint. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

7. You can use the **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.
8. A green tick mark will be displayed in the **Certificate Authority** pane against **the Google** option to indicate the details are successfully stored.

9. Click the **Create Policy** button to create a new policy.
10. The policy is created and a confirmation message displays.

Configuring Policy for HashiCorp Vault CA

Before You Begin

- Certificate Group(s) must be available to map the Policy to them.
- CA accounts (settings) must be available to which the policy is going to be created.
- Key Algorithm, Encryption Type must be available under the CA accounts.
- AppViewX permission required (Accounts > Roles - Click here to check Accounts management).

To configure a HashiCorp Vault CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.
3. On the **CA Policy: Create** page, enter details in the **Policy Details** section as described in the tables below.

Policy Details - Field and Description Table

Field Name	Description
***Policy Name**	Provide a unique name to identify the CA policy name. **NOTE**: No special characters other than '.', '-', and '_' are allowed. The name should not start with special characters.
Description	Provide a description of the policy.
Type	Select Strict (or) Suggestive. By default, Strict is selected. • **Strict** - This enforces the standards defined in the policy where a user cannot modify any parameters. • **Suggestive** - This suggests users with policy parameters. A user can modify suggested values if required.

Field Name	Description
Approval Required	When enabled, it will enforce the peer approval process for any requests made for new/renew/regenerate/reissue or revocation of certificates. Peer approving the request is defined in the approval workflow
Private Key Access	When enabled allows the user to download private keys from the holistic view.
Include Root and Intermediate certificates for compliance check	Enabling the option would validate if the Issuer and Root of the certificate are also compliant with the standard defined in the policy.
**: Mandatory fields*	

4. In the **CA Details** section, a list box **Certificate Authority** on the left displays all the available CAs. Select **Hashicorp Vault**.
5. Update the following fields in the **CA Details** section as described in the table below.

CA Details - Field and Description Table

Field Name	Description
***CA Settings**	The dropdown contains the names of the accounts created.
***Secret Engine**	The single-select dropdown contains all the secret engines associated with the account. In a secret engine, a role describes an identity with a set of permissions, groups, or policies you want to attach to a user of the secret engine. User identity is often mapped to a specific role. Hence, a single secret engine needs to be selected to populate Role (below) specific to it.
***Role**	The dropdown contains all the roles mapped to the secret engine.
**: Mandatory fields*	

6. Click the **Add** button.

   The **HashiCorp Vault CA Details** table below the Add button displays the selected values in the dropdown.

   Multiple values can be configured based on the available CA settings and secret engines with different *Bit Length - Key Type* and *Hash Function*. The supported values are as follows:

a. **Key Type**: RSA, EC

b. **Bit Length:**

i. *RSA key type*: 2048 (default), 3072, or 4096

ii. *EC key type*: 224, 256 (default), 384, or 521

c. **Hash Function**: SHA-256, SHA-384, SHA-512

The CA Details table has options to View, Edit, and Delete.

a. To view the CA details, click the View link in the View column - The CA account details are displayed in a pop-up window with the *Bit Length - Key Type* and *Hash Function.*

b. To update the CA details, select the edit or pencil icon in the Edit column.

c. To delete the CA details, select the bin or delete icon.

7. Update the Certificate Parameters in the CA details section as described in the table below.

Note: The parameters are useful during certificate enrollment. One can pre-populate the listed certificate parameters when setting-up/requesting a certificate.

Certificate Parameters - Field and Description Table

Field Name	Description
Common Name	Enter the fully qualified domain name (FQDN) or common name that exactly matches the web browser. **NOTE**: The only special characters allowed are the asterisk (*), hyphen (-), and period (.).
Organization	The name of the organization.
Organization Unit	The name of the organization unit.
Locality	The city in which the organization is located.
State	The state in which the organization is located.
Country	Country in which the organization is located.
Email	The email Ids of the organization contact. **NOTE**: Multiple comma-separated values are allowed. *Example*: admin@email.com, user@email.com

Field Name	Description
Subject Alternative Name	Enter the additional hostnames such as alternative websites or IP addresses that have to be protected with a single SSL certificate. It helps enforce additional domains for which a certificate can be requested. Subject Alternative Name is enforced while performing certificate request operations such as New, Renew, and Regenerate. **NOTE**: The only special characters allowed are the asterisk (*), hyphen (-), period (.), and the at sign (@).

Note: The discovered certificate's parameters (Organization, Organization Unit, Locality, State, Country, and Email) will be compared against the same parameters provided in the policy to identify if they are complaints. They are enforced while performing any certificate request operations such as New, Renew, and Regenerate.

**: Mandatory fields*

8. Click the **Save CA Details** button.

   A green tick mark is displayed in the Certificate Authority pane against the HashiCorp Vault option to indicate the details are successfully saved.

9. In the **Group Selection**, select one or more groups to map to the policy. Refer to the **Certificate Group** section to add/update groups.
10. Under the **Compliance Check** section, enable the **Perform Compliance Check** option to perform an immediate compliance check.
11. Click the **Create Policy** button.
    The Policy is created successfully.

Configuring Policy for Hydrant ID CA

To configure the Hydrant ID CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.

3. Refer Configuring Policy Details section to configure the following,
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
4. On the **CA Policy: Create** page, click **Hydrant ID** in the **Certificate Authority** pane on the left side of the page.
5. In the **Vendor Specific Details section**, select the challenge type from the dropdown list.
6. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
7. You can use the **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.
8. Select **Bit Length -Key Type, ECDSA curve, Hash Function in the CA details** section.
   The following table provides the field description in the **CA Details** section:

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend to use P256/ P384/ P521 ECDSA curve while enrolling.

Name	Description
***Hash Function**	Supported **Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*	

9. You can fill the **Certificate parameters** section based on your organization's policies and standards. The following table provides the field description in the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced

Name	Description
	while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.

Name	Description
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

10. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **the Hydrant ID** option to indicate the details are successfully stored.
11. Click the **Create Policy** button to create a new policy.
    The policy is created and a confirmation message displays.

Configuring Policy for Let's Encrypt CA

To configure the Let's Encrypt CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.
3. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

4. Refer Configuring Policy Details section to configure the following:

- **Policy Details**
- **Group Selection**
- **Compliance Check**

5. To configure a policy with LetsEncrypt details, click **LetsEncrypt** in the **Certificate Authority** pane on the left side of the page.
6. In the **Vendor Specific Details section**, select the challenge type from the dropdown list.
7. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
8. You can use the **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.
9. Select **Bit Length -Key Type, ECDSA curve, Hash Function in the CA details** section.
   The following table provides the field description in the **CA Details** section:

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description	Purpose
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.	
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA**	

Name	Description	Purpose
	curve from the drop-down. for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend to use P256/ P384/ P521 ECDSA curve while enrolling.	
***Hash Function**	Supported **Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.	
**: Mandatory fields*		

10. You can fill the **Certificate parameters** section based on your organization's policies and standards. The following table provides the field description in the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state.

Name	Description
	The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

11. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **the LetsEncrypt** option to indicate the details are successfully stored.

12. Click the **Create Policy** button to create a new policy.
    The policy is created and a confirmation message displays.

Configuring Policy for Microsoft Enterprise CA

To configure the Microsoft Enterprise CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.
3. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

4. Refer Configuring Policy Details section to configure the following:
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
5. To configure a policy with Microsoft Enterprise details, click **Microsoft Enterprise** in the **Certificate Authority** pane on the left side of the screen.
   The following table provides the field description under **CA Details** section:

   CA Details - Field and Description Table

Name	Description
***CA Accounts**	The Microsoft Enterprise CA accounts configured in the CA settings screen are listed. Select a CA account from the list to create the policy.
***MS Template List**	The MS template(s) configured for the selected CA account are listed. Select MS template(s) from the list to associate with the policy.
**: Mandatory fields*	

6. Select **CA accounts** and **MS Template List** under **CA details** section.
7. Click **Add** button. The CA details are saved to the table and the confirmation message displays.
8. You can use **the Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.

9. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve, and Hash Function.**

   The following table provides the field description in the **CA Details** section:

   Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend to use P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported **Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*	

10. You can fill the **Certificate parameters** section based on your organization's policies and standards.

    The following table provides the field description in the **Certificate parameters** section:

    Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.

Name	Description
	Note: Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing

Name	Description
	any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN) It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

11. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **the Microsoft Enterprise** option to indicate the details are successfully stored.
12. Click the **Create Policy** button to create a new policy.
13. The policy is created and a confirmation message displays.

Configuring Policy for Microsoft Standalone CA

To configure the Microsoft Standalone CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.
3. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

4. Refer Configuring Policy Details section to configure the following:
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
5. To configure a policy with Microsoft Standalone details, click **Microsoft Standalone** in the **Certificate Authority** pane on the left side of the screen.

   The following table provides the field description in the **CA Details** section:

CA Details - Field and Description Table

Name	Description
***CA Accounts**	The Microsoft Standalone accounts configured in CA settings screen are listed. Select a CA account from the list to create the policy.
**: Mandatory fields*	

6. Select **CA accounts** in the **CA details** section.
7. Click **Add** button. The CA account is saved to the table and confirmation message displays.
8. You can use the **Remove** option to delete the configuration.
9. In the **CA details** section, select the **Bit Length -Key Type, ECDSA curve, and Hash Function**.

   The following table provides the description of other fields in the **CA Details** section:

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any

Name	Description
	certificate request operations such as New, Renew, and Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend to use P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported**Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
**: Mandatory fields*	

10. You can fill the **Certificate parameters** section based on your organization's policies and standards.

    The following table provides the field description under **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will

Name	Description
	only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address.

Name	Description
	The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are Complaint. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

11. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **Microsoft Standalone** option to indicate the details are successfully stored.
12. Click the **+ Create Policy** button to create a new policy.
13. The policy is created and a confirmation message displays.

Configuring Policy for Nexus CA

To configure the Nexus CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.

3. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

4. Refer Configuring Policy Details section to configure the following:
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
5. To configure a policy with Nexus details, click **Nexus** in the **Certificate Authority** pane on the left side of the screen.
6. Select **CA accounts and Certificate Type** under the **CA details** section and provide **the Validity** period.

   The following table provides the field description under **CA Details** section:

   CA Details - Field and Description Table

Name	Description
***CA Accounts**	The Nexus CA accounts configured in the CA settings screen are listed. Select a CA account from the list to create the policy.
***Certificate Type**	The **Certificate Types** corresponding to the selected CA account are listed. Select one (or) more **Certificate Type** from the list to create the policy.
***Validity**	Provide the value and press **Enter.** Enforce **Validity** period for selected Certificate Type(s). The validity for Nexus CA can be represented in Day(s)/ Month(s)/ Year(s). One (or) more than one **Validity** period can be added.
**: Mandatory fields*	

7. Once the CA account and the validity fields are populated, a new section called **Vendor Specific Details** is enabled. Select the **Procedure** field from the drop-down list and click **Add** to save the CA details to the table. You can also use the **Remove** option to delete the configuration.

   Select the details as follows:

   a. **Case 1 -** Select **Server** check box, the procedures listed in the **Procedures** dropdown will be - Mapped to servers and default procedures.
   b. **Case 2 -** Select **Client** check box, the procedures listed in the **Procedures** dropdown will be - mapped to client and default procedures.
   c. **Case 2 -** Select **Server** and **Client** check box, the procedures listed in the **Procedures** dropdown will be - mapped to server, client, and default procedures.

Note: The procedures displayed in the dropdown should have the **cert type** appended in the value. For example: - *Procedure name_Server/client/default/Code-Signing*

8. Click the **Add** button. The CA details are saved to the table and the confirmation message displays.
9. You can use the **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.
10. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve**, and **Hash Function**.
    The following table provides the field description under **CA Details** section:

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend using P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported **Hash Function(s)** are listed. You can select one (or) more than one **Hash Function(s)** from the drop-down.

Name	Description
	The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
**: Mandatory fields*	

11. You can fill the **Certificate parameters** section based on your organization's policies and standards.

    The following table provides the field description in the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit.

Name	Description
	The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate.

Name	Description
	Note: Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
*: Mandatory fields	

12. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against the **Nexus** option to indicate the details are successfully stored.
13. Click **Create Policy** button to create a new policy.
14. The policy is created and a confirmation message displays.

Configuring Policy for OpenTrust CA

To configure the OpenTrust CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.
3. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

4. Refer Configuring Policy Details section to configure the following:
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
5. To configure a policy with OpenTrust details, click OpenTrust in the **Certificate Authority** pane on the left side of the page.
   The following table provides the field description in the **CA Details** section.

Field	Description
***CA Account**	The OpenTrust CA accounts configured in CA settings screen are listed. Select a CA account from the list to create the policy.
***Certificate Management Profile**	Select the certificate management profile from the dropdown list.
*Zone	Select the zone from the dropdown list.
**: Mandatory fields*	

6. In the **Profile Parameters** section, select/enter the details as listed in the table.

Profile Parameters Section - Field and Description Table

Field	Description
***Common Name**	Enter the common name for the policy.
Organizational Unit	Enter the organizational unit.
Organization	Enter the name of the organization.
**: Mandatory fields*	

7. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
8. You can use the **Remove** option to delete the configuration.
9. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve, and Hash Function.**
   The following table provides the description of other fields in the **CA Details** section:

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down. The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any

Name	Description
	certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down. for a certificate. The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend to use P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported**Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down. The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*	

10. You can fill the **Certificate parameters** section based on your organization's policies and standards. The following table provides the field description under the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name**.** For example, *.domain.com It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing

Name	Description
	any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. The **organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. The **locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaint. The **state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code.

Name	Description
	The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are Complaint. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN) It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

11. Click the **Save CA Details** button to save the configuration. A green tick mark displays in the **Certificate Authority** pane against the **OpenTrust** option to indicate the details are successfully stored.
12. Click **Create Policy** button to create a new policy.
13. The policy is created and a confirmation message displays.

Configuring Policy for Symantec CA

To configure the Symantec CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.
3. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

4. Refer Configuring Policy Details section to configure the following:
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
5. To configure a policy with Symantec details, click **Symantec** in the **Certificate Authority** pane on the left side of the page.
   The following table provides the field description in the **CA Details** section.

CA Details - Field and Description Table

Field	Description
***CA Account**	The Symantec CA accounts configured in CA settings screen are listed. Select a CA account from the list to create the policy.
***Certificate Type**	The **Certificate Types** corresponding to the selected CA account are listed. Select one (or) more **Certificate Type** from the list to create the policy.
***Validity**	Enter the validity period for the certificate. The available options are: **Days** - You can enter more than one validity period in days, to choose one in certificate enrolment. **Month** - You can enter more than one validity period in Months, to choose one in certificate enrolment. **Year** - You can enter more than one validity period in Year, to choose one in certificate enrolment.
**: Mandatory fields*	

6. In the **Vendor Specific Details section**, select the server type from the dropdown list.
7. Click the **Add** button.
   The CA details are saved to the table and the confirmation message displays.
8. You can use the **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.

9. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve,** and **Hash Function.**
   The following table provides the field description in the **CA Details** section:

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description	Purpose
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down.	The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down for a certificate.	The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend using P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported**Hash Function(s)** are listed**.** You can select one (or) more than one **Hash Function(s)** from the drop-down.	The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*		

10. You can fill the **Certificate parameters** section based on your organization's policies and standards.
    The following table provides the field description in the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

Name	Description
Common Name	You can provide the common name. For example, *.domain.com. It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints. **The state** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.

Name	Description
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are Complaint. **Mail address** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

11. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against **the Symantec** option to indicate the details are successfully stored.
12. Click the **Create Policy** button to create a new policy.
13. The policy is created and a confirmation message displays.

Configuring Policy for Trustwave CA

To configure the Trustwave CA policy:

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.

   Note: KUBE+ is packaged with default policies they are Default and Certificate-Gateway.

2. Create a custom policy by clicking the **Create** button on the upper right corner of the CA Policy page.
3. Click **+ Create** on the top-right of the page.

   The **CA Policy: Create** page is displayed.

4. Refer Configuring Policy Details section to configure the following:
   - **Policy Details**
   - **Group Selection**
   - **Compliance Check**
5. To configure a policy with Trustwave details, click **Trustwave** in the **Certificate Authority** pane on the left side of the screen.
6. Select **CA accounts and Certificate Type** under the **CA details** section and provide **the Validity** period.
   The following table provides the field description under **CA Details** section:

   CA Details - Field and Description Table

Name	Description
***CA Accounts**	The Trustwave CA accounts configured in the CA settings screen are listed. Select a CA account from the list to create the policy.
***Certificate Type**	The **Certificate Types** corresponding to the selected CA account are listed. Select one (or) more **Certificate Type** from the list to create the policy.
***Validity**	Provide the value and press **Enter.** Enforce **Validity** period for selected Certificate Type(s). The validity for Trustwave CA can be represented in Day(s)/ Month(s)/ Year(s). One (or) more than one **Validity** period can be added.
**: Mandatory fields*	

7. Click the **Add** button. The CA details are saved to the table and the confirmation message displays.
8. You can use the **Edit** option in the table to modify the configuration and **Remove** option to delete the configuration.
9. In the **CA details** section, select **Bit Length -Key Type, ECDSA curve, and Hash Function.**

The following table provides the field description under **CA Details** section:

Bit Length, ECDSA curve, and Hash Function - Field and Description Table

Name	Description	Purpose
***Bit Length - Key Type**	All the **Key Types** are listed with corresponding **Bit Length.** You can select one (or) more than one **Bit Length - Key Type(s)** from the drop-down.	The discovered certificate's Key Type and Bit length will be compared against the selected **Bit Length - Key Type(s)** to identify if they are complaint with the policy. Selected **Bit Length - Key Type(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
***ECDSA curve**	When **Key Type** is selected as EC, **ECDSA curve** corresponding to selected **Key Type** is listed. You can select one (or) more than one **ECDSA curve** from the drop-down for a certificate.	The discovered certificate's Key elliptic curves will be compared against the selected **ECDSA curve(s)** to identify if they are complaint with the policy. Selected **ECDSA curve(s)** is enforced while performing certificate request operations such as New, Renew, and Regenerate. We recommend using P256/ P384/ P521 ECDSA curve while enrolling.
***Hash Function**	Supported **Hash Function(s)** are listed. You can select one (or) more than one **Hash Function(s)** from the drop-down.	The discovered certificate's Key Hash Algorithm will be compared against the selected **Hash Function(s)** to identify if they are complaint with the policy. Selected **Hash Function(s)** is enforced while performing any certificate request operations such as New, Renew, Regenerate.
**: Mandatory fields*		

10. You can fill the **Certificate parameters** section based on your organization's policies and standards.

    The following table provides the field description in the **Certificate parameters** section:

Certificate Parameters - Field and Description Table

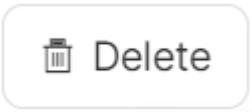
Name	Description
Common Name	You can provide the common name. For example, *.domain.com It helps enforce domains for which a certificate can be requested. **Common Name** is enforced while performing any certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.).
Organization	You can provide the organization's name. The discovered certificate's **Subject Organization** will be compared against the organization provided in the policy to identify if they are complaints. **The organization** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Organization Unit	You can provide an organization unit. The discovered certificate's **Subject Organization Unit** will be compared against the organization unit provided in the policy to identify if they are Complaint. **Organization Unit** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Locality	You can provide a locality. The discovered certificate's **Locality** will be compared against the locality provided in the policy to identify if they are complaints. **The locality** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
State	You can provide state. The discovered certificate's **State** will be compared against the state provided in the policy to identify if they are complaints.

Name	Description
	The state is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Country code	You can provide a country code. The discovered certificate's **Country code** will be compared against the country code provided in the policy to identify if they are complaints. **Country code** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Email	You can provide an organization unit mail address. The discovered certificate's **mail address** will be compared against the email address provided in the policy to identify if they are complaints. **Mail address** is enforced while performing any certificate request operations such as New, Renew, and Regenerate.
Subject Alternative Name	You can provide the subject alternative name (SAN). It helps enforce additional domains for which a certificate can be requested. **Subject Alternative Name** is enforced while performing certificate request operations such as New, Renew, and Regenerate. **Note:** Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.), At (@).
**: Mandatory fields*	

11. Click the **Save CA Details** button to save the configuration. A green tick mark will be displayed in the **Certificate Authority** pane against the **Trustwave** option to indicate the details are successfully stored.
12. Click **Create Policy** button to create a new policy.
13. The policy is created and a confirmation message displays.

Deleting a Policy

To delete a policy,

1. Go to **menu** > **KUBE+** > **Groups & Policies** > **CA Policy**.
   On the **CA Policy** page, the configured policies are displayed, if any.
2. Select a policy from the list.
3. Click 

.
4. On the **Confirmation** popup, click **Yes**.

Certificate Authority (CA) Integration

Service Mesh can be integrated with your Enterprise Internal PKI only for signing application workloads with mTLS certificates.

AppViewX supports integrating with EJBCA and Microsoft CA for signing the mTLS service mesh workloads.

- Custom CA
- AppViewX CA
- AppViewX PKIaaS CA
- Amazon and Amazon Private CA
- Segito CA
- DigiCert CA
- DigiCert MPKI
- EJBCA CA
- Entrust
- Entrust MPKI
- GlobalSign MSSL CA
- GlobalSign Atlas CA
- GoDaddy CA
- Google CA
- Certificate Authority Service CA
- HashiCorp Vault CA

- InCommon CA
- Let's Encrypt CA
- Microsoft Enterprise CA
- Nexus
- OpenTrust CA
- Symantec CA
- Trustwave CA

Custom CA

- Before you Begin
- Configuring Custom CA

Before you Begin

Following are the prerequisites for configuring Custom CA account in AppViewX:

- A logo to use it for the custom CA.
- An optional CA certificate and key to be used as a root certificate.

Configuring Custom CA

To configure the custom CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***Custom CA Name**	A unique name to identify the CA name. **Note:** No special characters allowed.
***Upload Custom CA Logo**	Upload a logo for the custom CA. This logo will appear in the product representing the custom CA.

Name	Description
Custom CA Certificate	Upload a certificate for the custom CA. This certificate will become the root certificate. **Note:** The <.*pfx*> and <.*p12*> are certificate types are supported.
**: Mandatory fields*	

3. Once the logo and certificate are uploaded, the entered CA will appear in the CA list with the logo presented.
4. Once the logo is added, users can click **Configure Now** to input the CA details.
5. Update the following details in the **General Information** section as described in the table:

Name	Description
***Name**	Client authentication certificate for API communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

6. Update the following details in the **ROOT CSR parameters** section as described in the table:

Root CSR - Field and Description Table

Name	Description
Common Name	The common name of the root certificate. **Note:** • Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. • Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.)
Algorithm	Type of the root certificate.
Hash Function	The hash function for the root certificate.

Name	Description
Organization Unit	Name of the Organisation unit.
Key Length	Key length for the root certificate.
Organization	Organization attribute for the root certificate.
Locality	Locality attribute for the root certificate.
State or Province	State attribute for the root certificate.
Country	Country attribute for the root certificate.
Email Address	Email address for the root certificate.
**: Mandatory fields*	

7. Update the following details in the **Root Validity** section as described in the table.

Name	Description
***Start Date**	Start date of the certificate issuance.
***End Date**	End date of the certificate issuance.
**: Mandatory fields*	

8. Click **Save**.

   Once the setting is saved, the user will be directed to the root certificate submission holistic view.
9. Users can submit and fetch the root certificate.
10. On the CA setting page user can see the status of the created setting.

AppViewX CA

- Configuring AppViewX CA
- Validating AppViewX CA

Configuring AppViewX CA

To configure the AppViewX CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Select **AppViewX** from the left-side vendor list.
   The **AppViewX** home page is displayed.
3. Click the **Configure Now** or **+Add** icon from the middle or top-right of the page respectively.

   Note: The **Configure Now** option is displayed if you are configuring a CA for the first time.

   The **AppViewX** configuration page is displayed.
4. Enter/Select the following details in the **General Information** section:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. Example: Server, Client
CRL Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Enter/Select the following **Credentials**-related information:

Credentials - Field and Description Table

Field	Description
Credential type*	From the dropdown list, from the following options, select the credential type: • **Manual Entry**: Manually enter the access and secret key for the customer's AWS account)
Access key*	Enter the access key for the customer's AWS account.

Field	Description
	Note: This field is displayed only when **Credential** type is set to **Manual Entry**.
Secret key*	Enter the secret key for the customer's AWS account. **Note:** This field is displayed only when **Credential** type is set to **Manual Entry**.
**: Mandatory fields*	

6. Update the following details in the **CSR Parameters** section as described in the table:

CSR Parameters - Field and Description Table

Name	Description
Common Name	The common name of the root certificate. **Note:** • Use Asterisk (*) for the host part of the FQDN to enforce the domain. For example, *.domain.com will only allow users to request certificates with domain domain.com. • Allowed Special Characters: Asterisk (*), Hyphen (-), Period (.)
Issuer Name	Name of the certificate issuer.
Algorithm	Type of the root certificate.
Hash Function	The hash function for the root certificate.
Organization Unit	Name of the Organisation unit.
Key Length	Key length for the root certificate.
Organization	Organization attribute for the root certificate.
Locality	Locality attribute for the root certificate.

Name	Description
State or Province	State attribute for the root certificate.
Country	Country attribute for the root certificate.
Email Address	Email address for the root certificate.
**: Mandatory fields*	

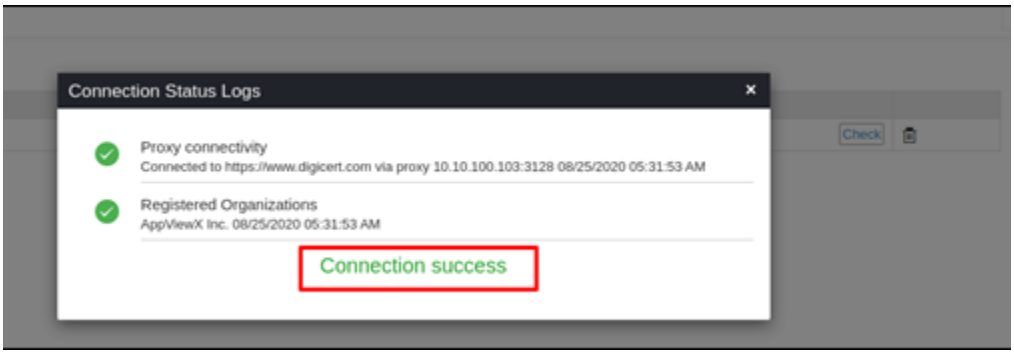
7. Update the following details in the **Validity** section as described in the table.

Name	Description
***Start Date**	Start date of the certificate issuance.
***End Date**	End date of the certificate issuance.
**: Mandatory fields*	

8. Click **Save**.
   Once the setting is saved, the user will be directed to the root certificate submission holistic view.
9. Users can submit and fetch the root certificate.
10. On the CA setting page, user can see the status of the created setting.

Validating AppViewX CA

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **AppViewX** in the left side vendor list.
3. Click **Check** to validate the CA setting that is created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



AppViewX PKIaaS CA

Configuring AppViewX PKIaaS CA

To configure the AppViewX PKIaaS CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **AppViewX PKIaaS** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table.

General Information - Field and Description Table

Name	Description
***Name**	A unique name to identify the CA setting. No special characters other than '.', '-','_' are allowed. The name should not start with special characters.
***Purpose/ Usage**	Certificate Type for which CLM actions will be enabled. For example, server and clients
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.

Name	Description
Data Center	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table.

CA Configuration Section - Field and Description Table

Options	Description
***Region**	Region of the certificate.
***Configure With**	The method of certificate type to upload.
***Upload JSON**	Option to upload the certificate file.
***Email Address**	Email address of the user.
***Project Id**	ID of the project.
**: Mandatory fields*	

6. Select **Fetch Procedures**.

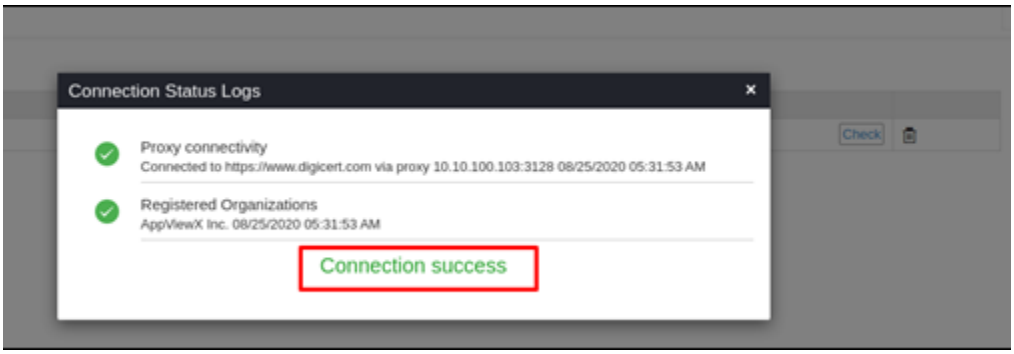
Note: The procedures available in the AppViewX PKIaaS CA account will be fetched and listed for the specific user.

7. To map the fetched procedures, click on one or many and click the **Actions** dropdown:
   a. **CASE 1** - If the user selects Server only in Purpose and Usage, then the fetched procedure by default will be of server/client both. The Action dropdown will only have - **Map as Server**. and **MAP as Default**
   b. **CASE 2** - If the user selects Client only in Purpose and Usage, then the fetched procedure by default will be of server/client both. The Action dropdown will only have - **Map as Client**. and **MAP as Default**
   c. **CASE 3** - If the user selects Server and Client both in Purpose and Usage, then the fetched procedure by default will be of server/client both. The Action dropdown will have both the actions **Map as Client** , **Map as Server**, and **MAP as Default**
8. Click **Save**.

Validating AppViewX PKIaaS CA

Once the AppViewX PKIaaS settings are added, the validation must be done to check whether the connection between AppViewX and AppViewX PKIaaS is configured properly.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **AppViewX PKIaaS** in the left side vendor list.
3. Click **Check** to validate the CA setting that is created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



Amazon and Amazon Private CA

- Before you Begin
- Configuring Amazon CA
- Configuring Amazon Private CA
- Validating Amazon

Before you Begin

Following are the prerequisites for configuring Amazon CA or Amazon Private CA account in AppViewX

- An Amazon account for a user having necessary access for enrolling the certificates and other CLM operations.
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure the proxy. **https://adminguide.appviewx.com/proxy-4**

- Policy JSON for AWS Ec2 Instance Certificate Management.
- Prerequisite for Amazon CA:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "ssm:SendCommand",
      "ssm:DescribeDocument",
      "ec2:DescribeInstances",
      "ec2:DescribeRegions",
      "s3:ListBucket",
      "ssm:CreateDocument",
      "ssm:GetCommandInvocation",
      "s3:GetObject",
      "s3:ListAllMyBuckets",
      "ssm:DescribeInstanceInformation",
      "ssm:GetDocument",
      "s3:DeleteObject",
      "s3:GetBucketLocation"
    ],
    "Resource": "*"
  }
 ]
}


// Policy JSON for Certificate Management in AWS Classic and Application LoadBalancers:
{
 "Version": "2012-10-17",
 "Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "iam:GetServerCertificate",
```

```
      "elasticloadbalancing:DescribeLoadBalancers",
      "elasticloadbalancing:ModifyListener",
      "elasticloadbalancing:DescribeListeners",
      "acm:GetCertificate",
      "ec2:DescribeRegions",
      "elasticloadbalancing:DescribeTargetHealth",
      "acm:ImportCertificate",
      "elasticloadbalancing:SetLoadBalancerListenerSSLCertificate",
      "iam:UploadServerCertificate"
    ],
    "Resource": "*"
  }
 ]
}

// Policy JSON for Certificate Management in AWS Cloudfront:
{
 "Version": "2012-10-17",
 "Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeRegions",
      "cloudfront:ListDistributions",
      "cloudfront:UpdateDistribution",
      "cloudfront:GetDistributionConfig"
    ],
    "Resource": "*"
  }
 ]
}

// Policy JSON for IAM Certificate Management:
{
 "Version": "2012-10-17",
 "Statement": [
```

```
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "iam:GetServerCertificate",
      "iam:UpdateServerCertificate",
      "iam:ListServerCertificates",
      "ec2:DescribeRegions",
      "iam:UploadServerCertificate"
    ],
    "Resource": "*"
  }
 ]
}

// Policy JSON for ACM Certificate Management:
{
 "Version": "2012-10-17",
 "Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "acm:DescribeCertificate",
      "acm:RequestCertificate",
      "acm:GetCertificate",
      "ec2:DescribeRegions",
      "acm:ListCertificates",
      "acm:ImportCertificate"
    ],
    "Resource": "*"
  }
 ]
}

// Prerequisite for Amazon Private CA. Policies and Permissions required for AWS IAM User:
{
```

```
"Version": "2012-10-17",
"Statement": [
 {
  "Sid": "VisualEditor0",
  "Effect": "Allow",
  "Action": [
   "s3:PutObject",
   "s3:GetObjectAcl",
   "s3:GetObject",
   "s3:PutObjectAcl"
  ],
  "Resource": [
   "arn:aws:s3:::<bucketname>",
   "arn:aws:s3:::<bucketname>/*"
  ]
 },
 {
  "Sid": "VisualEditor1",
  "Effect": "Allow",
  "Action": [
   "acm-pca:GetCertificate",
   "ec2:DescribeRegions",
   "acm-pca:GetCertificateAuthorityCertificate",
   "acm-pca:RevokeCertificate",
   "acm:RenewCertificate",
   "acm-pca:ListCertificateAuthorities",
   "acm-pca:DescribeCertificateAuthorityAuditReport",
   "acm-pca:CreateCertificateAuthorityAuditReport",
   "s3:ListAllMyBuckets",
   "acm:DescribeCertificate",
   "acm-pca:IssueCertificate",
   "acm:RequestCertificate",
   "acm:GetCertificate",
   "acm:ListCertificates",
   "acm-pca:DescribeCertificateAuthority"
  ],
  "Resource": "*"
```

```
  }
 ]
}

// AWS Simple Storage Service (S3) Bucket Policy for parsing Audit log:
{
 "Version": "2012-10-17",
 "Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "acm-pca.amazonaws.com"
    },
    "Action": [
      "s3:PutObject",
      "s3:PutObjectAcl",
      "s3:GetBucketAcl",
      "s3:GetBucketLocation"
    ],
    "Resource": [
      "arn:aws:s3:::bucket_name/*",
      "arn:aws:s3:::bucket_name"
    ]
  }
 ]
}
```

Configuring Amazon CA

To configure the Amazon CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Select **Amazon** from the left-side vendor list.
   The **Amazon** home page is displayed.
3. To configure Amazon CA, click the **ACM CA** tab from the home page.
4. Click the **Configure Now** or **+Add** icon from the middle or top-right of the page respectively.

Note: The **Configure Now** option is displayed if you are configuring a CA for the first time.

The **Amazon** configuration page is displayed.

5. Enter/Select the following details in the **General Information** section:

General Information - Field and Description Table

Field	Description
***Account Type**	From the dropdown list, select one of the following account types: • **Standalone** (Traditional access key- and secret key-based communication) • **Cross or Federated** (Authentication using assume role)
***Account Name**	Unique name for the certificate authority (CA) account represented during certificate enrollment and policy creation
***Account Number**	Valid AWS account number
Account Description	Additional information related to the CA account being configured
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. The available options are, • Server • Client.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
***Default Region**	Default region for API communication
***Data Center (AppViewX's CA agent)**	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

6. Enter/Select the following **Credentials**-related information:

Credentials - Field and Description Table

Field	Description
Credential type*	From the dropdown list, from the following options, select the credential type: • **Manual Entry**: Manually enter the access and secret key for the customer's AWS account)
Access key*	Enter the access key for the customer's AWS account. **Note:** This field is displayed only when **Credential** type is set to **Manual Entry**.
Secret key*	Enter the secret key for the customer's AWS account. **Note:** This field is displayed only when **Credential** type is set to **Manual Entry**.
**: Mandatory fields*	

7. Enter/Select the following details in the **Discover resources** section:

Discover Resources - Field and Description Table

Field	Description
Role ARN for Resource Discovery*	**Note:** This field is displayed only when **Account Type** is **Cross or Federated**. To let the master account assume role for the child account (get temporary privileges to discover resources from the child account), configure the role ARN for resource discovery: a. Click  . b. Enter the following details:

Field	Description
Role Session name	**Role Session name** is an identifier for the assumed role session.

Field	Description
	<table><tr><th>Field</th><th>Description</th></tr><tr><td></td><td>Use the **Role Session name** to uniquely identify a session when the same rule is assumed by different principals or for different reasons.</td></tr><tr><td>**Duration Seconds**</td><td>Enter the duration, in seconds, for which the credentials should remain valid. Acceptable durations for IAM user sessions: • **Minimum**: 900 seconds (15 minutes) • **Maximum**: 129,600 seconds (36 hours) **Default**: 3600 seconds (1 hour)</td></tr><tr><td>**External Id**</td><td>**External Id** is a unique identifier that might be required when you assume a role in another account.</td></tr><tr><td>**Source Identity**</td><td>The source identity is specified by the principal that is calling the **AssumeRole** operation.</td></tr><tr><td>**Session Tags**</td><td>**Session Tags** are key-value pairs that you pass when you assume an IAM role or federate a user in AWS STS. To create a session tag: i. In the **Enter Key** field, enter a key for the key-value pair. ii. In the **Enter Value** field, enter a value for the key-value pair. iii. Click **Add**. The added key-value pair is shown in the table below the fields.</td></tr></table>

Field	Description
Service Region*	To select a service region: a. To fetch the service regions for the account information provided, click **Fetch Region**. The retrieved service regions are populated in the **Select the Region(s)** dropdown list. b. From the **Select the Region(s)** dropdown list, select the required service region.
Discover Certificate	To enable instant certificate discovery at the time of device addition, select this checkbox.
Cert Sync*	Select from one of the following options: • **Managed**: AppViewX will connect with the customer's AWS account and discover certificates. These certificates will be added to the inventory. Users with the relevant permissions can then perform the required certificate-related actions. • **Monitored**: AppViewX will connect with the customer's AWS account and discover certificates. These certificates will be added to the inventory where the users will be allowed to only view the certificates. • **Ignored**: AppViewX will connect with the customer's AWS account but certificate discovery will be disabled.
Auto Sync	To enable/disable automatic schedule-based synchronization: a. For **Auto Sync**, select the **Yes** checkbox. b. For **Schedule based discovery**, use the two dropdown lists to select a duration. For example, to schedule the auto sync after every 2 days, from the first dropdown list, select **2** and from the second dropdown list, select **Days**. By default, the auto sync is set to **1 Hours**. **Note:** The **Schedule based discovery** dropdown lists are displayed only when **Auto Sync** is enabled.
Route53 Zone Auto Approval	To support DNS validation as an automatic process, enable this toggle. **Important:** If Route53 has been configured for any of the older Amazon Public CAs, ensure that, after migration, the zones are manually updated.
**: Mandatory fields*	

Configuring Amazon Private CA

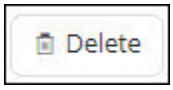
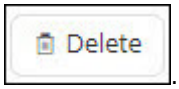
To configure the Amazon Private CA:

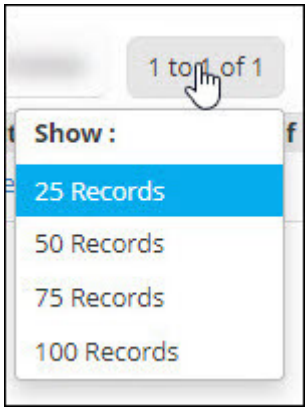
1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select **Amazon** from the left-side vendor list.
   The **Amazon** home page is displayed.
3. To configure Amazon CA, click the **AWS Private CA** tab from the home page.

   The **Amazon** home page is updated to display the inventory grid as shown in the image. In the inventory grid for the Amazon Private CA, master and child account details are logged as separate entries, instead of having just one master entry.

   Fields in the inventory grid are explained in the table below:

AWS Private CA - Screen Description Table

Field	Description
Search	Use the **Search** field to search for accounts, by entering the value of one of the details listed in the inventory grid.
	To delete one or more accounts: a. From the inventory grid, select the checkbox corresponding to the account(s) you want to delete. b. Click  . **Tip:** To delete all accounts listed in the inventory grid, select the checkbox in the grid header.
	To set the number of records that should be displayed on one page: a. Click  . b. From the **Show** menu displayed, select the required value.

Field	Description
	
	If the inventory grid spans more than one page, use this control to navigate the pages, one page at a time.
Account Name	This is the unique name for the Certificate Authority (CA) account entered at the time of account creation.
Account Number	AWS account number
Account Type	**Multi account**: Indicates that the account is a cross account **Single account**: Indicates that the account is a standalone account
CA Status	For an account, after all configuration details for Amazon Private CA are entered, you will be required to click the **Fetch issuer and save** button to sync and discover the issuers and the respective certificates for that account. The **CA Status** field shows the current status of this sync and discovery process. Possible values for this field are: • Completed • In progress **Note:** An account entry in the grid will be disabled till the **CA Status** is **In progress**.

Field	Description
Connection Status	To validate if connection has been established with the CA, click **Check**. If a connection has been established, this field is updated to display **Success** or **Failure**.
No. of Issuers	This field displays the number of issuers associated with the account. **Note:** For a master account, this field will show the number of issuers associated with only the master account. The value does not include the number of issuers associated with the child account.
**: Mandatory fields*	

4. To add an account, click **Configure Now** (if you are creating your first account) or click 

from the top-right corner of the screen.
The **Amazon** page is updated to display fields for entering the CA configuration-related information.
5. On this screen, enter the following **Basic Information**:

Basic Information - Field and Description Table

Field	Description
Account type*	From the dropdown list, from the following options, select the customer's AWS account type: • **Standalone**: The user account and the resources are available in the same account. • **Cross or Federated**: Resources are available across multiple accounts and users are given role-based access.
Account name*	Enter a unique name for the Certificate Authority (CA) account that will be used during certificate enrollment and policy creation.
Account number*	Enter the customer's AWS account number.
Account Description	Enter any additional details related to the account, if required.

Field	Description
Purpose/ Usage*	From the dropdown list, select the purpose of the certificate that can be requested using this account.
Proxy Required	To allow all communication to the Certificate Authority (CA) to use the proxy details (provided in general settings; refer the Platform User Guide for more details), select this checkbox.
Default Region*	From the dropdown list, select the default region for API communication.
Data Center (AppViewX's CA Agent)	From the dropdown list, select the data center that will be used to establish communication with the Certificate Authority (CA)
**: Mandatory fields*	

6. Enter the following **Credentials**-related information:

Credentials - Field and Description Table

Field	Description
Credential type*	From the dropdown list, from the following options, select the credential type: • **Manual Entry**: Manually enter the access and secret key for the customer's AWS account)
Access key*	Enter the access key ID for the customer's AWS account. The access key and the secret access key (entered in the following field) are used together to authenticate requests. **Note:** This field is displayed only when **Credential** type is set to **Manual Entry**.
Secret key*	Enter the secret access key ID for the customer's AWS account. The access key (entered in the previous field) and the secret access key are used together to authenticate requests. **Note:** This field is displayed only when **Credential** type is set to **Manual Entry**.

Field	Description
Credential name*	If the customer's AWS credentials are stored in CyberArk, from the dropdown list, select the CyberArk credential name. **Note:** This field is displayed only when **Credential** type is set to **Credential List - CyberArk**.
**: Mandatory fields*	

7. In the **Discover resources** section, enter the following details:

Discover Resources - Field and Description Table

Field	Description
Role ARN for Resource Discovery*	**Note:** This field is displayed only when **Account Type** is **Cross or Federated**. To let the master account assume role for the child account (get temporary privileges to discover resources from the child account), configure the role ARN for resource discovery: a. Click  . b. Enter the following details:

Field	Description
Role Session name	**Role Session name** is an identifier for the assumed role session. Use the **Role Session name** to uniquely identify a session when the same rule is assumed by different principals or for different reasons.
Duration Seconds	Enter the duration, in seconds, for which the credentials should remain valid. Acceptable durations for IAM user sessions:

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>• Minimum: 900 seconds (15 minutes)
• Maximum: 129,600 seconds (36 hours)
Default: 3600 seconds (1 hour)</td></tr>
<tr><td>External Id</td><td>External Id is a unique identifier that might be required when you assume a role in another account.</td></tr>
<tr><td>Source Identity</td><td>The source identity is specified by the principal that is calling the AssumeRole operation.</td></tr>
<tr><td>Session Tags</td><td>Session Tags are key-value pairs that you pass when you assume an IAM role or federate a user in AWS STS.
To create a session tag:
i. In the Enter Key field, enter a key for the key-value pair.
ii. In the Enter Value field, enter a value for the key-value pair.
iii. Click Add.
The added key-value pair is shown in the table below the fields.</td></tr>
</table>
</td></tr>
<tr><td>Service Region*</td><td>Service regions are regions that are supported by the selected service.
To select a service region:
a. To fetch the service regions for the account information provided, click Fetch Region. The retrieved service regions are populated in the Select the Region(s) dropdown list.
b. From the Select the Region(s) dropdown list, select the required service region.</td></tr>
<tr><td>CA Operation Mode*</td><td>From the following options, select one/both operation mode(s) for discovering all the certificates enrolled by the Private Certificate Authority:</td></tr>
</table>

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>• ACM Private CA
• AWS Certificate Manager (ACM)</td></tr>
<tr><td>S3 Bucket*</td><td>Note: This field is displayed only when the ACM Private CA operation mode is selected.

Enter the S3 bucket name.</td></tr>
<tr><td>Role ARN for S3 Bucket</td><td>Note: This field is displayed only when the ACM Private CA operation mode is selected for a Cross or Federated account.

a. Click  .
The ARN Advanced Settings action pane is displayed.
b. In the ARN Advanced Settings action pane, enter the following details:
<table><tr><th>Field</th><th>Description</th></tr><tr><td>Role Session name*</td><td>Role Session name is an identifier for the assumed role session.
Use the Role Session name to uniquely identify a session when the same rule is assumed by different principals or for different reasons.</td></tr><tr><td>Duration Seconds</td><td>Enter the duration, in seconds, for which the credentials should remain valid.
Acceptable durations for IAM user sessions:
• Minimum: 900 seconds (15 minutes)
• Maximum: 129,600 seconds (36 hours)
Default: 3600 seconds (1 hour)</td></tr></table></td></tr>
</table>

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>External Id</td><td>External Id is a unique identifier that might be required when you assume a role in another account.</td></tr>
<tr><td>Source Identity</td><td>The source identity is specified by the principal that is calling the AssumeRole operation.</td></tr>
<tr><td>Session Tags</td><td>Session Tags are key-value pairs that you pass when you assume an IAM role or federate a user in AWS STS.
To create a session tag:
i. In the Enter Key field, enter a key for the key-value pair.
ii. In the Enter Value field, enter a value for the key-value pair.
iii. Click Add.
The added key-value pair is shown in the table below the fields.</td></tr>
</table>
c. Click Apply.
</td></tr>
<tr><td>Discover Certificate</td><td>To enable instant certificate discovery at the time of device addition, select this checkbox.</td></tr>
<tr><td>CA Sync*</td><td>Select from one of the following options:
• Managed: AppViewX will connect with the customer’s AWS account and discover certificates. These certificates will be added to the inventory. Users with the relevant permissions can then perform the required certificate-related actions.
• Monitored: AppViewX will connect with the customer’s AWS account and discover certificates. These certificates will be added to the inventory where the users will be allowed to only view the certificates.
• Ignored: AppViewX will connect with the customer’s AWS account but certificate discovery will be disabled.</td></tr>
</table>

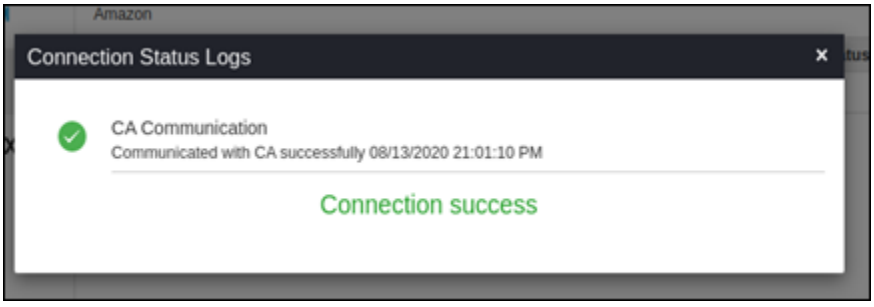
Field	Description
Auto Sync	To enable/disable automatic synchronization, use the **Auto Sync** key. If **Auto Sync** is enabled, to set the frequency of the schedule-based sync: a. From the first dropdown list, select the interval between two schedule-based syncs. b. From the second dropdown, select a unit for the interval (**Hours**/**Days**). For example, to set the frequency of the schedule-based sync to every 2 hours, from the first dropdown list, select **2** and from the second dropdown list, select **Hours**.
**: Mandatory fields*	

8. Click **Fetch issuer and save**.
   - AppViewX will now discover all the Private CA Certificate Authorities across the selected region(s).
   - The inventory grid on the Amazon CA home page will be populated with the properties and details retrieved from this discovery.

Validating Amazon

Once the Amazon settings are added, you need to validate the connection between AppViewX and Amazon, to make sure that the connection is properly configured.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. On the **Amazon** home page, select **Amazon** or **Amazon Private CA**.
3. Click **Check** to validate the CA setting that is created.
   This action validates the CA communication and the **connection status** is displayed as either **Connection success** or **Failure**.



Segito CA

Configuring Segito

To configure the Segito CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Segito** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. Example: Server, Client
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the Segito CA APIs for Certificate Management:

CA Configuration - Field and Description Table

Name	Description
***Base URL**	This URL will contain just the hostname of the Segito CA instance. For example, <https://www.segito.com>

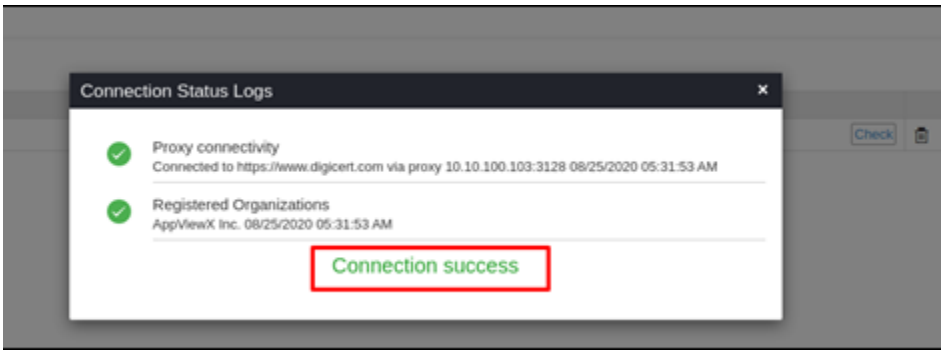
Name	Description
	Note: vendorSpecificSettings.url - invalid URL.
***Credential Type**	Select the type of credential as desired from the dropdown list. The available options are, • Manual EntryCredential • List - CyberArk.
***Credential List**	Select the required credential from the dropdown list. **Note:** This field will be enabled if the Credential Type is selected as Credential List - CyberArk.
Account ID	Account id details of Segito CA Account.
***API Key**	API key specific to the CA account. This API key should have required permission to make API Calls. Space is not allowed.
Auto Approve	Enable **the Auto Approve** option if all CLM requests from AppViewX do not need to be approved from Segito CA Account.
**: Mandatory fields*	

6. Select **Fetch Divisions and Certificate Types**.
   The Division and Certificate types available in the Segito CA account will be fetched.
7. Click **Save.**

Validating Segito Connection

Once the Segito settings are added, the validation must be done to check whether the connection between AppViewX and Segito is configured properly.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Segito** in the left side vendor list.
3. Click **Check** to validate the CA setting that is created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



DigiCert CA

Before you begin

Following are the prerequisites for configuring DigiCert CA account in AppViewX:

- A DigiCert CertCentral Account with **Administrator** role Access.
- **API Key** configured in DigiCert with required permissions to make API Requests from AppViewX.
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure a proxy. https://adminguide.appviewx.com/proxy-4

Configuring DigiCert

To configure the DigiCert CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **DigiCert** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. Example: Server, Client
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the DigiCert CA APIs for Certificate Management:

CA Configuration - Field and Description Table

Name	Description
***Base URL**	This URL will contain just the hostname of the DigiCert CA instance. For example, <https://www.digicert.com> **Note:** vendorSpecificSettings.url - invalid URL.
***Credential Type**	Select the type of credential as desired from the dropdown list. The available options are, • Manual EntryCredential • List - CyberArk.
***Credential List**	Select the required credential from the dropdown list. **Note:** This field will be enabled if the Credential Type is selected as Credential List - CyberArk.

Name	Description
Account ID	Account id details of DigiCert CA Account, which can be found under account manager details in DigiCert CertCentral Account.
***API Key**	API key specific to the CA account. This API key should have required permission to make API Calls. Space is not allowed.
Auto Approve	Enable **the Auto Approve** option if all CLM requests from AppViewX do not need to be approved from DigiCert CA Account.

Note: Note: Auto approval checkbox is optional and features work only for **one-step certificate requests** configured in the DigiCert Cert Central Account.

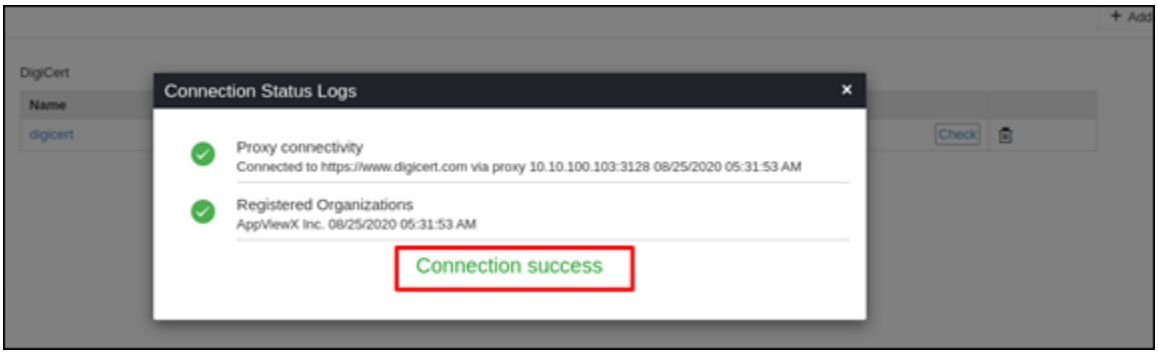
**: Mandatory fields*

6. Select **Fetch Divisions and Certificate Types**.
   The Division and Certificate types available in the DigiCert CA account will be fetched.
7. Click **Save.**

Validating DigiCert Connection

Once the DigiCert settings are added, the validation must be done to check whether the connection between AppViewX and DigiCert is configured properly.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **DigiCert** in the left side vendor list.
3. Click **Check** to validate the CA setting that is created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



DigiCert MPKI

The following are the prerequisites for configuring a Digicert MPKI CA account in AppViewX:

- A Digicert MPKI Account with **Administrator** role Access.
- An **API Key** configured in Digicert MPKI with required permissions to make API Requests from AppViewX.
- The AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure the proxy. https://adminguide.appviewx.com/proxy-4

- Configuring DigiCert MPKI CA
- Validating DigiCert MPKI Connection

Configuring DigiCert MPKI CA

To configure the DigiCert MPKI CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **DigiCert MPKI** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table.

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting.

Name	Description
	No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. Example: Server, Client
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the DigiCert CA APIs for Certificate Management.

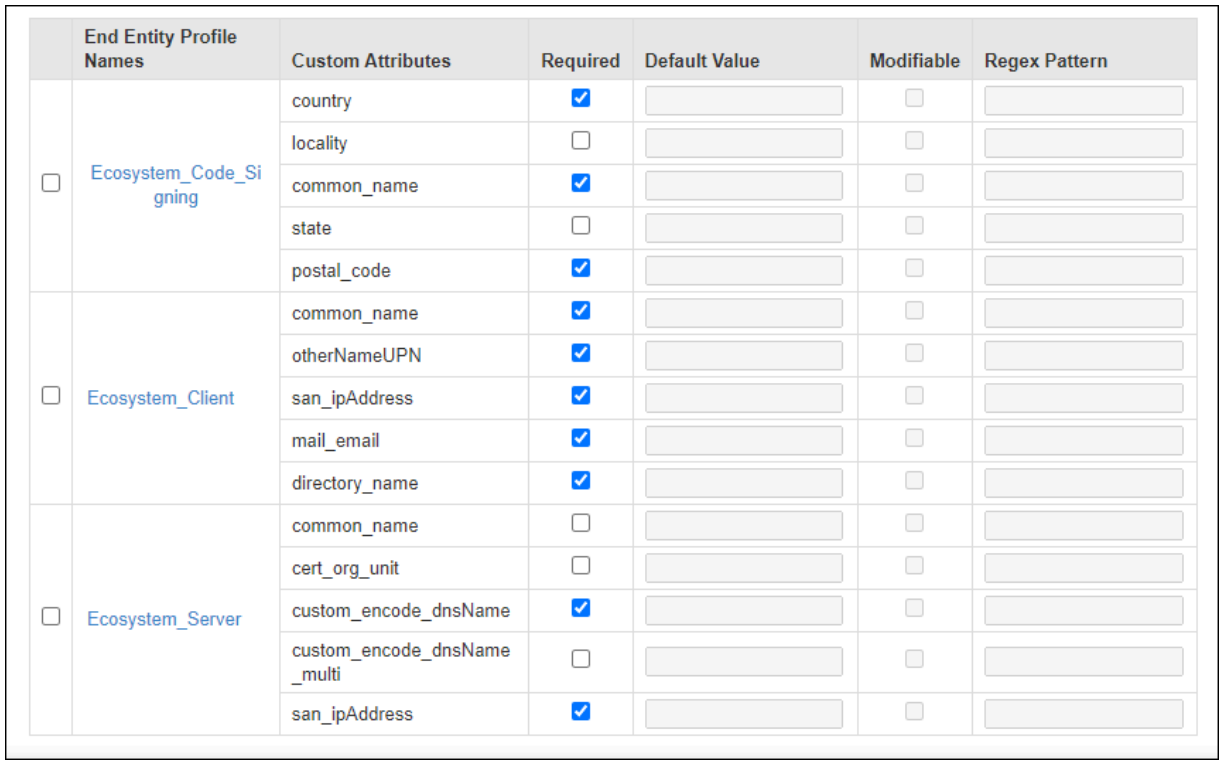
CA Configuration - Field and Description Table

Name	Validation
***Base URL**	
***Seat ID**	NA.
***API Key**	

Note:

- **: Mandatory fields*
- Auto approval checkbox is optional and features work only for **one-step certificate requests** configured in the DigiCert Cert Central Account.

6. Select **Fetch Divisions and Certificate Types.**
   The Division and Certificate types available in the DigiCert CA account will be fetched and listed for the specific API key user in the table as shown below.



	End Entity Profile Names	Custom Attributes	Required	Default Value	Modifiable	Regex Pattern
☐	Ecosystem_Code_Signing	country	☑		☐	
		locality	☐		☐	
		common_name	☑		☐	
		state	☐		☐	
		postal_code	☑		☐	
☐	Ecosystem_Client	common_name	☑		☐	
		otherNameUPN	☑		☐	
		san_ipAddress	☑		☐	
		mail_email	☑		☐	
		directory_name	☑		☐	
☐	Ecosystem_Server	common_name	☐		☐	
		cert_org_unit	☐		☐	
		custom_encode_dnsName	☑		☐	
		custom_encode_dnsName_multi	☐		☐	
		san_ipAddress	☑		☐	

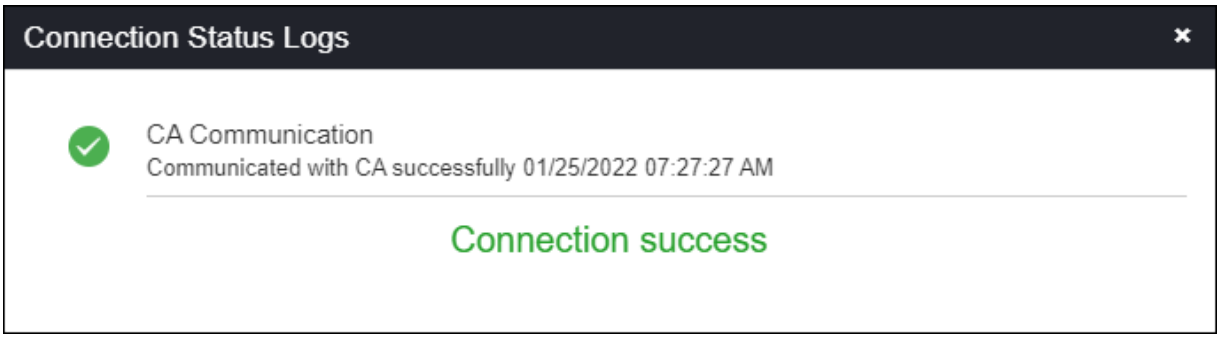
7. Click **Save.**

Note: The pop-up message is displayed as <CA_name> Settings Added.

Validating DigiCert MPKI Connection

Once the DigiCert MPKI settings are added, the validation must be done to check whether the connection between AppViewX and DigiCert MPKI is configured properly.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **DigiCert** in the left side vendor list.
3. Click **Check** to validate the CA setting that is created.



The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.

EJBCA CA

Before you begin

Following are the prerequisites for configuring EJBCA account in AppViewX:

- An EJBCA client certificate for a user having the necessary access for enrolling the certificates and other CLM operations.
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings.

Configuring EJBCA

To configure the EJBCA CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **EJBCA** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Options	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/ Usage**	Certificate Type for which CLM actions will be enabled. Example: Server, Client.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the APIs for Certificate Management.

CA Configuration - Field and Description Table

Options	Description
***Client Authentication**	Client authentication certificate for API communication. • Enter the valid password once the **Authentication Details** window is displayed. • Click **OK**. **Note**: Must be a valid <.*p12*> or <.*pfx*> file.
***URL**	EJBCA URL
***Discover by expiry days**	To get all the certificates that are expired and valid for specified days. **Note:** Must be a number.
End entity profile names	Required end entity profiles for CA setting.
Custom attributes	Required custom attributes for the specific end entity profile.

Options	Description
	Note: Validation can be added by the user in the regex box.
**: Mandatory fields*	

6. Click **Validate and Fetch**.
   The **End entity profiles** available for the CA account will be fetched along with the certificate profile from the **Certificate Authority**.
7. Update the following details in the **Certificate Attributes** section as described in the table:

Certificate Attributes Section - Field and Description Table

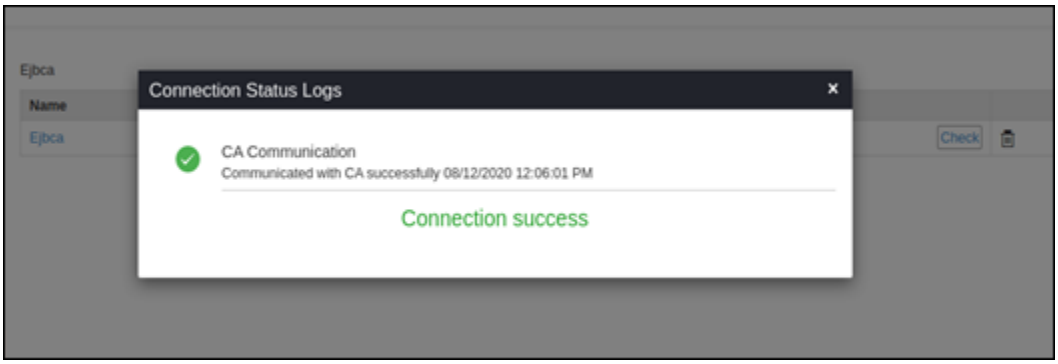
Options	Description
***End Entry Profile Names**	Select the profile that is used in the certificate enrollment from the dropdown list.
Custom Attributes	Select the list attributes configured in CA to enroll certificates.
Note: • **: Mandatory fields* • Custom attributes should be configured as exactly as it is available in the EJBCA portal.	

8. Click **Save**.

Validating EJBCA

Once the EJBCA settings are added, validation needs to be done to check whether the connection between AppViewX and EJBCA is properly configured.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **EJBCA** in the left side vendor list.
3. Click **Check** to validate the CA setting that has been created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



Entrust

Before You Begin

Following are the prerequisites for configuring Entrust CA account in AppViewX:

- An Entrust client authentication certificate and credentials having necessary access for CLM actions.
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure the proxy.

Configuring Entrust

To configure the Entrust CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Entrust** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting.

Name	Description
	Note: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. For example: Server and Client
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the Entrust CA APIs for Certificate Management.

CA Configuration - Field and Description Table

Name	Description
***Client Authentication**	The client authentication certificate from Entrust for API communication. **Note**: Must be a valid <*.p12*> file. To generate an CSR within AppViewX refer to Generating a CSR and download the CSR. Further, upload the CSR to the Entrust homepage as described in section - XXXXX.
***Base URL**	This URL will contain just the hostname of the Entrust CA instance. The value is https://api.entrust.net/enterprise/v2
User Name	Enter the API Username to communicate with the CA.

Name	Description
Password	Enter the API Password to communicate with the CA.
Auto Approve	Select the checkbox to avoid queuing of new certificates in the CA portal.
**: Mandatory fields*	

6. Update the following details in the **Advanced Settings** section as described in the table.

Advanced Settings - Field and Description Table

Name	Description
Poll after CSR Submission	A check box field when selected will fetch the certificated immediately after CSR Submission on enrollment, renew, and reissue of certificate with the retry count and retry frequency as described below.
***Retry Count**	The number of times the polling will take place after CSR submission. Enter a value between 1 and 10.
***Retry Frequency**	The duration of the polling. enter the value between 1 and 30seconds
**: Mandatory fields*	

7. Click **Fetch Custom Attributes**.
   The attributes available for the CA account will be fetched from the Certificate Authority along with the CA and profile names.

Note: The pop-up message is displayed as **CA and profiles fetched**.

8. Click **Save**.
   The created Entrust configuration settings will be added.

Note: The pop-up message is displayed as **<CA_name> Settings Added**.

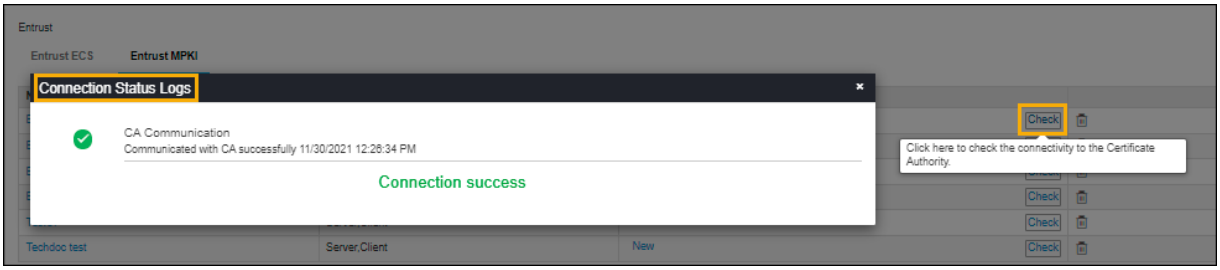
Validating Entrust CA

Once the Entrust settings are added, validation needs to be done to check whether the connection between AppViewX and Entrust is properly configured.

To validate the Entrust connection,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Entrust** in the left side vendor list.
3. Click **Check** to validate the CA setting that has been created.

   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



Entrust MPKI

- Before you begin
- Configuring Entrust MPKI
- Validating Entrust MPKI Connection

Before you begin

Following are the prerequisites for configuring Entrust MPKI CA account in AppViewX:

- An Entrust client authentication certificate and credentials having necessary access for CLM actions.
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure the proxy.

Configuring Entrust MPKI

To configure the Entrust MPKI CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Entrust** in the left side vendor list, and then select the **Entrust MPKI** tab.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. For example: Server and Client
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the Entrust MPKI CA APIs for Certificate Management.

CA Configuration - Field and Description Table

Name	Description
***Client Authentication**	Client authentication certificate for API communication. **Note**: Must be a valid <*.p12*> file.

Name	Description
***Base URL**	This URL will contain just the hostname of the Entrust CA instance. For example, https://api.entrust.net/enterprise/v2
**: Mandatory fields*	

6. Click **Fetch CA and Profile Names.**
   The attributes available for the CA account will be fetched from the Certificate Authority along with the CA and profile names.

   Note: The pop-up message is displayed as **CA and profiles fetched**.

7. Click **Save**.
   The created Entrust MPKI configuration settings will be added.

   Note: The pop-up message is displayed as **<CA_name> Settings Added**.

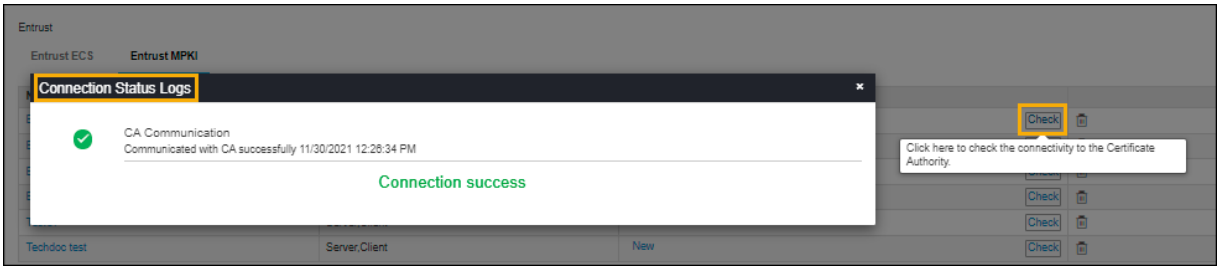
Validating Entrust MPKI Connection

Once the Entrust settings are added, validation needs to be done to check whether the connection between AppViewX and Entrust is properly configured.

To validate the Entrust MPKI connection,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Entrust MPKI** in the left side vendor list.
3. Click **Check** to validate the CA setting that has been created.

   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



GlobalSign MSSL CA

Configuring GlobalSign MSSL CA

To configure the GlobalSign MSSL CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **GlobalSign** in the left side vendor list, and then click the **GlobalSign MSSL** tab.
4. Update the following details in the **General Information** section as described in the table.

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. No special characters other than '.', '-','_' are allowed. The name should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. For example, server and clients
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table.:

CA Configuration Section - Field and Description Table

Options	Description
***SSL URL**	Base URL of the SSL API

Options	Description
***User Name**	Provide a username of the GCC to communicate with the CA.
***Password**	Provide a password for the GCC to communicate with the CA.
**: Mandatory fields*	

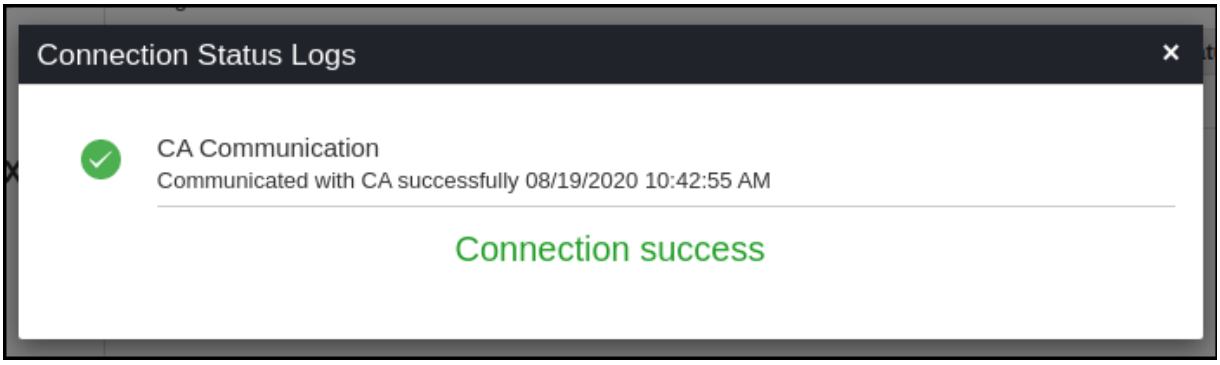
6. Once all the details are configured, click **Save**.
7. In GlobalSign MSSL, we can now fetch profiles and domains by clicking on the **Fetch Profiles and Domain** button.

> **Note:** The supported CSR key types are RSA **2048-8192, ECC P-256, ECC P-384** .

Validating GlobalSign MSSL

Once the GlobalSign settings are added, validation needs to be done to check whether the connection between AppViewX and GlobalSign is properly configured.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **GlobalSign** in the left side vendor list, and then click the **GlobalSign MSSL** tab.
3. Click **Check** to validate the CA setting that is created.
4. CA communication will be validated and the Connection Status will be shown as either Success or Failure.



Limitations

Case/ Ticket number	Fix Description
CA Setting Update	Users need to click on the **Cancel** button once the MSSL domain/profile. ID details are fetched from the GlobalSign MSSL account. If the user clicks the Update button, MSSL domain/profile ID details will be removed from the associated policy. The steps to follow to update CA settings are as follows: 1. On the GlobalSign MSSL CA settings page, after adding/editing values, click the Update button. 2. Navigate back to updated CA settings and click the **Fetch Profiles and Domain** button. 3. Click the **Cancel** button instead of Update to bypass the existing issue.
Default CA policy mapping	The default CA policy is defined with all available values selected and validity data is mapped based on commonly used validity. Hence, it will not have values equivalent to API documents or CA portals. This can be modified or updated in the application accordingly to the default CA policy if changes are required.
Email Address	The email address provided in the email address field on the enrollment page is not considered as the primary email value during CLM actions, instead, the email address field defined in the contact information of the logged-in user will be used. The help info message besides the Email address field on enroll/edit page is as – “If the user email address is configured, that will be used for GlobalSign CA approval actions. If the user email is not configured, then the email address provided in this field will be used" - the second part is not valid anymore.

GlobalSign Atlas CA

Configuring GlobalSign Atlas

Before you Begin

The prerequisites for configuring GlobalSign Atlas CA are

- Login and password to access AppViewX.
- Base URL, API Key, API Secret Key.
- A client certificate provided by the GlobalSign Atlas team.

To configure the GlobalSign Atlas CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **GlobalSign** in the left side vendor list, and then select the **GlobalSign Atlas** tab.
4. If creating a CA for the first time click **Configure Now** or click **+ Add** button on the top right.
5. Update the following details in the **General Information** section as described in the table.

General Information - Field and Description Table

Field Name	Description
***API Credential Friendly name**	Enter the API Credentials Friendly name (which is the CA Account name that will be used for the CA Policy and Enrollment).
***Purpose/Usage**	Select the purpose of the certificate that can be requested using this account. **NOTE**: Users can select *Server*, *Client* or both.
Proxy Required	Select the checkbox if communication to the Certificate Authority (CA) has to use the proxy details provided in the general settings
Data Center (AppViewX's CA agent)	Select the data center that will be used to establish the communication with the CA.
**: Mandatory fields*	

6. Update the following details in the **CA Configuration** section as described in the table.

CA Configuration - Field and Description Table

Field Name	Description
***Base URL**	Enter the base URL required for constructing the API request.
***API Key**	Enter the API key which is the unique identifier used to authenticate a user.

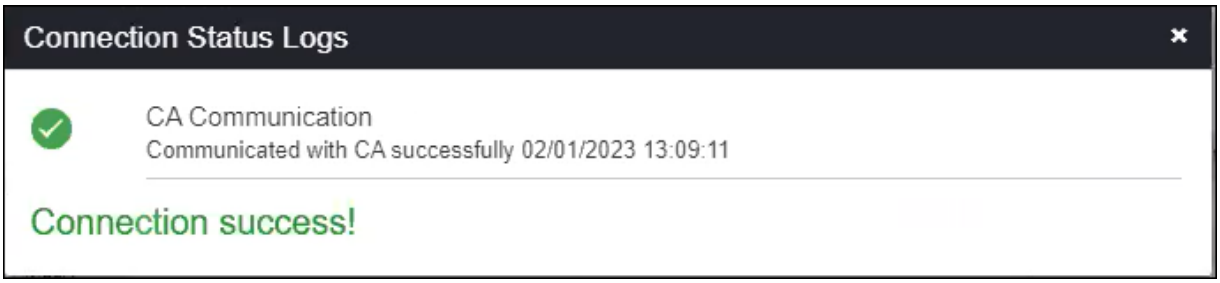
Field Name	Description
	NOTE: The API Key will be displayed as asterisks (*)
***API Secret**	Enter the API secret to communicate with the CA. **NOTE**: The API Secret will be displayed as asterisks (*)
***Client Authentication**	Upload the certificate for client authentication in the .p12 or .pfx format only. The uploaded file format is invalid. Please upload in PKCS12 (.p12 or .pfx) file format only.
**: Mandatory fields*	

7. Click the **Fetch Validation Policy and Save** button.
   A confirmation message will appear “Validation Policy fetched and settings have been updated.” and the CA is created successfully. The connection status for the CA is displayed as New.

Validating GlobalSignAtlas

Once the GlobalSign Atlas settings are added, validation needs to be done to check whether the connection between AppViewX and GlobalSign is properly configured.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **GlobalSign** in the left side vendor list, and then select the **GlobalSign Atlas** tab.
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that is created.
4. CA communication will be validated and the Connection Status will be shown as either Success or Failure.



GoDaddy CA

Configuring GoDaddy

To configure the GoDaddy CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **GoDaddy** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. Example: Server, Client.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the GoDaddy CA APIs for Certificate Management.

CA Configuration - Field and Description Table

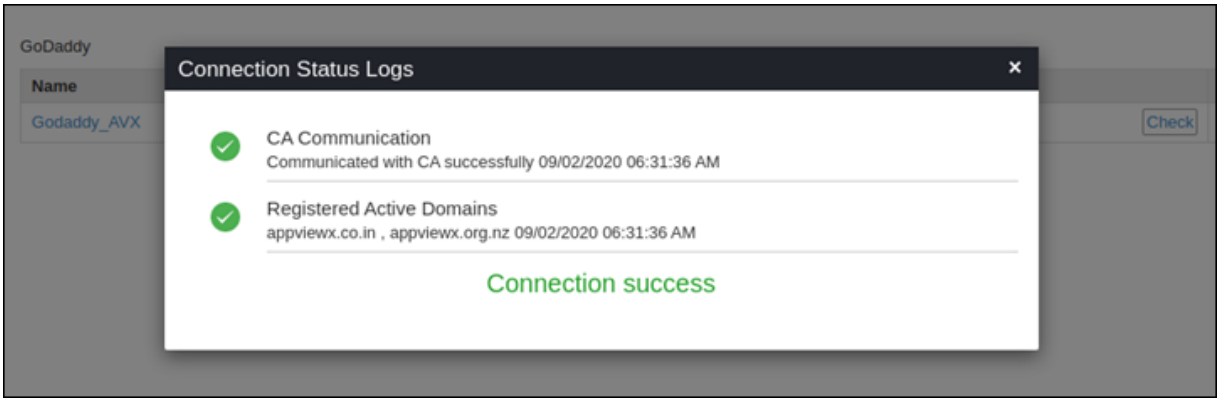
Name	Description
***Base URL**	This URL will contain the Base URL of the GoDaddy CA API instance. For example: https://api.godaddy.com
***Customer Number**	Each user will have a unique customer number which is used to obtain the certificates from the GoDaddy CA account.
***API Key**	API key generated in the GoDaddy portal which is used for GoDaddy API communications.
***API Secret**	API Secret generated in the GoDaddy portal which is used for GoDaddy API communications.
First Name	First name of the GoDaddy Account user's name as provided in the portal to be used for certificate creation purposes.
Last Name	Last name of the GoDaddy Account user's name as provided in the portal to be used for certificate creation purposes.
Email Address	Email Id of the GoDaddy Account user's name as provided in the portal to be used for certificate creation purposes. **Note**: Valid email address.
Phone Number	Phone number of the GoDaddy Account user as provided in the portal to be used for certificate creation purposes. **Note:** Phone numbers must contain a minimum of 7 and a maximum of 15 numeric values.
**: Mandatory fields*	

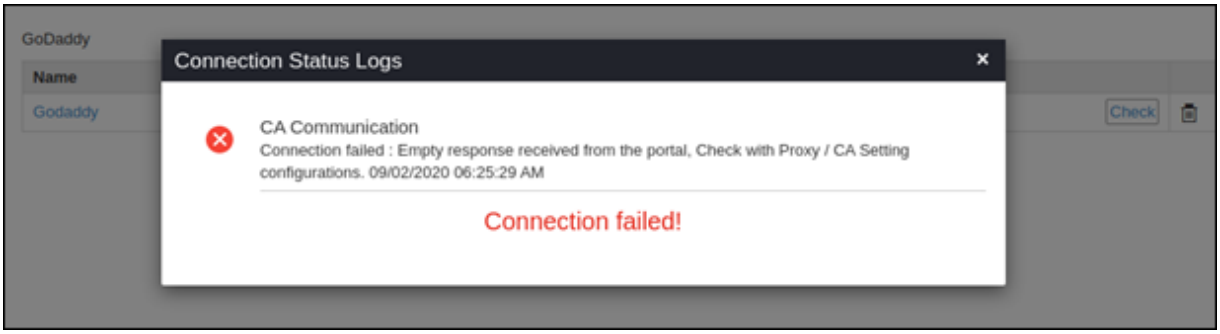
6. Click **Save**.

Validating GoDaddy

Once the GoDaddy settings are added validation needs to be done to check whether the connection between AppViewX and GoDaddy is properly configured. To validate GoDaddy CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **GoDaddy** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that has been created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.





View GoDaddy Product Units

Each GoDaddy account will have different types of SSL products and units, below said steps would allow users to know the availability of the products and their remaining units.

To view the GoDaddy product units:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **GoDaddy** in the left side vendor list
3. On the **GoDaddy** page, click **View** to fetch the product types and the units available for the GoDaddy account configured. Once clicked, users can view the available and used product units.

Google CA

Before you begin

Following are the prerequisites for configuring a Google CA account in AppViewX:

- A Google client certificate or Google client authentication JSON for a user having necessary access for enrolling the certificates and for other Certificate Lifecycle Management(CLM) operations.
- AppViewX servers should either have internet access or have a proxy configured in AppViewX general settings.
- The URL https://www.googleapis.com should be reachable from AppViewX.

Configuring Google

To configure the Google CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Google** in the left side vendor list.
4. Click the **+Add** icon on the top right of the page.
   The Google configuration page is displayed.
5. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting.

Name	Description
	Note: No special characters other than '.', '-','_' are allowed. The name should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. For example, Server and Client
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

6. Configure it either Certificate Upload or JSON Upload. These fields are necessary for invoking the Google CA APIs via Certificate Upload for Certificate Management. Select the Certificate Upload check box,

CA Configuration - Field and Description Table

Options	Description
***Certificate and Key**	Client authentication certificate for API communication.
***Email address**	Email address of the user
***Project Id**	ID of the project
**: Mandatory fields*	

7. Select the JSON Upload check box and configure a CA. Click the Upload button to upload the JSON file.
8. Click Validate and Fetch. The issuer names available for the CA account will be fetched along with the validity of the issuers from the Certificate Authority.
9. Click **Save**.

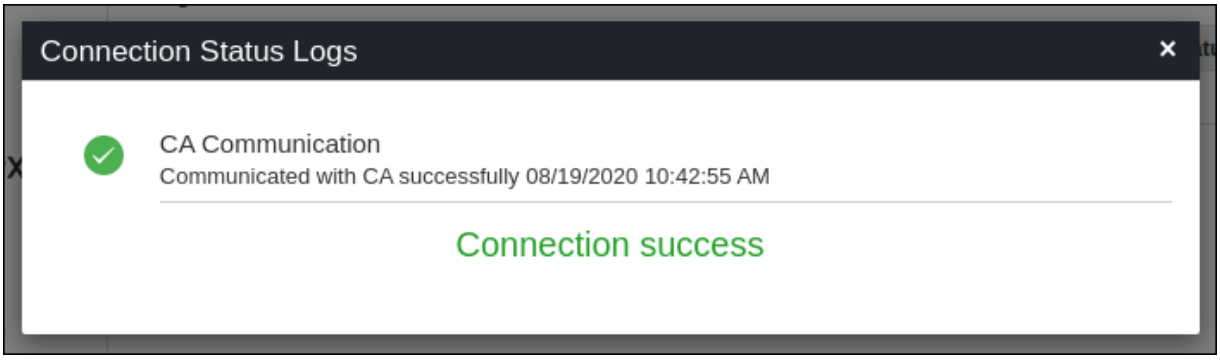
Validating Google

Once the Google settings are added validation needs to be done to check whether the connection between AppViewX and Google is properly configured. To validate the Google CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Google** in the left side vendor list

   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that is created.

   CA communication is validated and the Connection Status is shown as either Success or Failure.



Certificate Authority Service CA

Before you begin

Following are the prerequisites for configuring a Google CA account in AppViewX:

- A Google client certificate or Google client authentication JSON for a user having necessary access for enrolling the certificates and for other Certificate Lifecycle Management (CLM) operations.
- AppViewX servers should either have internet access or have a proxy configured in AppViewX general settings.
- The URL https://www.googleapis.com should be reachable from AppViewX.

Configuring Certificate Authority Service

To configure the Certificate Authority Service CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
   The Google configuration page is displayed.
3. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note:** No special characters other than '.', '-','_' are allowed. The name should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions are enabled. For example, Server and Client.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

4. Configure it either Certificate Upload or JSON Upload. These fields are necessary for invoking the Google CA APIs via Certificate Upload for Certificate Management. Select the Certificate Upload check box,
   Update the following details in the **CA Configuration** section as described in the table.

CA Configuration Section - Field and Description Table

Options	Description
***Certificate and Key**	Client authentication certificate for API communication.
***Email address**	Email address of the user
***Project Id**	ID of the project
**: Mandatory fields*	

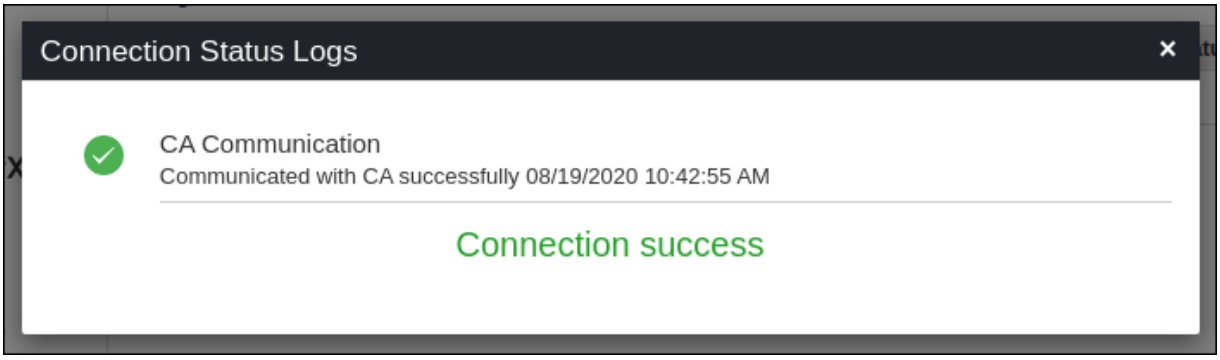
5. Select the JSON Upload check box and configure a CA. Click the Upload button to upload the JSON file.

6. Click Validate and Fetch. The issuer names available for the CA account will be fetched along with the validity of the issuers from the Certificate Authority.
7. Click **Save**.

Validating Certificate Authority Service

Once the Google settings are added validation needs to be done to check whether the connection between AppViewX and Google is properly configured. To validate the Google CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Google** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that is created.
4. CA communication will be validated and the Connection Status will be shown as either Success or Failure.



HashiCorp Vault CA

- Before You Begin
- Configuring HashiCorp Vault
- Validating HashiCorp Vault

Before You Begin

The prerequisites for configuring HashiCorp Vault CA are

- Login and password to access AppViewX.
- Base URL, Role ID, and Secret Key for the APP ROLE method and Base URL, Access Key, Secret Key, and Role Name for the AWS method in the CA Configurations.

Configuring HashiCorp Vault

Steps to Configure HashiCorp Vault CA

To configure HashiCorp Vault CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **HashiCorp Vault** in the left side vendor list.
4. Update the details in the General Information section as described in the table below:

General Information - Field and Description Table

Field Name	Description
***CA Account name**	A unique name to identify the CA setting. **NOTE**: No special characters other than '.', '-', and '_' are allowed. Names should not start with special characters
***Purpose/Usage**	The dropdown contains checkboxes for the certificate type for which the CLM actions will be enabled. The values are: • Server, • Client • Code Signing One or more values can be selected depending on the type of account users need to create.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.

Field Name	Description
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the details in the **CA Configuration** section as described in the tables below. These fields are necessary for invoking the HashiCorp CA Secret Engine for Certificate Management. The fields displayed in the CA Configuration section depend on the value selected in the **Method** field. The two auth method values are:
   - **APP ROLE** - The APP ROLE auth method allows machines or apps to authenticate with Vault-defined roles. An "AppRole" represents a set of Vault policies and login constraints that must be met to receive a token with those policies. An AppRole can be created for a particular machine, or even a particular user on that machine or a service spread across machines. The credentials required for successful login depend upon the constraints set on the AppRole associated with the credentials.
   - **AWS** - The AWS auth method provides an automated mechanism to retrieve a Vault token for IAM principals and AWS EC2 instances. Unlike most Vault auth methods, this method does not require manual first-deploying or provisioning of security-sensitive credentials (tokens, username/password, client certificates, etc), by operators under many circumstances.

CA Configuration for App Role - Field and Description Table

Field Name	Description
***Base URL**	The base of URL of the CA account. This is a user input value.
***Method**	**APP ROLE**
***Role ID**	User input value RoleID is an identifier that selects the AppRole against which the other credentials are evaluated. When authenticating against this auth method's login endpoint, the RoleID is a required argument at all times. By default, RoleIDs are unique UUIDs, which allow them to serve as secondary secrets to the other credential information.
***Secret Key**	User input value

Field Name	Description
	Secret Key (SecretID) is a credential that is required by default for any login and is intended to always be secret. They can be created against an AppRole either via generation of a 128-bit purely random UUID by the role itself or via specific, custom values. Similarly to tokens, Secret keys have properties like usage-limit, TTLs and expirations.
**: Mandatory fields*	

CA Configuration for AWS - Field and Description Table

Field Name	Description
***Base URL**	The base of URL of the CA account. This is a user input value.
***Method**	**AWS**
***Access Key**	User input value Access Keys are used to sign the requests that are sent. Access Key and Secret Key are used for programmatic (API) access to AWS services.
***Secret Key**	User input value Secret Key (SecretID) is a credential that is required by default for any login and is intended to always be secret. Similar to tokens, SecretIDs have properties like usage-limit, TTLs, and expirations.
***Role Name**	User input value The basic mechanism of operation (AWS authorization workflow) is per-role. Roles are registered in the method and associated with a specific authentication type that cannot be changed once the role has been created. Roles can also be associated with various optional

Field Name	Description
	restrictions, such as the set of allowed policies and max TTLs on the generated tokens.
**: Mandatory fields*	

The correct values entered in the fields establish a connection with the HashiCorp vault to be able to fetch the secret engine.

6. Click the **Fetch Secret Engine** button.
   A list of PKI secret engines is displayed. These will be presented to users in the policy. from where they can select the respective secret engines.
7. Click **Save**.

   The Account details are displayed in a grid at the bottom of the screen, with options to options to edit, check (connection status), and delete.

Editing an Account

1. Go to **Start** > **Certificate**, and under Advanced Configuration, click **Create a Certificate Authority**.
   The list of Accounts is displayed in the grid.
2. Click the account name from the ‘Name’ column in the grid.
   The General Information and CA Configuration sections are displayed with pre-populated values.
3. Change any of the editable fields and click the **Fetch Secret Engine** button.
4. Click the **Update** button.

Deleting an Account

1. Go to **Start** > **Certificate**, and under Advanced Configuration, click **Create a Certificate Authority**.
   The list of Accounts is displayed in the grid.
2. In the last column of the grid with the listed accounts, click the **delete** or bin icon.
   The Delete Confirmation pop-up is displayed.
3. Click **Yes**.

Validating HashiCorp Vault

To check the connection status of an account,

1. Go to **Start** > **Certificate**, and under Advanced Configuration, click **Create a Certificate Authority**.
   The list of Accounts are displayed in the grid.
2. In the Status column of the grid with the listed accounts, click the **Check** button

The Success or Failure value is displayed.

3. Update the account details accordingly to get a “success” status.

InCommon CA

Before you begin

Following are the prerequisites for configuring InCommon CA account in AppViewX:

- InCommon Certificate Manager credentials having necessary access for enrolling the certificates.
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure the **proxy. https://adminguide.appviewx.com/proxy-4**
- Username and Password as set up in the Certificate Manager tool.
- An OrgID as provided by InCommon Certificate Manager.
- The login URL and URI.

Configuring InCommon CA

To configure the InCommon CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **InCommon** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting.

Name	Description
	Note: No special characters other than '.', '-','_' are allowed. The name must not start with special characters.
***Purpose/ Usage**	Certificate Type for which CLM actions will be enabled. For example, Server, Client.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the InCommon CA APIs for Certificate Management.

CA Configuration - Field and Description Table

Name	Description
***Base URL**	This URL will contain just the hostname of the InCommon CA instance. For example, https://cert-manager.com/customer/<<customer_uri>>/ssl- here base URL is **https://cert-manager.com.** **Note:** No special characters other than '.', '-','_' are allowed. The name must not start with special characters.
***Login URL**	URI specific to the InCommon CA Customer Account. For example, https://cert-manager.com/customer/<<customer_uri>>/ssl- here URI is **customer_uri**.

Name	Description
***User Name**	User name for the account created with InCommon CA.
***Password**	Password for the account created with InCommon CA.
***Organization ID**	InCommon supports organization hierarchy. Id of the **Organization Unit/Department** in which Certificates need to be managed has to be specified here. **CLM** actions done using this CA account will be specific to this particular organization's id/department.

Note:

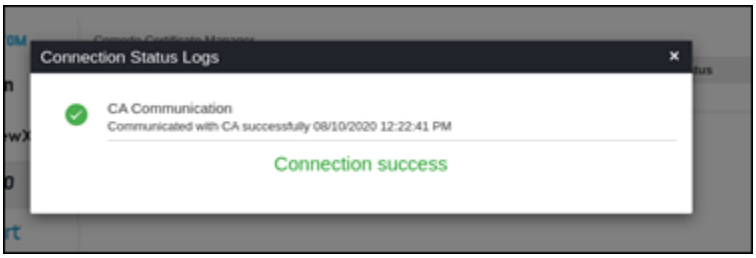
- **: Mandatory fields*
- If the certificates from multiple organization's units/departments need to be managed, then a separate CA has to be configured for each organization unit/department in the InCommon CA setting page.

6. Select **Fetch Certificate Types**.
   The Certificate types available for the CA account will be fetched from the Certificate Authority.
7. Click **Save**.

Validating InCommon CA

Once the InCommon settings are added validation needs to be done to check whether the connection between AppViewX and InCommon is properly configured. To validate the InCommon CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **InCommon** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that has been created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



Let’s Encrypt CA

Before you begin

Following are the prerequisites for configuring Let’s Encrypt CA account in AppViewX:

- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure proxy. https://adminguide.appviewx.com/proxy-4
- Any one of the following Let’s Encrypt certificate enrolment URL as per requirement :
  1. https://acme-staging-v02.api.letsencrypt.orgfor **staging**.
  2. https://acme-v02.api.letsencrypt.org for **production**.

Configuring Let’s Encrypt CA

To configure the Let's Encrypt CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Let's Encrypt** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***Name**	A unique name to identify the CA setting.

Name	Description
	Note: No special characters other than '.', '-','_' are allowed. The name must not start with special characters.
***Purpose/Usage**	The certificate types will be managed by these settings. For now, Let's Encrypt is having only one purpose **Server**.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the Let's Encrypt CA APIs for Certificate Management.

CA Configuration - Field and Description Table

Name	Description
***Base URL**	Let's Encrypt certificate enrolment URL either staging or production based on the requirement.
***Email ID(s)**	Enter email ID(s) in this field to receive notifications from Let's Encrypt. Multiple email ID must be separated by comma (,).
**: Mandatory fields*	

6. Click **Save**.

Validating Let's Encrypt

Once the Let's Encrypt settings are added validation needs to be done to check whether the connection between AppViewX and Let's Encrypt is properly configured. To validate the Let's Encrypt CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **InCommon** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that has been created. The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.

Microsoft Enterprise CA

Before you begin

Following are the prerequisites for configuring Microsoft Enterprise CA in AppViewX

AppViewX Windows Gateway installer should be installed in a windows machine, running and reachable from AppViewX vendor plugin(s) through the Communication Modes described below.

Communication Mode Table

Communication mode	Category	Windows gateway machine	Microsoft CA
NATIVE API	User account type	Service account	Service account .
	User permission		Read, Request certificates, Issue and Manage certificates permission at CA level for the service account or the service account group or authenticated users Enroll permission at Certificate template level for the service account or the service account group or authenticated users
	Services	RPC service	RPC service certutil.exe command availability

Communication Mode Table (continued)

Communication mode	Category	Windows gateway machine	Microsoft CA
	Ports		135 as the incoming port
POWERSHELL	User account type	Service account	Service account.
	User permission		Full control permission to C:\Windows \Temp Read, Request certificates, Issue and Manage certificates permission at CA level for the service account or the service account group or authenticated users
	Services	RPC Service, WinRM Service, WinRM Configuration, Powershell remoting,certutil.exe command availability	RPC Service, WinRM Service, WinRM Configuration, Powershell remoting,certutil.exe command availability.
	Ports		5985
WMI	User account type	Service account	Service account
	User permission		Full control permission to C:\Windows \Temp Read, Request certificates, Issue and Manage certificates permission at CA level for the service account or the service account group or authenticated users
	Services	WMI service certutil.exe command availability	WMI service certutil.exe command availability
	Ports	NA.	135, 445 or 139

Configuring Microsoft Enterprise CA

To configure the Microsoft enterprise CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Microsoft** in the left side vendor list, and then click the **Enterprise** tab.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. Example. Server, Client, Code Signing
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table.

CA Configuration - Field and Description Table

Name	Description
***Windows Gateway URL**	Enter the URL where the AppViewX agent is running.
***Windows Gateway Type**	The mode of communication types from Windows Gateway machine to CA machine. Available types are **NATIVE API, POWERSHELL, WMI.** Refer **Communication Mode**

Name	Description
Client Authentication Certificate	The client certificate used while installing Windows Gateway. Users can use the default client certificate (ClientCertificateGateway.pfx) or the custom certificate given by the Customer.
***Credential Type**	Type of credential to be used. Either **Manual Entry** or **Credential List.**
Username	User name of the credentials.
Password	Password for the username.
**: Mandatory fields*	

6. Click **Fetch CA Names** to retrieve CAs accessible from Windows Gateway installed machine.
   Upon successful completion of Fetch CA Names, all reachable CAs listed in **Select CA**.
7. Click on one specific CA and proceed.

Dynamic Fields for the Select CA Section

Name	Description
Select CA	All the reachable CAs are listed here.
***CA Machine Hostname**	Host name of the CA Machine will be auto-filled.
***CA Name**	Name of the CA chosen which will be auto-filled.
CA Manager Approval	Approves the pending enroll / Renew request submitted from AppViewX Certificate.
***Time Zone**	To perform scheduled and Optimized CA discovery, please provide time zone value.
**: Mandatory fields*	

Using Native API

Using Powershell and WMI

1. Configure the **Template Details**.
   Once CA is selected from the **Select CA** list, the **Template** details should have auto-filled.

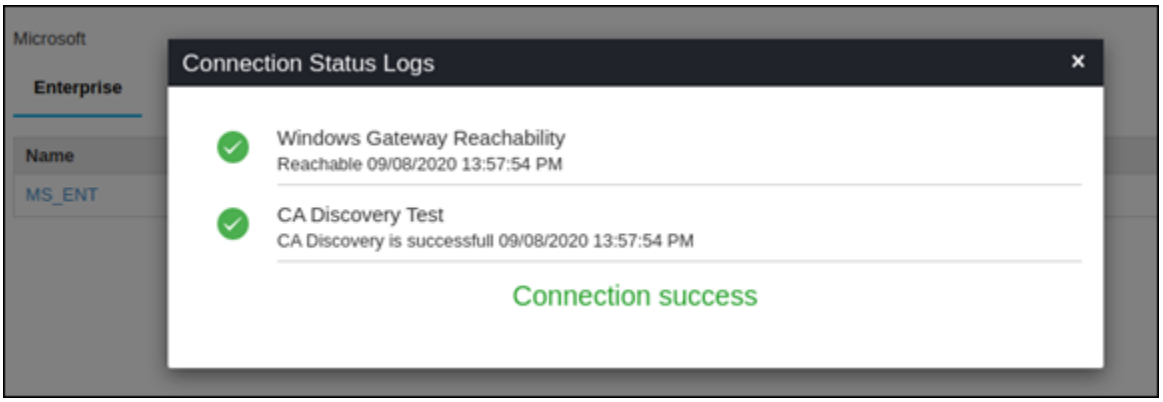
> **Note:** If the desired template is not listed, it might not be published in AD. Users can add it manually through MS Template name and OID fields.

2. In the Template Details section, select/enter the details.
3. Click **Save**.

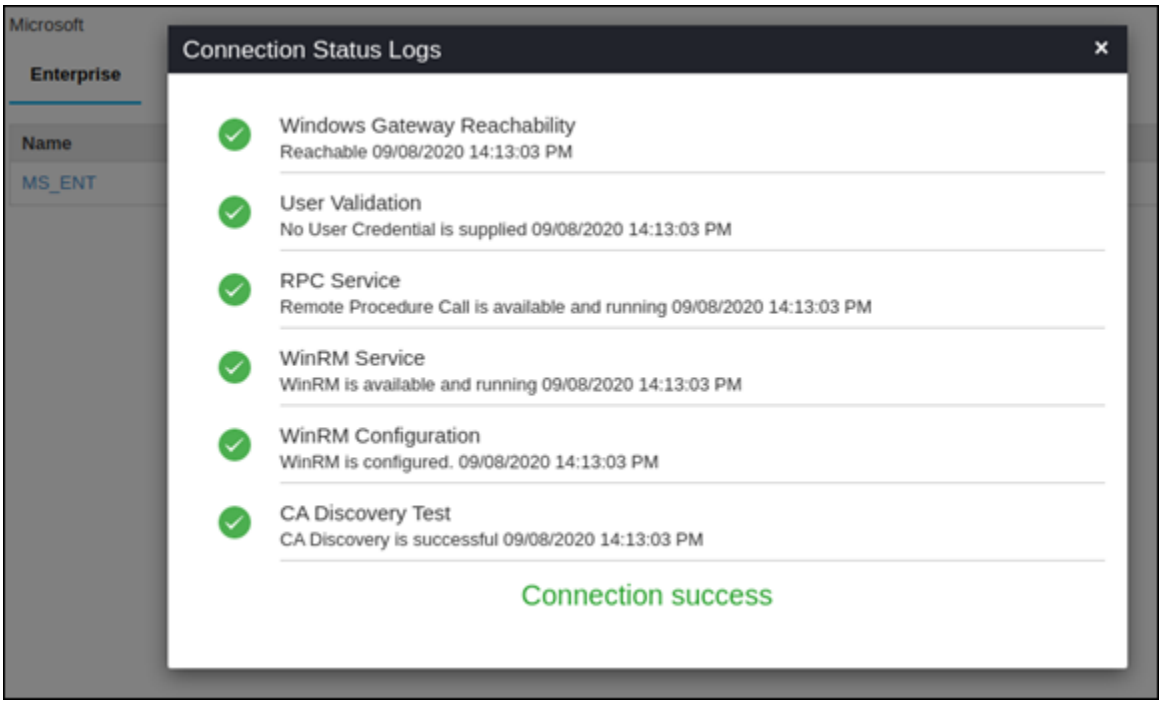
Validating Microsoft Enterprise

Once the Microsoft Enterprise settings are added validation needs to be done to check whether the connection between AppViewX and Microsoft Enterprise is properly configured. To validate the Microsoft enterprise CA,

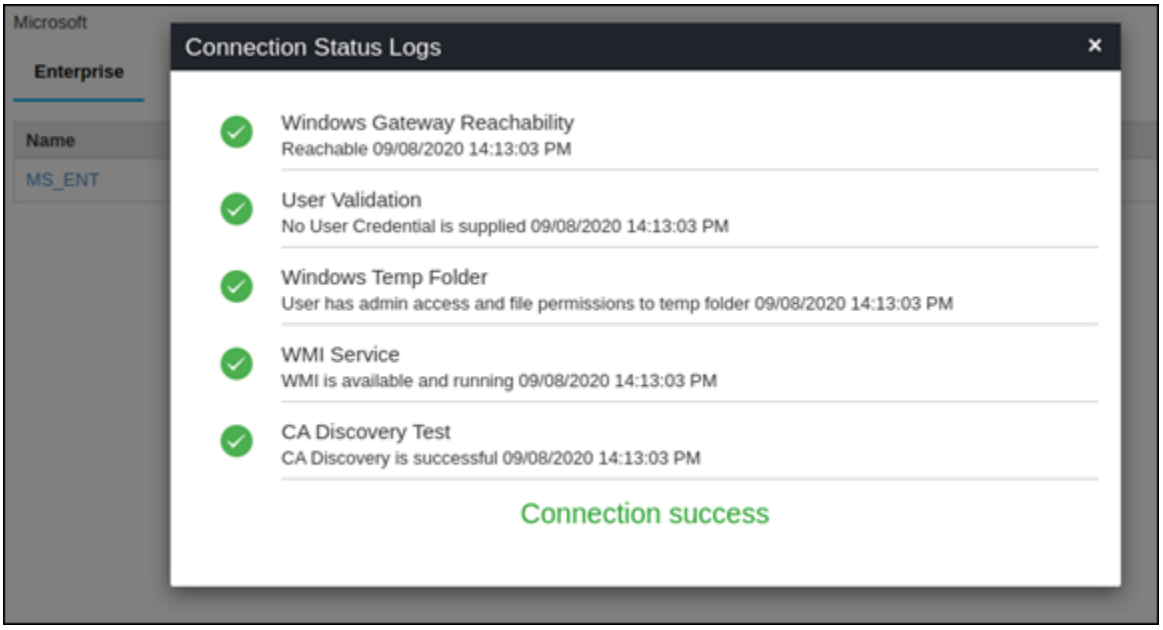
1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **InCommon** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that has been created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.
   Success Message.



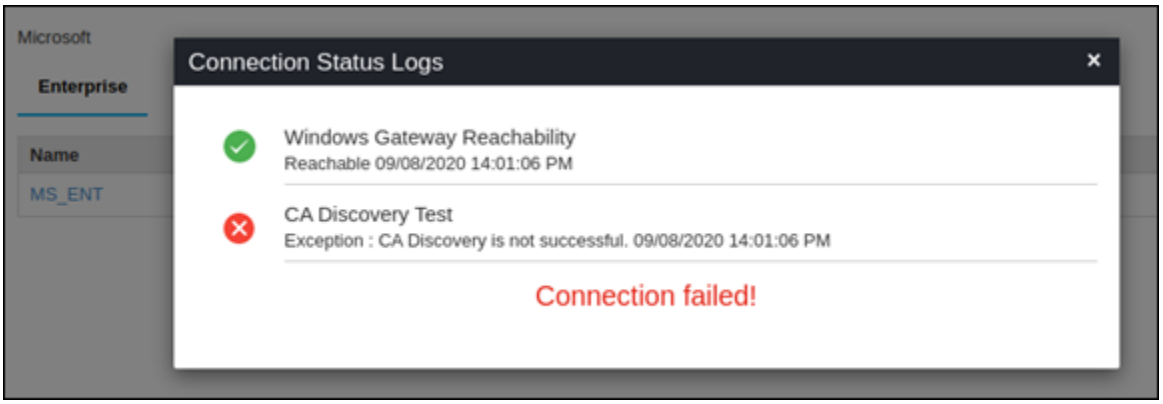
Success Scenario for Native API



Success Scenario for Powershell



Success scenario for WMI



Nexus

Prerequisites for Nexus

The following are the prerequisites for configuring a Nexus CA account in AppViewX:

- A Nexus Account with Administrator role Access.
- Before discovery or enrollment, the customer must upload the CA certificates manually.
- The AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure the proxy.

Configuring Nexus CA

To configure the Nexus CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Nexus** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table.

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. No special characters other than '.', '-','_' are allowed. The name should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. For example, server and clients
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table.

CA Configuration Section - Field and Description Table

Options	Description
***SSL URL**	Base URL of the SSL API
***User Name**	Provide a username of the GCC to communicate with the CA.
***Password**	Provide a password for the GCC to communicate with the CA.
**: Mandatory fields*	

6. Select **Fetch Procedures**.

> **Note:** The procedures available in the Nexus CA account will be fetched and listed for the specific user.

7. To map the fetched procedures, click on one or many and click the **Actions** dropdown:

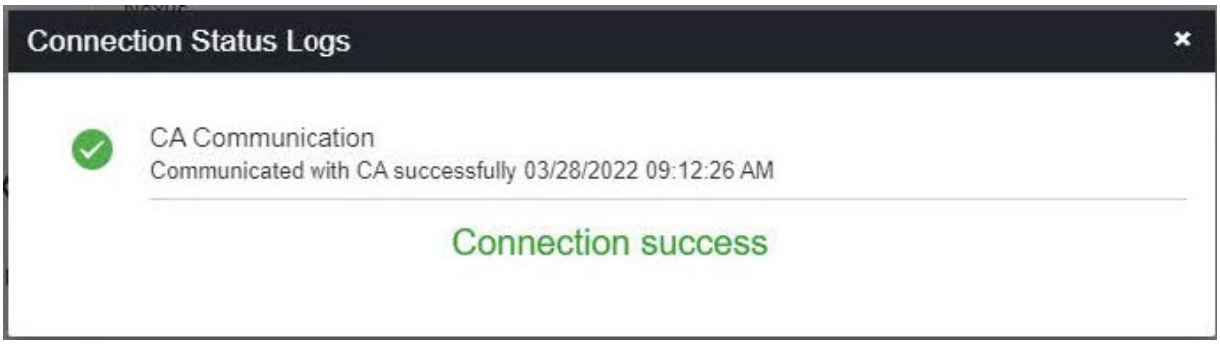
a. **CASE 1** - If the user selects Server only in Purpose and Usage, then the fetched procedure by default will be of server/client both. The Action dropdown will only have - **Map as Server**. and **MAP as Default**
b. **CASE 2** - If the user selects Client only in Purpose and Usage, then the fetched procedure by default will be of server/client both. The Action dropdown will only have - **Map as Client**. and **MAP as Default**
c. **CASE 3** - If the user selects Server and Client both in Purpose and Usage, then the fetched procedure by default will be of server/client both. The Action dropdown will have both the actions **Map as Client** , **Map as Server**, and **MAP as Default**

8. Click **Save**.

Validating Nexus

Once the Nexus settings are added, the validation must be done to check whether the connection between AppViewX and Nexus is configured properly.

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Nexus** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that is created.
4. The CA communication will be validated and the Connection Status will be shown as either Success or Failure.



OpenTrust CA

Configuring OpenTrust CA

To configure the OpenTrust CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **OpenTrust** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table:

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note**: No special characters other than '.', '-','_' are allowed. Names should not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. For example: Server and Client
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the OpenTrust CA APIs for Certificate Management.

CA Configuration - Field and Description Table

Name	Description
***Client Authentication**	Client authentication certificate for API communication. **Note**: Must be a valid <.*p12*> file.
***Base URL**	This URL will contain just the hostname of the OpenTrust CA instance. Eg - https://api.opentrust.net/enterprise/v2
**: Mandatory fields*	

6. Click **Fetch CA and Profile Names.**
   The attributes available for the CA account will be fetched from the Certificate Authority along with the CA and profile names.

   Note: The pop-up message is displayed as **CA and profiles fetched**.

7. Click **Save**.
   The created OpenTrust configuration settings will be added.

   Note: The pop-up message is displayed as **<CA_name> Settings Added**.

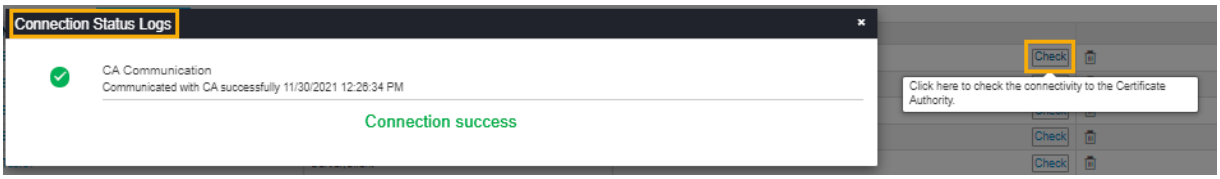
Validating OpenTrust Connection

Once the OpenTrust settings are added, validation needs to be done to check whether the connection between AppViewX and OpenTrust is properly configured.

To validate the OpenTrust connection,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **OpenTrust** in the left side vendor list.
3. Click **Check** to validate the CA setting that has been created.

   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.



Symantec CA

Before you begin

Following are the prerequisites for configuring a Symantec account in AppViewX:

- A Symantec client certificate for a user having the necessary access for enrolling the certificates and other Certificate Lifecycle Management(CLM) operations.
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure proxy. https://adminguide.appviewx.com/proxy-4
- Symantec users should be associated with the role **“w=VICE2 web services application”**.
- Required organization status should be “valid”.
- If the EV certificate type is enabled, then the EV status of the organization should be “Yes”.
- The required domain should be registered with the organization.
- The required certificate types should be enabled with the required values in the portal.
- Unit values should be available for the required certificate type.

Configuring Symantec CA

To configure the Symantec CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Symantec** in the left side vendor list.
4. Update the following details in the **General Information** section as described in the table.

General Information - Field and Description Table

Name	Description
***CA Account name**	A unique name to identify the CA setting. **Note:** No special characters other than '.', '-','_' are allowed. The name must not start with special characters.
***Purpose/Usage**	Certificate Type for which CLM actions will be enabled. For example, Server and Client.
Proxy Required	Enable this field if the CA communication needs to happen via Proxy. The proxy details configured in general settings will be used for communication.
Data Center (AppViewX's CA agent)	Select the data center through which the CA communication needs to happen.
**: Mandatory fields*	

5. Update the following details in the **CA Configuration** section as described in the table. These fields are necessary for invoking the Symantec CA APIs for Certificate Management.

CA Configuration - Field and Description Table

Name	Description
***Certificate and Key**	Client authentication certificate for API communication. **Note:** Must be a valid <.p12> or <.pfx> file.
***URL**	Symantec URL used for API communications. For example, **https://certmanager-webservices.websecurity.symantec.com/vswebservices/**

Name	Description
***Jurisdiction hash**	Jurisdiction hash of the Symantec account. Available in the top right corner of the Symantec portal.
***First name**	First name of the user.
***Last name**	Last name of the user.
**: Mandatory fields*	

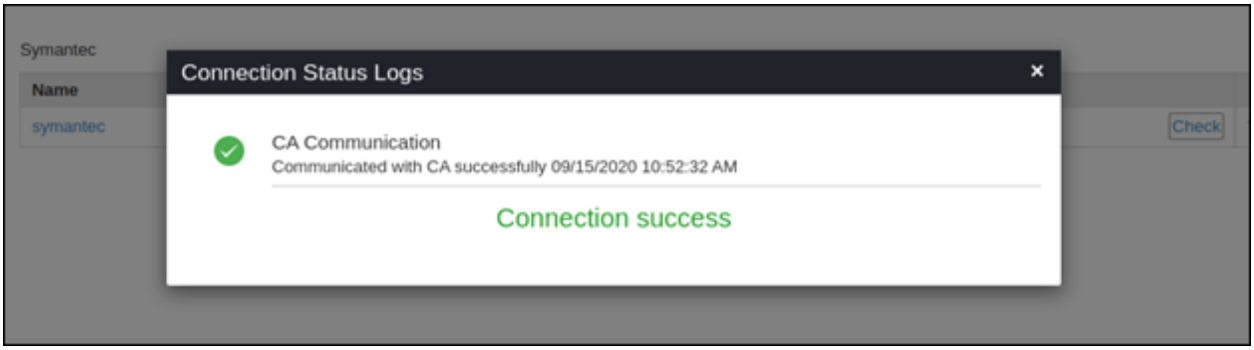
6. Click **Save**.

Validating Symantec

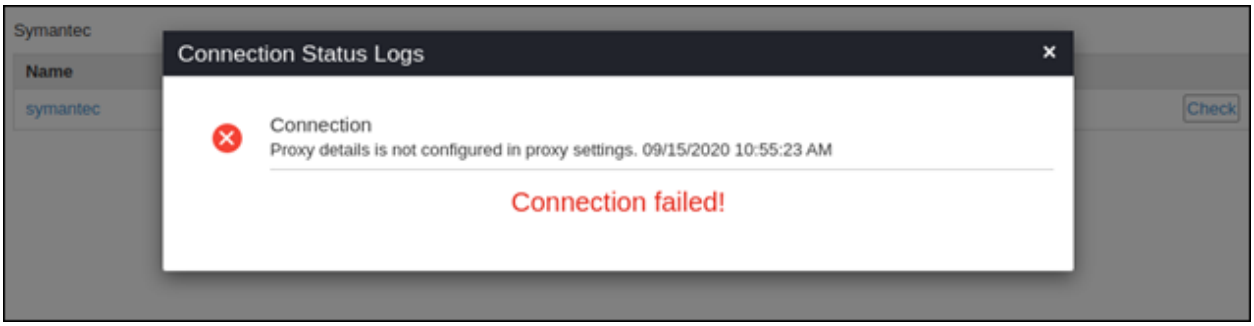
Once the Symantec settings are added validation needs to be done to check whether the connection between AppViewX and Symantec is properly configured. To validate the Symantec CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Symantec** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that has been created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.

Success Scenario



Failed Scenario



Trustwave CA

- Before you Begin
- Configuring Trustwave CA
- Validating Trustwave

Before you Begin

Following are the prerequisites for configuring Trustwave CA in AppViewX:

- Trustwave API URL. Ex: https://testapi.ssl.trustwave.com/3.0/
- Reachability from AppViewX southbound to Trustwave API URL via proxy or direct internet connection
- Valid credentials for communicating to Trustwave CA via API
- Reseller id
- Account details provided in Trustwave account such as Organization Name, Email address, Organization Address, City, State, Zip code, Country, Phone number
- AppViewX server should either have internet access or have a proxy configured in AppViewX general settings. Check Proxy Setup for the steps to configure the proxy. **https://adminguide.appviewx.com/proxy-4**

Configuring Trustwave CA

To configure the Trustwave CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**.
2. Click the **+Add** icon on the top right of the page.
3. Select the **Trustwave** in the left side vendor list.
4. Configure the **General Information** details as follows:

a. **CA Account name** - Provide an account name for the CA setting.
b. **Purpose/Usage** - Choose the certificate categories that will be managed by this setting. Possible certificate categories could be:
   i. Server
   ii. Code Signing
c. **Proxy Required** - Enable this field if the CA communication needs to happen via **Proxy**.
d. Choose the appropriate **Data Center**.

5. Configure the **CA Configuration** with information you want to configure:
   a. **API URL**: The Trustwave API URL to communicate. Ex: https://testapi.ssl.trustwave.com/3.0/
   b. **Username**: The username for API authentication.
   c. **Password**: The password for API authentication.
   d. **Reseller ID**: The Reseller Id for the account.
6. Configure the **Account Details** with information you want to configure:
   a. **Name**: The Organization name given in the Trustwave account.
   b. **Email Address**: The Administrator or organization email address given in the Trustwave account.
   c. **Address**: The Organization Address given in the Trustwave account.
   d. **City**: The city name given in the Trustwave account.
   e. **State**: The state name given in the Trustwave account.
   f. **Zip code**: The zip code given in the Trustwave account.
   g. **Country**: The country code given in the Trustwave account. Ex: US.
   h. **Phone number**: The phone number given in the Trustwave account.
7. Click **Save**.

Validating Trustwave

Once the Trustwave settings are added validation needs to be done to check whether the connection between AppViewX and Trustwave is properly configured. To validate the Trustwave CA,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Certificate Authority**
2. Select the **Trustwave** in the left side vendor list
   The newly created and older settings are displayed in the grid.
3. Click **Check** to validate the CA setting that is created.
   The CA communication will be validated and the **Connection Status** will be shown as either **Success** or **Failure**.

Groups

Before starting the configuration of Certificate Groups, it is important to be aware of the following key points:

- **Certificate Groups** are used to classify certificates based on different business units
- In certain organizations, **Certificate Groups** are used for assigning access permissions. Only privileged users, typically those inheriting permissions from the Resource > User Group, have the ability to view the corresponding **Certificate Groups**.
- Administrators should be assigned to a **Role** that grants them access to perform the following actions:
  - Create/ modify a group
  - Delete a group
  - Edit Default group

Creating a Group

Assign the user to a user group that (inherits from resource and role) enables them to access the certificate group.

1. Go to **menu** > **KUBE+** > **Groups & Policy** > **Groups**.

> **Note: KUBE+** is packaged with default certificate groups **Default** and **Certificate-Gateway** .

2. Click the **+ Create** button in the command bar to create a new group.
3. On the **Group : Create**, enter or select the required information.

Fields and Description for the Group Details Section

Name	Description
* **Select Group Hierarchy**	Select the parent group to which the new group should be associated
* **Group Name**	Enter a unique name for the new group. The name should not start with special characters and spaces. No special characters are allowed except('.', '-', '_') and name cannot end with space
Application ID	Provide organization ID (if any) to associate with the new group
Description	Provide the purpose of the new group
**: Mandatory field*	

Fields and Description for the Other Details Section

Name	Description
Contact Name	Provide contact person to whom changes should be intimated
Line of Business Name	Provide the name of the business unit
Email	Provide contact mail address
Environment Name	Provide environment name
Phone Number	Provide a phone number for contact
Inventory Number	Provide inventory number
Cost Center/ Hierarchy	Provide Cost Center code/ label
Push Certificate Automatically	By enabling the check box, the renewed/ reissued certificates in this group are automatically associated with their device
Renew Automatically	Turn **On** to automatically renew the certificate belongs to this group.
* **Associated Policy**	Displays the policy associated with this group.
**: Mandatory field*	

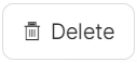
4. Click **Create** to create the group.
   Users can view the group only if it is associated with the **Resource** of their **User Group**. To associate the **Group** to a **Resource** click the **Create Group and Configure the Resources for User Access** button instead of **Create** button. This will create the group and go to **Resource.** Refer to Create a Resource section in the Platform User guide to configure user access.
5. The newly created **Group** is added to the Group inventory. Click the **Name** (Group name) to view the group details.
6. On the Certificate Count column, click on the count link of Server/Client/Device/Code Signing to view the certificates.

> **Note:** After the certificate discovery, you can view the count of certificates (Server, Client, Device, and Code Signing) associated with this group.

Modifying a Group

1. Go to **menu** > **KUBE+** > **Groups & Policy** > **Groups**.
2. On the Group Inventory page, click the **Name** (Group name).
3. Modify required fields in the group and click **Update**. Field descriptions are available in Create a Group section.

Deleting a Group

1. Go to **menu** > **KUBE+** > **Groups & Policy** > **Groups**.
2. On the group inventory page, select the check box against the group you want to delete.
3. Click 

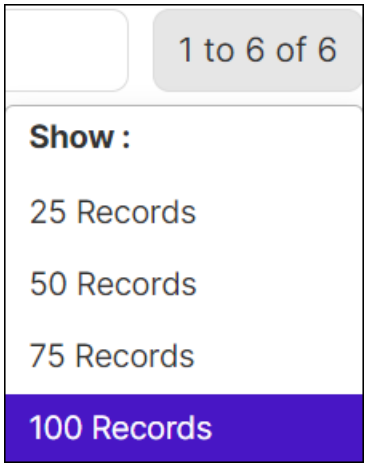
in the command bar.
4. On the confirmation pop-up , click **Yes** to proceed.

Cluster Policy

Cluster Policy enforces security prerequisites, standardizes certificate issuance, and ensures compliance, all while promoting secure certificate management practices throughout your clusters.

On the Cluster Policy page,

- refresh the list, click the (refresh) icon.
- go to the pages, click the (navigation) icon.
- hover the mouse over the number of row displayed on the page, the **Show** popup opens and choose the no. of rows to be displayed on the page.



The cluster policy inventory list includes the following information:

Column and Description table

Column Name	Description
Name	Unique policy name to be associated with one or more clusters. The special characters (-) and (_) are allowed. Maximum 255 characters are allowed.
Type	Type of cluster policy.
Created By	User ID of the policy creator.
No. of Clusters	Count of clusters associated with the policy.
No. of Namespace	Count of namespaces associated.
Last Updated At	Last updated Timestamp.
Edit	Allows to modify the clusters and namespaces associate with the policy.

Creating a Cluster Policy

Create Policy enables the Infosec teams / PKI administrators to create, define, and enforce policies for one more cluster managed in the inventory.

Note: The certificate automations (creation, renewal, etc.) initiated from a specific cluster must adhere to the policy parameters outlined in this policy inventory. Any cluster that is not a part of or does not align with the Cluster Policy will be denied certificate automations.

Why is Cluster Policy Essential?

Cluster Policy is your toolbox of rules and guidelines that you set up to manage the safe issuance of SSL/TLS certificates within your Kubernetes cluster. AppViewX offers various ways to ensure that these policies are followed when certificates are issued.

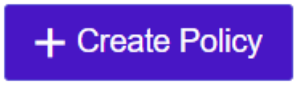
- **CA Setting** - When your application management teams work within a specific namespace, they often need access to a private CA to request certificates for their unique domains. The CA Setting policy type takes care of configuring this private CA and manages how certificates are provided within this specific namespace.
- **CA Setting Cluster** - If your application teams are deploying across the entire cluster, no matter where their apps land, the CA Setting Cluster policy type steps in. It handles the configuration of the CA and ensures certificates are issued seamlessly, maintaining security and consistency throughout the cluster.

Prerequisites:

- Ensure CA integration is completed.
- Ensure you configured organization PKI standards as CA Policy.
- Ensure the Group is created.

To create a cluster policy:

1. Go to **menu** > **KUBE+** > **GROUPS & POLICIES** > **Cluster Policy**
   On the **Cluster Policy** page, the created policies are displayed, if any.

2. Click 

.
3. Enter/select the policy information.

Policy Information - Field and Description Table

Field	Description
Policy Name*	Enter a unique policy name to be associated with one or more clusters.
Type*	Select a type from dropdown list. The options are: • **CA Settings Cluster** - cluster wide global policy. • **CA Setting** - Policy to be applied for a specific namespace or a project within a cluster.
Certificate Group*	Select a certificate group from the dropdown list.
Associate CA Policy*	The CA policy can be associated after creating the cluster policy. For associating the CA policy, see Associating a CA Policy.
Certificate Authority	Select a Certificate Authority from the dropdown list.
CA Settings	Select a CA Settings from the dropdown list.
**: Mandatory fields*	

4. Click **Add**.

Related Information

- Associating a CA Policy

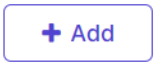
Associating a CA Policy

After creating the cluster policy, modify it to associate the cluster and/or namesapces along with certificate group, certificate authority, and CA settings.

Prerequisites:

- Ensure CA integration is completed.
- Ensure you configured organization PKI standards as CA Policy.
- Ensure the Group is created.

To modify a cluster policy:

1. Go to **menu** > **KUBE+** > **GROUPS & POLICIES** > **Cluster Policy**
   On the **Cluster Policy** page, the created policies are displayed, if any.
2. Click edit icon against the cluster policy.
3. On the **Policy Details Update Form**, select the policy details such as certificate group, certificate authority, and CA settings.
4. Under **Policy Association Details** section, click 

.
5. On the **Add policy association details** blade, add the following:
   - If the policy type is **CA Setting Cluster**, you can only select a cluster.
   - If the policy type is CA Setting, you can select both a cluster and its associated namespaces.
   - Click **Add**.
   - Close the **Add policy association details** blade.
6. Click **Update**.

Related Information

- Deleting a Cluster

Deleting a Cluster Policy

You can delete cluster policies from the inventory, if the user chooses to restrict and remove CLM operations on the designated cluster policy.

To delete a cluster policy:

1. Go to **menu** > **KUBE+** > **GROUPS & POLICIES** > **Cluster Policy**.
2. Select the designated cluster policy, regardless of whether it is in a Managed or Unmanaged state.
3. Click **Delete** on the menu bar.

Note:

- You can delete a cluster policy created from the inventory if the Policy is not applied or consumed by any of the Issuer CAs created in the Issuer CA inventory.

Automate Certificate Lifecycle Management

Steps for Automating Certificate Lifecycle Management

Request and provision certificates to your Kubernetes secrets or local volumes within a pod or a container and use them for securing your Kubernetes ingress or gateways. The provisioned certificates can also be automatically renewed before expiry.

The following outlines the step-by-step process to fully automate certificate lifecycle management within your clusters, ensuring compliance and promoting crypto-agility through simplified PKI policies.

1. Configure **Issuer CA** - Configure Certificate Authority Settings for your cluster and, if needed, fine-tune them to specific namespaces within the cluster to generate certificate signing requests
2. **Enroll Certificates** - The certificate request process involves obtaining certificates signed by the specified Certificate Authority (CA), which can then be deployed in Kubernetes secrets or pods.
3. **Download Certificates** - Process for retrieving certificates from the centralized inventory and deploying them to Kubernetes secrets or pods.

Issuer CA

In KUBE+, Issue CA refers to CA Settings, a Kubernetes resource that represents the configuration of certificate authorities (CAs) responsible for generating signed certificates through certificate signing requests.

To go to issuer CA inventory page, go to **menu** > **KUBE+** > **CLUSTER PKI** > **Issuer CA**.

On the Issuer CA Inventory page, the CA settings can be defined for your Kubernetes cluster, mapped with the Cluster, and pushed the CA settings to the cluster.

The issuer CA inventory list includes the following information:

Issuer CA Inventory - Column Description Table

Column Name	Description
Name	Name of the CA setting.
Cluster Policy	Cluster Policy associated to the cluster
CA Setting Type	The CA seeting type that has been select for this policy.
Created By	User ID of who created this CA setting.
Map to Cluster	Allows to map the CA setting to the cluster.
Push to Cluster	Allows to push, delete, or revoke the CA settings to the cluster.

- Creating a CA Settings
- Deleting an Issuer CA

Creating a CA Settings

Create CA Setting enables the DevOps / CloudOps teams to create and configure CA Settings for their cluster based on the PKI policy defined for their clusters.

Prerequisites:

- Ensure CA integration is completed.
- Ensure you configured organization PKI standards as CA Policy.
- Ensure the Group is created.
- Ensure Cluster Policy is created.

To create a CA Settings:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Issuer CA**.
   On the **Issuer CA** page, the created Issuer CAs are displayed, if any.

2. Click 

.
3. On the **Create CA** page, enter/select the policy information fields in the General Information section.

General Information - Field and Description Table

Field	Description
CA Settings Name*	Enter a CA settings name.
Select Policy*	Select a required Cluster Policy associated with the cluster.
Group Name*	Select a Certificate Group and the respective CA Issuer associated with the Group.
Type*	Choose a specific CA Account and the specific profiles or attributes required to send the certificate signing requests to the Certificate Authority.
**: Mandatory field*	

4. On the **Create CA** page, enter/select the policy information fields in the **Certificate Authority Setting** section.

Certificate Authority Setting - Field and Description Table

Field	Description
Certificate Authority	Select a preferred certificate authority from the dropdown list.
CA Account	Select a preferred CA Account from the dropdown list.
Connector Name	Select preferred clusters from the dropdown list.
Category*	Select a certificate group from the dropdown list.
**: Mandatory field*	

5. Click **Add**.
   CA Setting is created.

Mapping a CA Settings with a Cluster

Map the CA settings with a cluster.

Prerequisites:

- Ensure CA integration is completed.
- Ensure you configured organization PKI standards as CA Policy.
- Ensure the Group is created.
- Ensure Cluster Policy is created.

To create a CA Settings:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Issuer CA**.
   On the **Issuer CA** page, the created Issuer CAs are displayed, if any.
2. On the **Map to Cluster** column, click the map icon against a CA setting.
3. On the **Map To Cluster** pop-up window, select a cluster.
4. Click **Map**.
   All the mapped Issuer CAs are displayed in the **Push To Cluster** inventory. You can view the CA Setting YAML configuration by clicking the desired Issuer CA name column on the **Push To Cluster** inventory.

Related information
Pushing, Deleting, or Revoking a CA Settings with a Cluster

Push and Revoke Issuer CA

KUBE+ enables a self-service feature of managing KUBE+ related kubernetes resources in the clusters directly from the KUBE+ control plane.

Cluster administrators or DevOps team can Push the Issuer CA/Certificate Authority Setting directly from the inventory into the respective cluster, this enables to reduce the efforts of manually downloading the Issuer CA/Certificate Authority Setting as YAML, copying it to the cluster and deploying it.

Similarly the cluster administrators or DevOps team can also Revoke (Delete) the Issuer CA/Certificate Authority Settings deployed in the specific cluster instead of executing the delete commands manually in the cluster.

CA Setting Push To Cluster Inventory - Column Description Table

Column Name	Description
Name	Unique CA Settings name. Alpha numerals and special characters (-) and (_) are allowed.
Created By	User ID of the policy creator.
Cluster Name	Name of the cluster.
Namespace	Namespace of the cluster.
Cluster Policy	Name of the policy that has been associated for the cluster.
Last Updated	Last updated Timestamp.
State	State of the CA.
Status	Status of the CA.

• Pushing an Issuer CA
• Revoking an Issuer CA

Pushing an Issuer CA

To push the issuer CA created,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Issuer CA**.
   On the **Issuer CA** page, the created Issuer CAs are displayed, if any.
2. On the **Push to Cluster** column, click the push to cluster icon against a CA setting.
3. On the **Issuer CA - Push To Cluster** page, select a row(s).
4. Click the **Push** button.

Revoking an Issuer CA

To revoke the issuer CA created,

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Issuer CA**.
   On the **Issuer CA** page, the created Issuer CAs are displayed, if any.
2. On the **Push to Cluster** column, click the push to cluster icon against a CA setting.

3. On the **Issuer CA - Push To Cluster** page, select a row(s).
4. Click the **Revoke** button.

Deleting an Issuer CA

You can delete issuer CA from the inventory, if the user chooses to restrict and remove CLM operations on the designated issuer CA.

To delete an issuer CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Issuer CA**.
   On the **Issuer CA** page, the created Issuer CAs are displayed, if any.
2. Select a desired issuer CA from the inventory.
3. Click **Delete** on the menu bar.

> **Note:** You can not delete an issuer CA, if one or more issuer CAs mapped to a cluster are in deployed state.

• Deleting a Cluster Mapped with Issuer CA

Related information
Creating a CA Settings

Deleting a Cluster Mapped with Issuer CA

You can delete issuer CA from the inventory, if the user chooses to restrict and remove CLM operations on the designated issuer CA.

To delete an issuer CA:

1. Go to **menu** > **KUBE+** > **CLUSTER PKI** > **Issuer CA**.
   On the **Issuer CA** page, the created Issuer CAs are displayed, if any.
2. On the **Push to Cluster** column, click the push to cluster icon against a CA setting.
3. On the **Issuer CA - Push To Cluster** page, select a row(s).
4. Click **Delete** on the menu bar.

> **Note:**

- Issuer CA or Certificate Authority Setting configured for the cluster or a specific namespace can be deleted from the Issuer CA inventory. Deleting the Issuer CA setting or YAML configuration from the inventory will not delete the Setting deployed as a YAML configuration in the cluster.
- Issuer CA can be removed from Issuer CA inventory only if the associated certificates issued by the specific Issuer CA are deleted from the Secure Apps Inventory.
- Along with the above, users are recommended to delete the respective kubernetes Issuer CA resources created or deployed in the cluster manually, Issuer CA deployed in the cluster can be of type CASetting (or) CASettingCluster. The commands are:
  - **For CASetting:** kubectl delete CASetting <casetting-name> -n <namespace-name>.
  - **For CASettingCluster:** kubectl delete CASettingCluster <casettingcluster-name>.

Related information
Creating a CA Settings

Secure Apps Inventory

In KUBE+, Enroll Certificates and Download Certificates refer to the process of creating a Kubernetes resource known as "Cert" (Enroll new certs) and “CertLoad” (Download existing certs) which represents an SSL/TLS certificate deployed in Secrets and Pods respectively. The Cert resource is generated by the cert-orchestrator and includes an associated certificate signing request and CertLoad resource is generated by the cert-orchestrator and consumed by the appviewx-csi-provider which also includes an associated certificate signing request. This request is then sent to the Certificate Authority (CA) for signing through the KUBE+ platform.

To go to Enroll Certificate inventory, go to **menu** > **KUBE+** > **Cluster Security** > **Secure Apps**.

The Enroll Certificate inventory list includes the following information:

Enroll Certificate Inventory - Column Description Table

Column Name	Description
Certificate Name	Name of the certificate.
Common Name	The common name of the certificate. Click on the desired common name to be redirected to the certificate inventory page, which displays the certificates associated with the common name.
Cluster Name	Name of the cluster.

Enroll Certificate Inventory - Column Description Table (continued)

Column Name	Description
CA Settings Type	Type of the CA settings.
Enroll To	The endpoint to which the certificate is deployed. The options are: • Secret • Pod
Auto Renew	The status of auto-renewal for the enrolled certificates. The options are: • True • False
Created By	User ID who enrolled the certificate.
Created Source	The source of the certificate enrollment request.
Updated Time	Last updated Timestamp.

- Enrolling a Certificate
- Download a Certificate
- Deleting a Certificate
- Push and Revoke CERT

Enrolling a Certificate

Enroll enables the DevOps teams / application owners to request a certificate for their application deployed in the desired Kubernetes cluster. The certificate which is enrolled can be deployed directly to the Kubernetes secrets or even the local volumes of the Kubernetes pods (or) containers.

Prerequisites:

- CA Integration done.
- CA Policy created.
- Certificate Groups created.

- Cluster Policy created.
- Issuer CA configured in KUBE+ and deployed in your cluster.

To enroll a certificate:

1. Go to **menu** > **KUBE+** > **Cluster Security** > **Secure Apps**.
2. Click + Enroll.
3. On the **Enroll Certificate** page, enter/select the field information in the General Information section for Cert resource to be created on Kubernetes cluster.

General Information Section - Field and Description Table

Field	Description
Enroll Cert To	Select the endpoint where the cert is to be deployed. The options are: • **Secret**: KUBE+ enrolls certificate and stores signed certificate and key in k8s secret • **POD** : KUBE+ has CSI provider which provisions certificate in the pods local volume
Format	The certificate file format that should be downloaded to the pod. The supported formats include PEM, PFX, P12, and JKS.
Encoding	The encoding type of the file content. Supported types include utf-8, hex, and base64.
File Name	The name of the certificate file to be created in the pod.
Password	If the certificate download is password-protected, provide the password.
Alias Name	The alias name in the keystore for the certificate file format, when it is in JKS format.
Alias Password	The password for the alias.
Is CA Required	Download the trust store for the enrolled certificate. Set to "False" will result in the download only leaf certificates.

Field	Description
Cluster	Select a cluster where the certificate to be deployed from the dropdown list.
CA Setting Name	Select a certificate authority to be used for signing the CSR from dropdown list.
Certificate Authority	The Certificate Authority used for certificate enrollment as configured in the Cluster Policy.
Certificate Category	The type of certificate. The options include 'Client' and 'Server'.
Certificate Name	Enter a Certificate Name for certificate storage within the K8s cluster.
Namespace	The name of the namespace within the Kubernetes cluster where the secret is to be created.
Secret Name	Enter a Secret Name for certificate storage within the K8s cluster.
Enable Auto Renewal	Select a auto renewal option. The options are: • False (default) - Certificates will not be automatically renewed prior to their expiration. • True - Certificates can be automatically renewed before they expire.
Renewal Policy	If Auto renewal set to “True”, users have the option to renew certificates either by 'Regenerating a new key' or 'Renewing with the existing key'.
Renew Before ‘X’ days	Set the auto-renewal period prior to the certificate's expiration.
CSR Validity	The validity for the CSR in the cluster, if not approved. The default is 24 hours.
Overwrite valid certificates	Replace existing valid certificates with the newly enrolled certificate.

4. Enter/select field information in the CSR Parameter section.

CSR Parameter Section - Field and Description Table

Field	Description
CSR Generation Endpoint	The default CSR generation endpoint option is K8s Secret. In the scenario where private keys need to be generated in AppViewX, users can choose the CSR endpoint as 'AppViewX'.
Common Name	Enter the common name of the cert.
Subject Alternative Name	Select a Subject Alternative Name from the dropdown list. The options are: • DNS - DNS of the cert • IP Address - IP Address of the cer
DNS/IP Address	Enter the DNS/IP address of the cert.
Organization	Enter the organization of the cert.
Organization Unit	Enter the organization unit of the cert.
Locality	Enter the locality of the cert.
Street	Enter the street of the cert.
State	Enter the state of the cert.
Province	Enter the province of the cert.
Country	Enter the country of the cert.
Postal Code	Enter the postal code of the cert.
Email Address	Enter the email address of the cert.

5. Enter/select the field information in the **Private Key Parameters** section.

Private Key Parameters Section - Field and Description Table

Field	Description
Key Type	Select a key type of the cert from the dropdown list. The options are:

Field	Description
	• RSA • EC
Bit Length	Select the bit length from the dropdown list. • CSR param bit lengths for RSA are 2048/4096/3072. • CSR param bit lengths for EC are 256/384/521.

6. Click **Generate Cert YAML** to get the certificate for enrollment in the **Certificate YAML** field.

Note:

- To download the enrolled CERT configuration as a YAML file, click 
- To see the commands in the full screen view, click the  .
- To copy the command, click  .

7. Click **Add** to add the certificate to the Enroll Certificate inventory list.

Download a Certificate

Download enables the DevOps teams and application owners to download a retrieve certificate from certificate inventory for their application deployed in the desired Kubernetes cluster. The certificate which is downloaded can be deployed directly to the Kubernetes secrets or even the local volumes of the Kubernetes pods or containers.

Prerequisites

- A valid certificate and key available in the certificate inventory of KUBE+.

To download a certificate:

1. Go to **menu** > **KUBE+** > **Cluster Security** > **Secure Apps**.
2. Click **Download**.

3. On the **Download Certificate** page, enter/select the field information in the General Information section for CertLoad resource to be created on Kubernetes cluster.

Download Certificate - Field and Description Table

Field	Description
Certificate Type	The type of certificate to be downloaded from the inventory. The type can be either 'Server' or 'Client'.
Common Name	Common name of the certificate.
Look for certificates in Inventory	The 'Search' button allows you to retrieve certificates based on the provided common name.
Select Certificate	Select the certificate from the search results.
Serial Number	Select the associated serial number of the certificate
Certificate Name	Enter a Certificate Name for certificate storage within the K8s cluster.
Download To	Select the endpoint where the cert is to be deployed. The options are: Secret: KUBE+ enrolls the certificate and stores signed certificate and key in k8s secret. POD: KUBE+ has a CSI provider, which provisions certificate in the pod's local volume.
Namespace	The name of the namespace within the Kubernetes cluster where the secret will be created.
Secret Name	Enter a Secret Name for storing the certificate within the K8s cluster.
Format	The certificate file format that should be downloaded to the pod. The supported formats include PEM, PFX, P12, and JKS.

Field	Description
Encoding	The encoding type of the file content. Supported types include utf-8, hex, and base64.
Password	If the certificate download is password-protected, provide the password.
Alias Name	The alias name in the keystore for the certificate file format, when it is in JKS format.
Alias Password	The password for the alias.
Is CA Required	Download the trust store for the enrolled certificate. Set to "False" will result in the download only leaf certificates.
File Name	The name of the certificate file to be created in the pod.

4. Click **Generate YAML** to get the certificate to be downloaded in the **Download YAML** field.
5. Copy and deploy the YAML in the cluster to retrieve the certificate from the Inventory. After the deployment is finished, click **Cancel**.
6. Alternatively, the YAML configuration can also be downloaded as a YAML file.

Deleting a Certificate

You can delete certificates from the inventory, if the user chooses to restrict and remove CLM operations on the designated certificate.

To delete a certificate:

1. Go to **menu** > **KUBE+** > **CLUSTER SECURITY** > **Secure Apps**.
2. Select the designated certificate, regardless of whether it is in a Managed or Unmanaged state.
3. Click **Delete** on the menu bar.

Note:

- You can delete the YAML configurations generated for enrolling/downloading a certificate to the Kubernetes secret or Pod from the Secure Apps Inventory.
- Deleting the Certificate Enroll/Download configuration from the inventory does not automatically delete the deployed certificate on a secret or pod. Users are recommended to manually delete the certificate from the respective secret or pod.
- In addition to the above, users are encouraged to delete the corresponding Kubernetes resources created, such as "Cert" for Eroll and "CertLoad" for Download certificates. This step ensures a thorough cleanup of certificates deployed in the cluster. The commands are:
  - For certificates enrolled via KUBE+: `kubectl delete Cert <cert-name> -n <namespace-name>`.

    For certificates downloaded via KUBE+: `kubectl delete CertLoad <cert-name> -n <namespace-name>`.

Push and Revoke CERT

KUBE+ enables a self-service feature of managing KUBE+ related kubernetes resources in the clusters directly from the KUBE+ control plane.

Cluster administrators or DevOps team can Push the **Cert** and **CertLoad** resources used for generating certificate signing requests directly from the inventory into the respective cluster, this enables to reduce the efforts of manually downloading the Resources as YAML, copying it to the cluster and deploying it.

Similarly, the cluster administrators or DevOps team can also Revoke (Delete) the **Cert** and **CertLoad** Resources deployed in the specific cluster instead of executing the delete commands manually in the cluster.

- Pushing a Cert
- Revoking a Cert

Pushing a Cert

To push the Cert or CertLoad Resources created,

1. Go to **menu** > **KUBE+** > **CLUSTER SECURITY** > **Secure Apps**.
2. Select a Cert or CertLoad from the inventory that is in a deployed state.
3. Click **PUSH**.

Revoking a Cert

To revoke the Cert or CertLoad Resources created,

1. Go to **menu** > **KUBE+** > **CLUSTER SECURITY** > **Secure Apps**.
2. Select a Cert or CertLoad from the inventory that is in a deployed state.
3. Click **Revoke**.

Secure Service Mesh with mTLS authentication

KUBE+ enables the PKI and InfoSec team to move towards Zero Trust Container Security by enabling scalable mTLS certificate issuance for authentication between microservices in Service Mesh Integrations.

Why Zero Trust is Critical?

Zero Trust helps organizations achieve compliance with industry and government regulations such as HIPAA, GDPR, and PCI-DSS. It ensures that access to sensitive data is strictly controlled and monitored, and that security policies are consistently enforced across the organization.

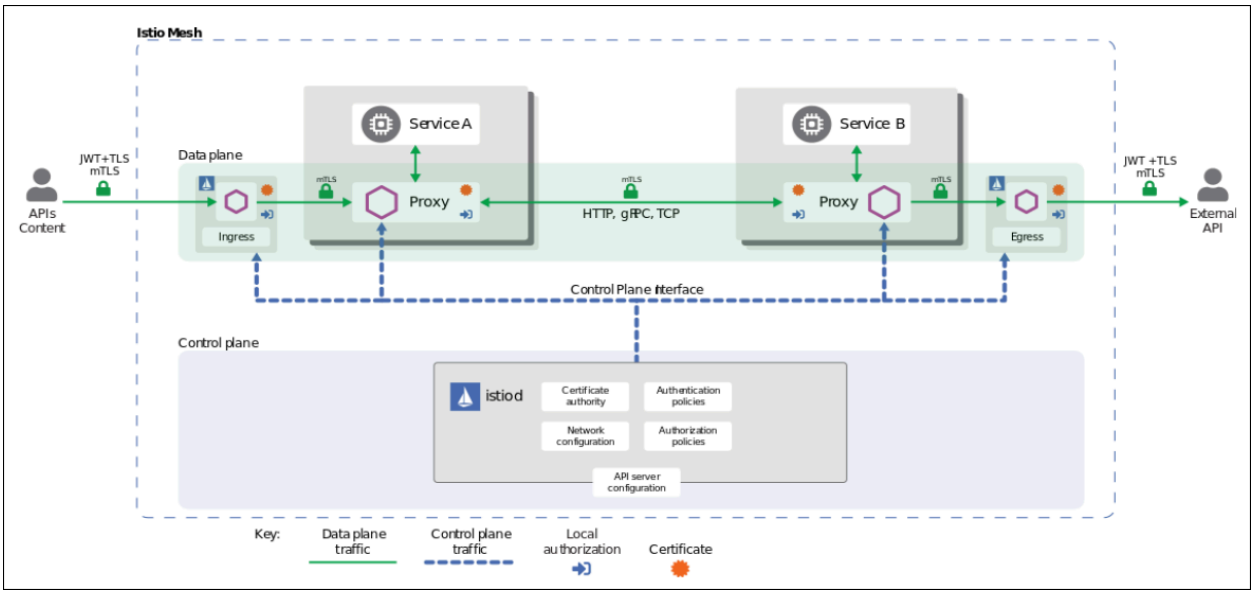
AppViewX Integration with Istio Service Mesh

Istio is a popular service mesh solution that provides traffic management, security, and observability for microservices-based applications. Integrating Zero Trust with Istio provides an additional layer of security for microservices-based applications. By enforcing access controls, Istio ensures that only authorized traffic is allowed between microservices.

mTLS with Istio Service Mesh

Istio can provide mutual TLS (Transport Layer Security) authentication, which ensures that both the client and server are authenticated before communication takes place. This is important for preventing

unauthorized access, man-in-the-middle attacks, and data breaches. Istio uses the Envoy proxy to handle all network traffic, which allows it to manage mTLS encryption and decryption for each service. Istio also provides a certificate authority (CA) that can issue certificates for each service, allowing them to authenticate each other.



• Certificate Issuance & Management

Certificate Issuance & Management

Istio's Certificate Authority (CA) is not compliant with industry standards, which require CAs to follow strict procedures for certificate issuance and management. The root certificate and private key of the Istio Certificate Authority (CA) is stored within the cluster which can be a potential security risk and this can lead to vulnerabilities in certificate management.

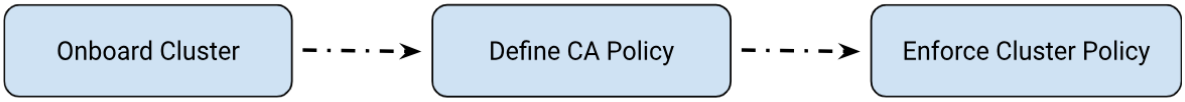
To eliminate the above there is a need to ensure the certificates in the control and data plane are rooted in the enterprise chain of trust but the real-world challenge with certificate management and Istio is how to integrate with existing enterprise PKI solutions.

Enabling AppViewX Signer

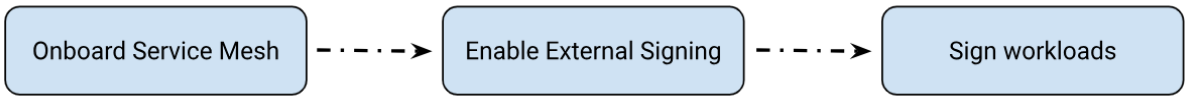
AppViewX Signer, the sub component of cert-orchestrator deployed in the Kubernetes cluster here solves the real-world challenge of Istio by signing the workload certificates with a trusted CA and installing the trust anchor within the cluster in auto enrolled fashion.

2 Steps to Enable the Zero Trust Security for Containers Using mTLS Certificates

1. Enforce PKI policies to ensure the use of compliant CAs and strong crypto-standards in your service mesh configuration.



2. Enable External CA signing mode for your Service Mesh configuration to sign workloads with mTLS certificates from your Enterprise PKI.



Steps to Enable a Signer for mTLS Certificate Issuance

To enable Signer for mTLS certificate issuance for your cluster, follow these below steps.

1. **Onboard Cluster** - Deploy / enable AppViewX Signer as a part of the KUBE+ component (cert-orchestrator).
2. **Policy Enforcement** - Define and enforce CA and Cluster Policy
3. **Onboard Mesh** - Configure CSR signing mode and the Certificate Authority to be used in Service Mesh.
4. **Enable External CA Mode** - Configure Service Mesh to External CA mode for CSR signing.

• Onboard Cluster

• Policy Enforcement for Secure ServiceMesh

Onboard Cluster

To deploy and enable signer component the KUBE+ component cert-orchestrator should be deployed in your Kubernetes cluster where the Service Mesh is enabled.

• New Cluster

• Existing Cluster

New Cluster

If you are deploying the cert-orchestrator for the first time in your cluster refer Onboaring a Cluster to obtain the deployment configuration for deploying cert-orchestrator.

> **Note:**
> While generating the deployment configuration select the Feature gate **Enable mTLS Certificates for Service Mesh** which enables AppViewX Signer as a part of the deployment.

Existing Cluster

For an already onboarded cluster in KUBE+ the signer component can be enabled via modifying the deployment configuration and executing the updated installation command at your cluster.

Refer to Modifying Feature Gates of Cluster for the steps on how to add (or) delete feature gates to an existing deployment.

> **Note:** While generating the modifying the deployment configuration select the Feature gate **Enable mTLS Certificates for Service Mesh** which enables AppViewX Signer as a part of the deployment.

Policy Enforcement for Secure ServiceMesh

To enable mTLS certificate issuance for application workloads from your Enterprise PKI, the PKI policies should be defined and enforced for your Service Mesh deployment.

The process for defining and enforcing the policy definition for your service mesh deployment is as follows.

1. **CA Integration** - Integrate AppViewX KUBE+ with your Internal CA for signing the certificates for your service mesh workloads.
2. **CA Policy** - Define CA Policy to enforce your organization crypto standards and map them to Certificate Groups (to categorize certificates based on business units).
3. **Enforce Cluster Policy** - Enforce dedicated CA Policy / PKI policy to one more cluster to promote secure and compliant certificate management practices.

CA Integration

Service Mesh can be integrated with your Enterprise Internal PKI only for signing application workloads with mTLS certificates.

AppViewX supports integrating with EJBCA and Microsoft CA for signing the mTLS service mesh workloads. Refer CA Integration for the steps on how to configure AppViewX KUBE+ with the respective Certificate Authority.

CA Policy

Configure certificate issuance parameters to enforce strong crypto standards for your mTLS workloads. Refer CA Policy for the steps on how to configure CA policy and Certificate Group for associate policies to certificate groups.

Cluster Policy

Configure Cluster Policy to enforce a dedicated CA/PKI policy for the service mesh external CA integration. Refer Create Policy for the steps on how to configure the Cluster Policy.

Note: For Service Mesh External CA signing the policy type must be set to **CA Setting Cluster** and the certificate authorities supported are EJBCA and Microsoft CA.

Mesh Inventory

Mesh Inventory Overview

Mesh Inventory displays a list of service mesh onboarded for a cluster. On Mesh Inventory page, you can:

• refresh the list, click the  (refresh) icon.

• go to the pages, click the (navigation) icon.

The mesh inventory list includes the following information:

Mesh Inventory - Column and Description Table

Mesh Name	Description
Name	Unique name to identify the mesh configuration for the designated cluster.
Cluster Name	Name of the Cluster for which the service mesh requires external CA signing.
CA Mode	Issuer's CA mode.
Vendor	The vendor name of the service mesh.
Certificate Authority	Certificate Authority to verify the identities of requesting certificates and links them to cryptographic keys by issuing digital certificates.
Created By	Refers to the user who created the mesh.

• Onboarding a Mesh

• Deleting a Mesh

• Significance of Issuer CA Mode

Onboarding a Mesh

To enable External CA signing for the Service Mesh deployed in your cluster, the Cert-orchestrator running with the Signer component needs to be enabled with a Certificate Authority Setting (CA Setting), a Kubernetes resource that represents the configuration of certificate authorities (CAs) responsible for generating signed certificates through certificate signing requests.

Prerequisites:

- CA Integration done.
- CA Policy created.

- Certificate Groups created.
- Cluster Policy created.

To onboard a mesh:

1. Go to **menu** > **KUBE+** > **Inventory** > **Mesh Inventory**.
2. Click **Onboard Mesh** on the menu bar.
3. On the **Onboard Mesh** page, enter/select the field information for the **General Information** and **Mesh Certificate Authority** sections.

General Information - Field and Description Table

Field	Description
General Information	
Name	Enter a unique name that can be used to identify the mesh configuration associated with the specified cluster.
Cluster	Select a cluster from the dropdown list in which the service mesh needs to be configured with an external CA for signing.
Vendor	Select a service mesh vendor from the dropdown list.
Mesh Certificate Authority	
Issuer CA mode	Select a radio button of Issuer CA mode. The options are: • via AppViewX - This option allows to send the workload certificate signing requests directly to AppViewX and signed by the configured CA Setting (Certificate Authority). The supported CA is EJBCA. • Air-Gapped - This option allows to sign the workload certificate signing requests by an Intermediate/SUB CA where the signing happens within the Kubernetes cluster. The Supported CAs are EJBCA and Microsoft CA.

Field	Description
Select Policy	Select the Cluster Policy from the dropdown list, which derives the associated CA for external CA signing.
Certification Authority	This field is not applicable, if you choose **via AppViewX** for Issuer CA mode. If you select Issuer CA mode as Air-Gapped, then enter/select the necessary details.
Ca Account	The account of the CA.
Common Name	Common name of the certificate.
Organization	Enter the name of the organization.
Organization unit	Enter the name of the organization unit.
Locality	Enter the locality of the certificate.
State	Enter the state of the certificate.
Country	Enter the country of the certificate.
Email Address	Enter the email address of the certificate.
Private Key Parameters	
Key Type	Select a key type of the certificate from the dropdown list. The values are: • RSA • ECDSA
Bit Length	Select a bit length for RSA or ECDSA. The values for RSA are: • 2048 • 4096 • 3072 The values for ECDSA are: • 256 • 384 • 521

4. Click **Generate YAML** to get the commands in the Issuer CA YAML field.

> **Note:**
>
> • To see the commands in the full screen view, click the  .
>
> • To copy the command, click  .

5. Click **Add** to add the mesh to the Mesh Inventory list.

Deleting a Mesh

You can delete mesh from the inventory, if the user chooses to restrict and remove CLM operations on the designated mesh.

To delete a mesh:

1. Go to **menu** > **KUBE+** > **Inventory** > **Mesh Inventory**.
2. Select the designated mesh, regardless of whether it is in a Managed or Unmanaged state.
3. Click **Delete** on the menu bar.

> **Note:**
>
> • A mesh can be removed from the inventory after deleting its associated configurations, including Cluster Policy, Issuer CA, and Certificates enrolled for the cluster, have been removed.
>
> • Deleting mesh from the inventory will not remove the in-cluster KUBE+ components. Users are required to follow the Uninstall and Clean Up steps within the mesh to complete the removal process.

Significance of Issuer CA Mode

Issuer CA mode for External CA signing plays a vital role in optimizing the speed of certificate issuance to your application workloads running as a part of the service mesh infrastructure.

AppViewX KUBE+ supports multiple modes of Issuer CA for mTLS certificate issuance.

- **Issuer Mode via AppViewX** : In this mode of issuing a mTLS certificate for a workload, the CSR requests are sent to AppViewX via API from the Cert-orchestrator (appviewx signer) deployed in your Kubernetes cluster to the cluster (or) environment where AppViewX is deployed.

  Example : Assume an on premises k8s cluster where your workloads are running and the CSR requests are sent to the AppViewX environment provided as a SaaS service to your enterprise. In this case the network hops and delays for the CSR request to land to AppViewX and then signed by CA will not be optimal.

  Also, the CSR generated by service mesh (Istio in this case) is valid for 10 seconds only for a workload and the entire process of signing the certificate by CA via AppViewX should be completed within this time frame.

  AppViewX recommends to use Issuer mode as AppViewX only when the clusters running workload and AppViewX environment are nearer to each other.

- **Issuer Mode via Air-Gapped** : In this mode of issuing a mTLS certificate for a workload, the AppViewX signer component running as a part of Cert-orchestrator generates a SUB CA / Issuing CA and the certificate and key used for signing the workloads are kept in the memory of the signer component itself for faster mTLS certificate issuance.

  The signer component also keeps the signing certificate and the private key encrypted in the k8s secret for reusing it when the cert-orchestrator pods or clusters are restarted.

  AppViewX recommends using Issuer mode as Air-Gapped when your AppViewX subscription is a SaaS service and when the clusters running workload and AppViewX environment are not at the nearest proximity.

Enable External Signing

Overview

Configuring the External CA configuration settings which includes Certificate Authority Settings and the Issuer CA mode, the AppViewX signer needs to be plugged in to the service mesh configuration to sign certificate from External CA

Prerequisites:

- Service Mesh onboarded in the Mesh Inventory.

Creating a Signer Profile

In the context of Service Mesh, the term "Singer" refers to Kubernetes resources that are responsible for configuring and enabling the external CA signing mode. These resources facilitate the provision of signed certificates to workloads within the Service Mesh.

To enable an external CA signing mode in service mesh:

1. Go to **menu** > **KUBE+** > **Cluster Security** > **Secure Service Mesh** .
2. Click 

.
3. On the **Create Signer** page, enter/select the field information.

Signer Creation - Field and Description Table

Field	Description
Name	Enter a unique name for the signer.
Mesh Name	Select a service mesh configuration associated to the cluster onboarded in earlier.
CA Mode	Select a designated issuance mode for the associated Mesh Certificate Authority.
Profile Name	Enter a unique profile name. For example, <domain>/istio.
External Secret Name	Enter the name of the secret where the trust store certificate should be synchronized.
External Secret Namespace	Enter the namespace of the secret.
Duration	Enter the duration in the hours.

4. Click **Generate YAML** to get the commands in the **Singer YAML** field.

Note:

- To see the commands in the full screen view, click the  .
- To copy the command, click  .

5. Click **Add** to add the signer to the Signer inventory list.

Integrating Istio with Custom CA

About Integrating Istio with Custom CA

AppViewX signer deployed in your cluster is now ready to sign all your service mesh application workload certificates from External Certificate Authority.

The Istio Service Mesh deployed in your cluster now needs to be reconfigured to enable Custom CA integration for provisioning workload certificates from External CA. Istio enables Custom CA integration using Kubernetes CSR API.

The root certificate for Custom CA in our case the Trust Certificate from EJBCA / Microsoft CA is loaded into the external ca secret configured in Signer Profile step. Refer to Custom CA integration steps in the Istio documentation to use the external CA secret.

Note: Change the Istio version accordingly in the URL to refer to the respective version of Istio deployed in your cluster.

Alerts and Logs

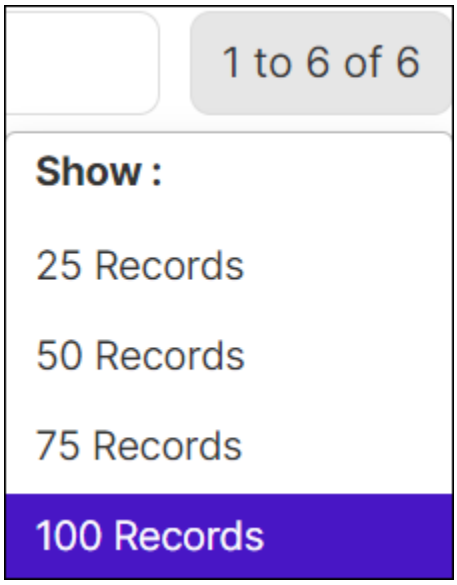
Logs

Logs keep track of all activities that take place within KUBE+ or any external entity that is connected to the AppViewX system. The Logging functionality in AppViewX tracks user activities and creates the service level event logs.

To access the logs related to KUBE+, go to **menu** > **KUBE+** > **LOGS** > **Logs** and then click the **Kube** tab, if not selected by default.

On the Logging page,

- go to the pages, click the (navigation) icon.
- hover the mouse over the number of row displayed on the page, the Show popup opens and choose the no. of rows to be displayed on the page.



Viewing Details of a Log

To view log details:

1. Go to **Menu** > **KUBE+** > **LOGS** > **Logs**.
2. On the **Logging** page, click the **Kube** tab.

3. On the log screen, view all details for a log by hovering your cursor over each column of data or click the ▸ (**Expand**) icon for the log you want to view. The table row expands to display all details for the log.

> **Note:** Search for any logs from the logging list based on User, Device name, log message, object details, source IP, AppViewX node, Method of login also search based on a particular column.

Filtering the Logs

To filter the logs:

1. Go to **Menu** > **KUBE+** > **LOGS** > **Logs**.
2. Click the **Kube** tab.
3. On the Logging page, follow any of the following method to filter the logs:
   - Click ⏲ (**DateTime**) icon, select the date range, and the click **OK** to view the logs within the date range.
   - Click the dropdown option beside the DateTime icon, and then select the preferred option. The options are:
     - Service
     - User
     - Type
     - Log Message
     - Severity

Alert Expiry

Digital certificates or clusters are issued with a validity period predefined by the certificate authorities. When a certificate or cluster outgrows this validity period, it is no longer trusted and the services consuming the certificate or cluster are considered broken.

AppViewX lets you create certificate or cluster expiry alerts, which are proactive notifications to customers to renew the certificates or clusters that are due for expiry.

The alerts are primarily created based on the period of certificate or cluster expiry (certificates or clusters expiring on a specific date, certificates or clusters expiring after x number of days, and so on). The emails also include a certificate renewal link, thus simplifying the CLM process.

The following sections list the prerequisites for configuring the expiry alerts and the configuration steps.

• Configuring a Certificate Expiry Alert

Configuring a Certificate Expiry Alert

Prerequisites:

Before configuring certificate expiry alerts over emails, ensure that permissions to create and modify the expiry alerts are assigned to the required role(s). To do this, follow the instructions given here and enable Expiry Alerts under Authorized Functions > KUBE+.

To configure a certificate expiry alert:

1. Go to **Menu** > **KUBE+** > **ALERTS & LOGS** > **Expiry Alerts**.
   The expiry alerts inventory is displayed.
2. To create a new email expiry alert, click **+ Create**.
   The **ExpiryAlert > Expiry Alert Add** page is displayed.

   Note: This page has a list of best practices you are advised to follow when creating a new expiry alert.

3. For the new alert, enter a unique **Alert Name** (mandatory) and a description (optional) with additional details about the alert.
4. In the **Alert Configuration** section, click **+ Add**.
   The **Add Configuration** pane is displayed.
5. Select the **Filter By** for which you want to configure the expiry alert.
   a. **Cluster**
   b. **Certificate**
6. For clusters, enter/select the **Filter Configuration** details.

Filter Configuration Section - Field and Description Table

Field	Description
Cluster Name*	**Note:** This field is displayed only when **Filter By = Cluster**.

Field	Description
	Select the clusters from the drop-down list for which you want to configure alerts.
Namespace*	**Note:** This field is displayed only when **Filter By = Cluster**. Select the corresponding namespace of the selected clusters from the drop-down list for which you want to configure alerts.
Certificate Expiry Period*	To filter clusters based on their expiry: • **Range in days**: Select this option to send an expiry alert email for clusters that will expire in the range of number days specified. For example, to set an alert for clusters that will expire between 10 to 40 days from today, enter **10** in the field before **to** and enter **40** in the field after **to**. • **On specified days**: Select this option to send an expiry alert email for clusters that will expire after x number of days from today. For example, to send an alert for clusters expiring on the 50th day days from today, enter 50. • **Range in dates**: Select this option to send an expiry alert for all clusters expiring in a specific duration. Use the calendar widget to select the **Start date** and **End date** of this duration.
Additional Filters	To add a filter in addition to the cluster expiry period: a. Select the required filter from the **Select option** dropdown list. b. Enter/select the corresponding value in the **Type to enter value** field.

Field	Description
	c. Click **Add**. d. The selected filter option and its value are displayed.
Notification Method*	To send an expiry alert over email or Slack message, select either **Email** or **Slack**.
Notification Format*	**Note:** This field is displayed only when **Notification Method = Email**. (Instead of emails for individual clusters, you will now receive a notification with a bulk of certificates in your chosen format.) Select the notification format to be sent via email, either as an attachment or in the email body using the table format. • **Attachment**: A CSV file with details of all clusters due to expire will be attached to the alert email. • **Email Body**: A table with details of all clusters due to expire will be included in the body of the alert email.
Ignore renewed certificate	When this field is enabled, results returned after the alert is triggered will not include clusters that have already been renewed, even though they match the filter criteria.
* - *Mandatory*	

7. Enter/select the **Email Configuration** details.

Email Configuration Section - Field and Description Table

Field	Description
To*	

Field	Description
Recipient(s)	Enter the recipient(s) email address. You can enter a maximum of 20 email addresses, separated by a comma.
Certificate Parameter(s)	To add more recipients for the alert email, you can retrieve email addresses from the following certificate parameters: • Subject Email • SAN (rfc822) • Certificate Group From the dropdown list, select the required parameter(s).
Certificate Attribute(s)	To add more recipients for the alert email, you can retrieve email addresses from any certificate attribute that has the email ID as its value, for example, **Cert Owner**. From the dropdown list, select the certificate attribute.
CC	
Recipient(s)*	Enter the email address of stakeholders who are to receive a copy of the expiry alert email.
Email Subject	
Subject*	Enter the subject line for the expiry alert email.
Email Body	
Body*	Enter the body text for the email using the following (recommended) format: The list of placeholder that can be used in the Subject <list and it's value> **Certificate with Common Name |Common Name|** **Serial number |Serial Number|** **issued by |Issuer Common Name|** **is about to expire on |Valid Until|**

Field	Description
	Note: The full list of placeholders and their corresponding values is given here.
Certificate Parameters For Email Body*	Select the certificate parameters from the drop-down menu that you want to include in the alert notification, either in the attachment or in the email body as a table. The following parameters are mandatory to be selected for the alert notification: • Common Name • Serial Number • Valid From • Valid Until
Certificate Attributes For Email Body*	Select the user defined certificate attributes from the drop-down menu that you want to include in the alert notification, either in the attachment or in the email body as a table.
* - *Mandatory*	

8. Click **Add**.
   The created alert is listed under the Expiry Alert page.

   Tip: To edit/delete an alert, use the **Edit/Delete** button, as required. You can add more alert configuration for the alert.

9. Under Alert Execution Type:

   Note: The settings configured here will be applicable to all expiry alerts listed in the above table.

   a. To trigger the alert immediately, select **Run Now** and click **Save**.
   OR

   b. To trigger the alert at a scheduled time, select **Schedule** and enter/select the following details:

- **Time Zone***
- **Frequency** (daily/weekly/monthly/yearly)
- **Starts On***
- **Ends**
  - For ends **ON**, use the calendar widget to select the date on which the last expiry alert email will be sent.
  - For ends **After**, enter the number of **Occurrences** after which the expiry alert will be discontinued.

10. Click **Save**.
    The alert notification trigger based on the selected notification alert and format.

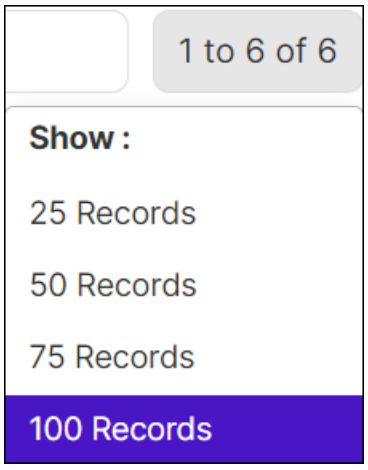
System Administration

- Crypto Minions
- Configuring Cluster Settings
- Configuring Policy Settings
- Configuring Email Settings
- Setting up Configurations for Expired Certificates
- Auto Enrollment

Crypto Minions

Crypto Minions displays the list of Crypto minions within a cert-orchestrator. On the Crypto Minions page,

- refresh the list, click the  (refresh) icon.
- go to the pages, click the (navigation) icon.
- hover the mouse over the number of row displayed on the page, the **Show** popup opens and choose the no. of rows to be displayed on the page.



The cluster inventory list includes the following information:

Column Name	Description
Cluster Name	Name of the cluster.
Status	Status of the cluster.
No. of Nodes	Nodes enabled for the cluster.
Status	Status of the resource. The possible statuses are Managed, Unmanaged, or Pending Approval.
Cert Orchestrator Version	The version of Cert Orchestrator.
CSI Provider Version	The version tag of the image.
Infra Orchestrator Version	The version of the Infra Orchestrator.

Configuring Cluster Settings

To configure repository settings:

1. Go to **Menu** > **KUBE+** > **System Administration** > **Cluster Settings**.
2. On the **Helm Repository** section, enter the details. Contact AppViewX support to get the authentication credentials for the Helm repository.

Helm Repository - Field and Description Table

Field	Description
Repository URL	Enter the helm repository URL.
Username	Enter the username.
Password	Enter the password.

3. When deploying AppViewX KUBE+ components in the cluster, kube rbac proxy image is used for RBAC authentication. This image is typically downloaded directly from the internet. However, if you need to pull the image from a custom repository, replace the URL and tag as follows:
   - URL: gcr.io/kubebuilder/kube-rbac-proxy
   - Tag: v0.13.0
4. On the Docker Repository section, enter the details. Contact AppViewX support to get the authentication credentials for the Docker repository.

Docker Repository - Field and Description Table

Field	Description
Repository URL	Enter the Docker repository URL.
Username	Enter the username.
Password	Enter the password.

Configuring Policy Settings

Policy settings manage the onboarding and offboarding process when a cluster is added to or deleted from the inventory. Configuring the policy automates the manual steps involved in integrating KUBE+ with the onboarded clusters and ensures data integrity and security when clusters are offboarded from KUBE +.

To configure policy settings:

1. Go to **Menu** > **KUBE+** > **System Administration** > **Policy Settings**.
2. Configure the off-boarding policy as follows:

a. Click **Yes** in the **Enable Off-boarding Policy** field.

b. In the **What needs to be Off-boarded from Cluster?** section, select any of the following options as needed:

- **Revoke Certificates deployed** - This option allows the removal of certificates associated with the cluster during the off-boarding process.
- **Revoke Issuer CA deployed** - This option allows the removal of the issuer CA associated with the cluster during the off-boarding process.

c. In the **What needs to be Off-boarded from AppViewX Inventory?** section, select any of the following options as needed:

- **Auto Manage Certificate Groups** - When the namepace is deleted in the cluster, all the resources under the certificate group (auto created) including the certificate and under the namespace will be deleted.
- **Delete Certificate YAML** - This option allows removing the certificate YAML from the AppViewX inventory.
- **Delete Issuer CA** - This option allows removing the issuer CA from the AppViewX inventory.
- **Disassociate Policy** - This option allows removing entry of cluster if policy is cluster wide. Otherwise, removes the namespace of the cluster.
- **Delete Certificates** - This option allows removing the certificates associated with the cluster.
- **Delete Certificate Groups** - This option allows removing the auto created certificate groups.
- **Delete Cluster** - This option allows removing the cluster from the Cluster inventory.

3. Configure the on-boarding policy as follows:
   a. Click **Yes** in the **Enable On-boarding Policy** field.
   b. Click **Yes** in the **Enable Auto provision of PKI Policy** field to onboard new clusters with automated policy and PKI configuration.
4. Click **Save**.

Configuring Email Settings

Within this section, you find email setting templates for certification action request. You have the capability to customize email IDs for each certification action requests. Once you've configured the email settings, emails will be automatically sent to the designated email addresses.

To configure email settings:

1. Go to **Menu** > **KUBE+** > **System Administration** > **Email Settings**.
2. Click on the preferred certification action request.
3. Enter the valid email IDs in the displayed fields.

   You can customize the field names by clicking on them and entering your preferred names. Additionally, you can add more fields by clicking the **Add** button. If any of the fields are not required, you can remove them by clicking the delete icon.

   Note: You can enter multiple email addresses, separated by commas.

4. Click **Save Changes**.

Setting up Configurations for Expired Certificates

You can configure the settings for handling expired certificates.

To configure the setting for the expired certificates:

1. Go to **Menu** > **KUBE+** > **System Administration** > **Expired Certificates**.
2. On **Expired certificate** section, enter/select the following details:

Expired certificate - Field and Description Table

Field	Description

Do you want to delete the expired certificates?	The options are:
	• **No** (default) - allows to retain the expired certificates.
	• **Yes** - to delete the expired certificates within the specified expiry date. When you select this option, you need to fill the following details:
	  • Number of days after expiry
	  • Backup Required
	  • Do you want to delete the certificates from all groups?

3. On **Expired root and intermediate certificate** section, enter/select the following details:

Expired root and intermediate certificate - Field and Description Table

Field	Description
Do you want to delete the expired root and intermediate certificates?	The options are: • **No** (default) - allows to retain the expired root and intermediate certificates. • Yes - to delete the expired root and intermediate certificates within the specified expiry date. When you select this option, you need to fill the **Number of days after expiry** field.

4. Click **Save**.

Auto Enrollment

• EST

EST

EST stands for Enrollment over Secure Transport, it is a certificate management protocol targeting Public Key Infrastructure (PKI) clients which require digital certificates to prove their authenticity. EST also helps the clients in acquiring associated CA certificates. Some prerequisites and features are as follows:

- There should be an agent (avx_vendor_cert_est_agent) up and running for EST in AppViewX.
- This EST plugin can either be in HTTPS or HTTP, but it needs a gateway which supports client certificate authentication running in order to communicate with the client.
- It is highly recommended to keep the 'Approval Required' flag OFF in the **Menu > KUBE+ > GROUPS & POLICIES > CA Policy** page.

• Configuring EST

• Validating EST

Configuring EST

To perform client certificate enrollments using EST protocol, the admin or a privileged user needs to first set up the EST server agent using the AppViewX portal. Upon successful set up of the EST server Agent through the portal, a URL will be generated. Clients can then use this URL to send enrollment requests to AppViewX via EST protocol.

The detailed steps for setting up the EST server agent are listed below:

1. Go to **Menu** > **KUBE+** > **System Administration** > **Auto Enrollment** > **EST**.
2. Select **Add** or **Configure Now.**
3. Configure the **End Point Details** details as follows:

   Prerequisites for entering the IP/FQDN field:

   - The "Cloud Connector Name" (in the Add Cloud connector page) must be the same as the FQDN name entered.
   - The CC should have the reachability to the Endpoint.
   - If entering the IP then ensure that a single cloud connector is used.

   The following table provides the field description for Agent Details section:

   End Point Details - Field and Description Table

Field Name	Field Type	Description	Validation
***Name**	Text	A unique name (alphanumeric string) to identify the agent setting. Acceptable Characters: A-Z, a-z, 0-9, '.', '_', '-'	Name should not start with special characters.
***FQDN/ IP**	Text	Enter the FQDN/IP address of the AppViewX cloud connector.	Invalid FQDN/IP address (example: xxx.xxx.xxx.xxx)
***Port**	Text	HTTP gateway port of the AppViewX node.	Port will accept only numerical values between 0 to 65535.
NOTE: Fields with * (asterisk) are mandatory.			

4. Configure the **Client Authentication** details as follows:

   Client Authentication = Only Certificate TLS

   Client Authentication = Certificate TLS with HTTPs as fallback or **Both Certificate TLS and HTTPs**

The following table provides the field description for CA Authentication section:

Details for CA Authentication- Field and Description Table

Field Name	Field Type	Description	Validation
Authentication Mode	Dropdown	Select any one authentication method to be carried out during communication with clients. • *Only Certificate TLS* - During client authentication, only certificate TLS based authentication will be performed. • *Certificate TLS with HTTPS fallback* - During client authentication, when the certificate TLS fails, HTTPs based authentication will be performed as fallback. • *Both Certificate TLS and HTTPs* - During client authentication, both certificate TLS and HTTPs authentication will be performed one after the successful completion of the other.	NA
***Issuer Certificate**	Dropdown	Select one or more issuer certificates which needs to be checked for the client certificate authentication.	NA
***HTTP Authentication Mode**	Radio button	Select the type of HTTP auth mode either Basic/Digest. • *Basic* - During Client authentication only the username and password values will be considered for HTTPs based authentication.	NA

Field Name	Field Type	Description	Validation
		• *Digest* - During Client authentication, along with username and password, nonce and realm values will also be supported.	
***Fallback Credentials**	Radio button	Select Manual/Logged on user credentials - based on the selection users can configure the credentials manually or save as credentials equivalent to the logged in user. • *Manual* - The Username and Password fields will be displayed to enter values. • *Logged on user credentials* - The Username and Password fields will not be displayed.	
***Username**	Text	Username for HTTP authentication.	NA
***Password**	Text	Password for HTTP authentication.	NA
NOTE: Fields with * (asterisk) are mandatory.			

5. Configure the **CA Accounts** details as follows:

   The following table provides the field description for CA Accounts section:

CA Accounts - Field and Description Table

Field Name	Field Type	Description	Validation
***Certificate Group**	Dropdown	Select a specific group under which certificate needs to be enrolled.	NA
***Certificate Category**	Radio button	Select a specific certificate type (Server/ Client) to be enrolled.	NA

Field Name	Field Type	Description	Validation
***Select CA**	Dropdown	Select the required CA from the available options. The certificate will be enrolled under the selected CA. The CAs associated with the Default certificate group are: • AppViewX • AppViewX PKIaaS • Amazon Private CA • DigiCert • DigiCert MPKI • Ejbca • Entrust • Entrust MPKI • GlobalSign Atlas • GlobalSign MSSL • Google • HydrantID • Microsoft Enterprise • Microsoft Standalone • Nexus **Note:** The Vendor Specific Details and Custom Attributes section are displayed for some of the CAs as follows: • DigiCert • EJBCA • Entrust • Entrust MPKI • GlobalSign MSSL • MS Enterprise • Nexus	NA
NOTE: Fields with * (asterisk) are mandatory.			

When **AppViewX** is selected as CA,The following table provides the field description for AppViewX CA:

Details for AppViewX CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Select	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***Server Certificate**	Select	Type 3 or more letters of the certificate keywords after which a list of server certificates issued from the above selected CA account will be displayed, one certificate can be selected for further communications with SCEP client machine.	NA
***CA Connector Name**	Text	Name of the CA connector after certificate is being enrolled.	NA
***Certificate Validity**	Text	Validity of the certificate to be enrolled.	Certificate validity accepts only numerical values
NOTE: Fields with * (asterisk) are mandatory.			

When **AppViewX PKIaaS** is selected as CA. The following table provides the field description for AppViewX PKIaaS CA:

Details for AppViewXPKIaaS CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***Issuer Location**	Dropdown	Select an issuer location that is associated with the CA account.	NA
***Pool Name**	Dropdown	Select a pool name to issue the certificate.	NA
***Issuer Name**	Dropdown	Select an issuer name to issue the certificate.	NA
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after certificate is being enrolled.	NA
***Certificate Validity**	Text	Validity of the certificate to be enrolled.	Certificate validity accepts only numerical values.

When **Amazon Private CA** is selected as CA. The following table provides the field description for Amazon Private CA:

Details for Amazon Private CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***Region**	Dropdown	Select a valid region associated with the CA account. The dropdown is populated with the first available value. Select an appropriate value as required.	NA
***Issuer**	Dropdown	Select a valid issuer associated with the CA account. The dropdown is populated with the first available value. Select an appropriate value as required.	NA
***Signature Algorithm**	Dropdown	Select a valid issuer associated with the CA account. The dropdown is populated with the first available value from the group's associated policy. Select an appropriate value as required.	NA
***CA Certificate**	Text	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after the certificate is enrolled.	NA
***Certificate Validity**	Dropdown	Validity of the certificate to be enrolled. (in years)	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

When **DigiCert CA** is selected as CA. The following table provides the field description for DigiCert CA:

Details for DigiCert CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***Division**	Dropdown	Select a division associated with the CA account. The dropdown is populated with the first available value. Select an appropriate value as required.	NA
***Certificate Type**	Dropdown	Select a valid cert type associated with the CA account. The dropdown is populated with the first available value. Select an appropriate value as required.	NA
***CA Certificate**	Text	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after the certificate is enrolled.	NA
***Certificate Validity**	Dropdown	Validity of the certificate to be enrolled. (in days/months/ years)	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

If the **Select CA =DigiCert**, then a separate section **Vendor Specific Details** is displayed after the CA Accounts section with two fields described below.

Vendor Specific Details for DigiCert CA - Field and Description Table

Field Name	Field Type	Description	Validation
***Server Type**	Dropdown	Select a server type. The dropdown is populated with the first available value. Select an appropriate value as required.	NA

Field Name	Field Type	Description	Validation
***Payment Method**	Dropdown	Select a payment method. The possible options are: 1. **Bill To Account Balance** - Pay with the account balance. Returns an error if this option is disabled for the account or if the account has an insufficient fund. 2. **Bill To Default Credit Card** - Pay with the account's default credit card. Returns an error if no default credit card is configured for the account	Alphanumeric characters, spaces, and the special characters -_.* are allowed.
NOTE: Fields with * (asterisk) are mandatory.			

When **DigiCert MPKI** is selected as CA. The following table provides the field description for DigiCert MPKI:

Details for DigiCert MPKI CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***Profiles**	Dropdown	Select a profile from the dropdown option.	NA
***CA Certificate**	Text	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after the certificate is enrolled.	NA
NOTE: Fields with * (asterisk) are mandatory.			

Note: Email address is a mandatory field on the enrollment form for DigiCert MPKI, but while passing it in the CSR, it is not added in certificate subject DN. Therefore, to successfully renew

DigiCert MPKI certificates using EST Fetch Certificate Parameters in EST Advanced Settings should be set to YES.

The **Custom Attributes** section is displayed on selecting the specific values from the **Profile** dropdown:

Custom Attributes for DigiCert MPKI CA - Field and Description Table

Field Name	Field Type	Mandatory	Description	Validation
***common_name**	Text	Yes	This field will be auto-populated from the CSR.	NA
***dnsName**	Text	Yes	Enter a valid DNS name.	NA
NOTE: Fields with * (asterisk) are mandatory.				

Note: Based on the DigiCert MPKI account configuration **Custom Attributes** section may also be displayed on the endpoint configuration page.

When **Ejbca** is selected as CA. The following table provides the field description for Ejbca CA:

Details for Ejbca CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA

Field Name	Field Type	Description	Validation
***CA Connector Name**	Dropdown	Name of the CA connector after certificate is being enrolled.	NA
***Certificate Validity**	Text	Validity of the certificate to be enrolled.	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

If the selected CA is Ejbca, a separate section **Vendor specific details** is displayed after the CA Accounts section. The following table provides the field description for Vendor specific details:

Vendor Specific Details for Ejbca CA - Field and Description Table

Field Name	Field Type	Description	Validation
***End Entity Profile Name**	Dropdown	Select a profile of an end entity.	NA
End entity user name	Text	Enter the user name for the end entity.	Alphanumeric characters, spaces, and the special characters -_.* are allowed.
***Issuer Common Name**	Dropdown	Select a common name of an issuer.	NA
***Certificate Profile Name**	Dropdown	Select a profile name of certificate.	NA
NOTE: Fields with * (asterisk) are mandatory.			

When **Entrust** is selected as CA. The following table provides the field description for **Entrust CA**:

Details for Entrust CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA

Field Name	Field Type	Description	Validation
***Certificate Type**	Dropdown	Select a valid cert type associated with the CA account. • If the **Certificate Category** radio button is selected to *Server***,** the dropdown is populated with the first available value. Select an appropriate value as required. • If the **Certificate Category** radio button is selected to *Client*, the dropdown is populated with 'None' as the default value.	NA
***CA Certificate**	Text	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after the certificate is enrolled.	NA
***Certificate Validity**	Dropdown	Validity of the certificate to be enrolled. (in days/months/ years)	Certificate validity accepts only numerical values.

• If the selected CA is **Entrust**, a separate section displaying **Vendor specific details** and **Custom Attributes** is displayed after the **CA Accounts** section.

Note: Based on the Entrust ECS account configuration **Custom Attributes section** may also be displayed as shown above.

Vendor Specific Details for Entrust CA - Field and Description Table

Field Name	Field Type	Description	Validation
Additional Emails	Text	Enter the valid email address in the field.	NA
Requester Name	Text	Enter the requester name	NA
Requester Email	Text	Enter a valid email id.	NA
Requester Phone	Text	Enter the 10-digit phone number.	NA

When **Entrust MPKI** is selected as CA. The following table provides the field description for Entrust MPKI CA:

Details for Entrust MPKI CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after certificate is being enrolled.	NA
NOTE: Fields with * (asterisk) are mandatory.			

When **GlobalSign Atlas** is selected as CA

The following table provides the field description for **GlobalSign Atlas CA**:

Details for GlobalSign Atlas CA - Field and Description Table

Field Name	Field Type	Description	Validation
***Select CA**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
API Credential	Dropdown	Select a CA Account to communicate with during the certificate enrollment actions.	NA

Field Name	Field Type	Description	Validation
Friendly name			
Certificate Profile	Dropdown	Select the certificate Profile from the dropdown.	NA
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Select	Name of the CA connector after the certificate is enrolled.	NA
***Certificate Validity**	Dropdown	Validity of the certificate to be enrolled. (in days/months/ years)	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

A **Generic Fields** section is also displayed below the CA Accounts section. It contains the fields related to the CSR parameters based on the profile (API Credential Friendly name) selected. Only the Organization field is mandatory and is fetched from the selected profile. Rest of the fields are optional.

- When **GlobalSignMSSL** is selected as CA,

  The following table provides the field description for GlobalSignMSSL CA:

Details for GlobalSign MSSL CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***Product Type**	Dropdown	Select the specific Product Type. The values are fetched from the CA Settings configuration.	NA

Field Name	Field Type	Description	Validation
***CA Certificate**	Select	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Select	Name of the CA connector after the certificate is enrolled.	NA
***Certificate Validity**	Dropdown	Validity of the certificate to be enrolled. (in days/months/ years)	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

The following field is displayed in the **Vendor Specific Details** section as per the selected CA:

Vendor Specific Details for GlobalSign MSSL CA - Field Description Table

Field Name	Field Type	Description	Validation
***Profile name**	Dropdown	Select the Profile based on the configurations made in the Certificate Authority Setting.	NA

The following field is displayed in the **Point of Contact** section as per the selected CA. The CA mandates the point of contact information for traceability. All auto-enrollment requests via this endpoint will be registered with the point of contact information entered here.

POC Details for GlobalSign MSSL CA - Field and Description Table

Field Name	Field Type	Description	Validation
***First Name**	Text	Enter the first name	NA
***Email Address**	Text	Enter the valid email address	NA
***Phone Number**	Text	Enter the valid phone number	NA

When **Google** is selected as CA. The following table provides the field description for Google CA:

Details for Google CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***Certificate Profile**	Dropdown	Select the certificate profile type.	NA
***Issuer Location**	Dropdown	Select an issuer location that is associated with the CA account.	NA
***Pool Name**	Dropdown	Select a pool name to issue the certificate.	NA
***Issuer Name**	Dropdown	Select an issuer name to issue the certificate.	NA
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after certificate is being enrolled.	NA
***Certificate Validity**	Text	Validity of the certificate to be enrolled.	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

When **HydrantID** is selected as CA. The following table provides the field description for HydrantID CA:

Details for HydrantID CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***HydrantID Policy**	Dropdown	Select the policy associated with the CA Account to be used for certificate operations.	NA
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after certificate is being enrolled.	NA
***Certificate Validity**	Text	Validity of the certificate to be enrolled.	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

When **Microsoft Enterprise** is selected as CA. The following table provides the field description for Microsoft Enterprise CA:

Details for Microsoft Enterprise CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after certificate is being enrolled.	NA
***Certificate Validity**	Text	Validity of the certificate to be enrolled.	Certificate validity accepts only numerical values.
NOTE: Fields with * (asterisk) are mandatory.			

- If the selected CA is Microsoft Enterprise, a separate section **Vendor specific details** is displayed with a **Template Name** dropdown after the CA Accounts section.

When **Microsoft Standalone** is selected as CA. The following table provides the field description for Microsoft Standalone CA:

Details for Microsoft Standalone CA - Field an Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to	NA

Field Name	Field Type	Description	Validation
		be used for certificate creation operations.	
***CA Certificate**	Dropdown	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Text	Name of the CA connector after certificate is being enrolled.	NA
NOTE: Fields with * (asterisk) are mandatory.			

When **Nexus** is selected as CA,

The following table provides the field description for **Nexus CA**:

Details for Nexus CA - Field and Description Table

Field Name	Field Type	Description	Validation
***CA Account**	Dropdown	Select a specific CA Account from the selected CA which is to be used for certificate creation operations.	NA
***CA Certificate**	Select	Select the specific issuer certificate, that will be used for signing CSR by the certificate authority. Certs available in the root or intermediate cert inventory are displayed.	NA
***CA Connector Name**	Select	Name of the CA connector after the certificate is enrolled.	NA
***Certificate Validity**	Select	Validity of the certificate to be enrolled. (in days/months/ years)	Certificate validity accepts

Field Name	Field Type	Description	Validation
			only numerical values.

The following field is displayed in the **Vendor Specific Details** section as per the selected CA:

Details for Nexus CA - Field and Description Table

Field Name	Field Type	Description	Validation
***Procedure**	Dropdown	Select the Procedure based on the configurations made in the Certificate Authority Setting.	NA

6. Configure the **Advanced Settings** details as follows:
   The following table provides the field description for Advanced Settings:

Advanced Setting - Field and Description Table

Field Name	Field Type	Description
***Switch to Enroll**	Radio button	Select Yes or No Selecting the radio button as Yes will convert the re-enrollment requests to enrollment requests
***Fetch Certificate Parameters**	Radio button	Select Yes or No Setting the radio button to Yes, will enable the system to automatically fetch certificate parameters from a Suggestive Policy, and append them to the client CSRs.
***Include Truststore Certificates**	Radio button	Select whether the issuer certificate needs to be sent to client machines after enrolment.
***High Speed Transactions**	Radio button	Based on the selection of this field, the endpoint will be

Field Name	Field Type	Description
		configured with or without High Performance transaction times. Request information pertaining to High-Performance can be viewed on the Direct Requests page.
***Return Existing Certificate**	Radio button	If this option is enabled (**Yes**) then for request with AppViewX should check and return the existing valid certificate for the same CSR & public key from inventory if available otherwise it should proceed with enrollment and return the certificate. • If it is set to **Yes**, the Certificate Threshold field is displayed. If the option is disabled (**No**) then the AppViewX will do the default behavior of enrolling a new certificate for each request.
Certificate Threshold	Text (numeric)	This field is enabled only if **Return Existing Certificate = Yes.** Enter the number of days in this field. This value is used to Initiate a new certificate request if the certificate is nearing the expiry date i.e., if existing certificate validity is less than the entered value.
***Retry Count**	Text	Values accepted between 5 - 99.

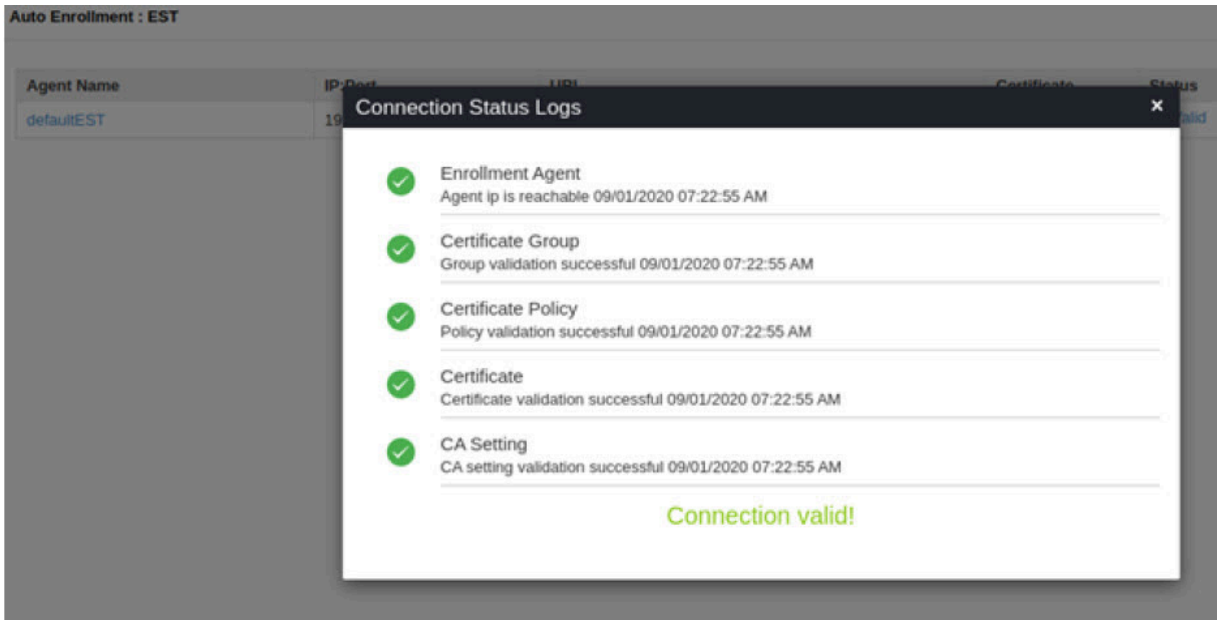
Field Name	Field Type	Description
		Based on this value, the EST agent will trigger the number of calls to collect the certificate from AppViewX until it is received.
***Retry Frequency**	Text	Values accepted between 10 - 99. The value specified in this field determines the duration taken between the trigger calls by the EST agent.
Note: Fields with * (asterisk) are mandatory.		

7. Click **Save**.

Validating EST

Once the EST settings are added validation needs to be done to check whether the CA and Agent-related details are properly configured.

1. Go to **Menu** > **KUBE+** > **System Administration** > **Auto Enrollment** > **EST**.
2. On the **EST** page, click **Check** to validate the EST setting that has been created.
   The EST settings will be validated and the **Status** will be shown as either **Success** or **Failure.**



Notification Event

Event notifications for a cluster can be configured for various events. The following events can be set up from the Notification Policies:

- Cluster status changes to unhealthy
- Cluster service account expiry

Configuring Notification Event

To view and configure notification events for a cluster:

1. Click (**Notification icon**) located at the upper right corner of the screen.
   The triggered notification are listed.
2. Click **Action Center**.
3. On the **Notification Policies** screen, sselect the appropriate field information to configure notifications for the cluster status changing to unhealthy.

Notification Policies - Field and Description Table

Field	Description
Event	Select an event from the dropdown list. The KUBE+ specific events are: • **Kube cluster status unhealthy Notification**: Triggers a notification when the cluster status changes unhealthy. • **Kube cluster service account expiry Notification**: Triggers a notification 15 days before the service account expires.
Channels	Select the channels for notification. You can select multiple options from the following: • **Push Notification**: Sends a push notification to the AppViewX application. • **Email**: Sends the notification via email.
Recipient Category	Select the recipient category for notifications. Recipients will be displayed based on the selected options: • **User**: Allows selection of users from the dropdown menu who have the ACF permission to receive this notification. • **User group**: Allows selection of user groups from the dropdown menu who have the ACF permission to receive this notification. • **Email / Distro**: Allows selection of an email or email distribution list from the dropdown menu for those with ACF permission to receive this notification.

4. Click **Update**.
   The event is added to the Event inventory.

Enabling Auto-Workflow Initiation for ConfigMap Creation

Whenever a ConfigMap is created, Cert-Orchestrator will read it and automatically proceed with workflow creation if the auto-trigger workflow option is enabled

Prerequisites

To enable auto-trigger workflow option:

1. Enable auto-trigger workflow option in helm by setting the value to true `--set certOrchestrator.autoTrigger.workflow=true`.
2. Ensure the annotation matches and contains the workflow name.

   The sample configmap json file to trigger the workflow:

```
apiVersion: v1
data:
  payload: |
       {
         "globalData": {},
         "task_action": 1
       }
kind: ConfigMap
metadata:
  name: my-config
  namespace: default
  annotations:
    appviewx.com/autotrigger: "true"
    appviewx.com/workflowname: "test_workflow"
```

Uninstall and Cleanup

Steps to Uninstall/Cleanup KUBE+ Deployment

The in-cluster components of KUBE+ deployed in the Kubernetes cluster must be completely uninstalled, if the administrator wishes to cease certificate lifecycle management for the designated cluster.

To achieve this, execute the following commands to uninstall the KUBE+ in-cluster component Cert-Orchestrator from your cluster.

- Uninstall the Cert-Orchestrator by executing the following Helm command:

```
helm uninstall crypto-mesh -n crypto-mesh.
```

- Uninstall the CSI provider if the Ephemeral Volume Feature gate has been installed by executing the following command:

```
helm uninstall csi -n crypto-mesh
```

> **Note:** If the CSI provider is already present in the cluster and was not installed as a part of the cert-orchestrator deployment you can skip the uninstall of the CSI provider step.

- Remove the KUBE+ helm repositories from your cluster by executing the following commands:

```
helm repo remove crypto-mesh

helm repo remove csi
```

> **Note:** You should proceed with the removal of the CSI provider from the repository only if it was enabled during the Cert-Orchestrator installation. If the CSI provider was already available, you can skip this step.

- Even after deleting the components of Cert-Orchestrator, Kubernetes secrets, configmaps, and other associated metadata may still remain in the cluster. You can remove this residual data by deleting the namespace where Cert-Orchestrator is installed.

```
kubectl delete ns crypto-mesh
```

> **Note:** If a custom namespace was used during the installation of Cert-Orchestrator, please replace 'crypto-mesh' with your specified custom namespace.

- During the installation of Cert-Orchestrator, it deploys multiple Custom Resource Definitions (CRDs) within the Kubernetes cluster. To remove these CRDs manually, execute the following command:

```
kubectl get crd | grep appviewx | awk ‘{print $1}’ | xargs kubectl delete crd
```

The steps outlined above completely remove the KUBE+ in-cluster components from the cluster. If the Cert-Orchestrator has been successfully uninstalled, the following commands will produce an empty output.

The commands are:

- `kubectl get namespace | grep crypto-mesh`

> **Note:** If the namespace name is not 'crypto-mesh,' replace it with your custom namespace.

- `kubectl get crd | grep appviewx`
- `helm repo list`

> **Note:** The command above should exclude both 'crypto-mesh' and the CSI repo.

- `kubectl get pods -n crypto-mesh`

> **Note:** The command above should not display any running pods.

FAQ and Troubleshooting

Command Line Cheat Sheet

List of kubectl commands for configuring and troubleshooting issues pertaining to KUBE+ in-cluster components.

Command Line Cheat Sheet Table

S. No.	Command	Purpose
1	`kubectl get all --all-namespaces`	lists resources (pods, services, deployments, etc.) in all namespaces.
2	`kubectl get namespace <namespace_name>`	Retrieves information about a specific namespace in a Kubernetes cluster.
3	`kubectl delete namespace <namespace_name>`	Deletes a specific namespace and all the resources contained within it in a Kubernetes cluster.

Command Line Cheat Sheet Table (continued)

S. No.	Command	Purpose
4	`kubectl get pod`	List all the pods in the current Kubernetes namespace.
5	`kubectl get pods -n=[namespace_name]`	lists all the pods in a specific namespace in a Kubernetes cluster. Replace "[namespace_name]" with the actual name of the namespace you want to target. This command displays information about the pods in that namespace
6	`kubectl delete pod <pod_name>`	Deletes a specific pod in a Kubernetes cluster.
7	`kubectl get secrets`	Lists all the secrets available in the current Kubernetes namespace.
8	`kubectl describe secret <secret_name>`	Displays detailed information about a specific secret in a Kubernetes cluster.
9	`kubectl delete secret <secret_name>`	Deletes a specific secret in a Kubernetes cluster.
10	`kubectl apply -f manifest_file.yaml`	Applies a configuration to an object by filename or stdin.
11	`kubectl logs -f <pod_name>`	Prints the logs for a pod and it will continuously stream the logs as new log entries that are generated in real-time. This is useful for monitoring and troubleshooting applications running in Kubernetes pods..
12	`kubectl logs -c <container_name> <pod_name>`	Prints the logs from a specific container within a pod in a Kubernetes cluster.
13	`kubectl logs <pod_name> pod.log`	Redirects and saves the logs from a specific pod in a Kubernetes cluster to a file named ‘pod.log’.

FAQ

1. How to configure HA ?
   - You can configure HA by increasing the number of pods running, By default this is set to one.
   - `"certOrchestrator.replicaCount"` field needs to be overridden in the deployment.
   - Can use `--set` flag to override the same as below using helm
     - --set `certOrchestrator.replicaCount=2` (example to increase the number of pods to two)
   - Always only one pod will be a leader, other pods will be non-leader, when the leader is down, an election will happen to elect the leader among the existing non-leader + new pod (created due to the kill of old leader), based on the election one of the pod will become Leader.
2. How to check if your pod is up and running ?How to configure monitoring for your pod running in a cluster?

   You can monitor the pod livenessProbe under the below path from the cert-orchestrator pod

   `path: /healthz`

   `port: 8081`
3. How to add tolerations to the deployment pod?

   Tolerations can be overridden using helm, Pl refer the helm chart configuration.
4. What is the permission to be allowed in the cluster for running your pod?
   - cert-orchestrator
     - configmaps
   - create, get
     - namespaces
       - get, list
   - nodes
     - get, list, watch
   - pods
     - get, list
   - secrets
     - create, delete, get, list, patch, update, watch
   - cert-orchestrator.certplus.appviewx:casettingclusters

    - create, delete, get, list, patch, update, watch
- cert-orchestrator.certplus.appviewx:casettingclusters/status
    - get, patch, update
- cert-orchestrator.certplus.appviewx:casettings
    - create, delete, get, list, patch, update, watch
- cert-orchestrator.certplus.appviewx:casettings/status
    - get, patch, update
- cert-orchestrator.certplus.appviewx:certreqs
    - create, delete, get, list, patch, update, watch
- cert-orchestrator.certplus.appviewx:certreqs/status
    - get, patch, update
- cert-orchestrator.certplus.appviewx:certs
    - create, delete, get, list, patch, update, watch
- cert-orchestrator.certplus.appviewx:certs/finalizers
    - update
- cert-orchestrator.certplus.appviewx:certs/status
    - get, patch, update
- cert-orchestrator.certplus.appviewx:discoveryrequests
    - create, delete, get, list, patch, update watch
- cert-orchestrator.certplus.appviewx:discoveryrequests/finalizers
    - update
- cert-orchestrator.certplus.appviewx:discoveryrequests/status
    - get, patch, update
- cert-orchestrator.certplus.appviewx:renewaljobs
    - create, delete, get, list, patch, update, watch
- cert-orchestrator.certplus.appviewx:renewaljobs/status
    - get, patch, update
- cert-orchestrator.certplus.appviewx:signers
    - create, delete, get, list, patch, update, watch

- cert-orchestrator.certplus.appviewx:signers/finalizers
  - update
- cert-orchestrator.certplus.appviewx:signers/status
  - get, patch, update
- certificates.k8s.io:certificatesigningrequests
  - get, list, patch, update, watch
- certificates.k8s.io:certificatesigningrequests/status
  - get, patch, update
- certificates.k8s.io:[SIGNER_NAME]/istio:signers
  - sign
- coordination.k8s.io:leases
  - create, delete, get, list, update, watch
- events
  - create, patch
- networking.k8s.io:ingresses
  - get, list,watch
- secrets-store.csi.x-k8s.io:secretproviderclasses
  - create, delete, get, list, patch, update, watch
- secrets-store.csi.x-k8s.io:secretproviderclasses/finalizers
  - update
- secrets-store.csi.x-k8s.io:secretproviderclasses/status
  - get, patch, update
- appviewx-infra-orchestrator
  - cert-orchestrator.certplus.appviewx:discoveryrequests
    - create, get
- appviewx-csi-provider
  - serviceaccounts/token
    - create
  - secrets

- create, get

- cert-orchestrator.certplus.appviewx:certs
  - create, get

5. Whats is the permission given for SA (cluster role and cluster role binding)?

   Refer to the answer provided for the previous question.

6. How to configure resource requirements?

   Refer to the helm chart configuration under `certOrchestrator.resources`.

cert-orchestrator : Helm chart configuration parameters

Qualifier	Parameter	Definition	Allowed Values
`certOrchestrator`	enabled	Enable certOrchestrator.	true / false
	`renewalEnabled`	Enable renewal.	true / false
	`namespace`	Namespace for the cert-orchestrator installation.	Valid namespace name
`certOrchestrator.discovery`	`enabled`	Enable Discovery	true / false
	`isGroupAutoGenerate`	Allow auto group creation at AppViewX.	true / false
	`credentialSecretName`	Secret with credentials to be used for Discovery with AppViewX.	Valid Secret Name
	`credentialSecretNamespace`	Namespace for the above.	Valid namespace Name
`certOrchestrator.global`	`logLevel`	Log level for the cert-orchestrator terminal log.	0 to 7
	`clusterName`	Name of the cluster for the current installation.	Valid Cluster Name

Qualifier	Parameter	Definition	Allowed Values
	k8sVendor	Type of vendor where the cert-orchestrator runs.	Valid vendor Name
certOrchestrator.image	repository	Repository name for the image	Valid image name with repo
	tag	tag for the image	Valid image tag
	pullPolicy	Image Pull Policy	Always, Never or IfNotPresent. Defaults to IfNotPresent
certOrchestrator.resources	limits.cpu	Describes the maximum amount of CPU allowed.	Default is 1000m, See Kubernetes - meaning of CPU
	limits.memory	Describes the maximum amount of Memory allowed.	Default is 1Gi. see Kubernetes - meaning of Memory
certOrchestrator.resources	requests.cpu	Describes the minimum amount of CPU required.	Default is 500m, see Kubernetes - meaning of CPU
	requests.memory	Describes the minimum amount of Memory required.	Default is 500Mi. See Kubernetes - meaning of Memory
certOrchestrator	tolerations	Describes the tolerations allowed for the pods to schedule.	

appviewx-csi-provider : Helm chart configuration parameters

Qualifier	Parameter	Definition	Allowed Values
appviewxCSIProvider	enabled	Enable appviewxCSIProvider.	true / false
appviewxCSIProvider.image	repository	Repository name for the image.	Valid image name with repo.

Qualifier	Parameter	Definition	Allowed Values
	tag	Tag for the image.	Valid image tag
	pullPolicy	Image Pull Policy	Always, Never or IfNotPresent. Defaults to IfNotPresent
certOrchestrator	tolerations	Describes the tolerations allowed for the pods to schedule.	

appviewx-signer : Helm chart configuration parameters

Qualifier	Parameter	Definition	Allowed Values
appviewxSigner	enabled	Enable appviewxSigner.	true / false

appviewx-infra-orchestrator : Helm chart configuration parameters

Qualifier	Parameter	Definition	Allowed Values
appviewxInfraOrchestrator	enabled	Enable certOrchestrator.	true / false
	tick	Sync frequency for the certificate scan.	Valid time period string. Example : "60m"
appviewxInfraOrchestrator.imag	repository	Repository name for the image.	Valid image name with repo
	tag	Tag for the image.	Valid image tag
	pullPolicy	Image Pull Policy	Always, Never or IfNotPresent. Defaults to IfNotPresent
appviewxInfraOrchestrator	tolerations	Describes the tolerations allowed for the pods to schedule.	

Chapter 2: KUBE+ API Guide

Understanding the AppViewX KUBE+ API

The AppViewX API is a programmatic way to get data in and out of the AppViewX subsystems. With access to RESTful AppViewX APIs, you can leverage the raw potential of AppViewX. It provides a powerful way to channel the data into native business applications. This document comprises of module-wise APIs used in AppViewX.

RESTful HTTPS Requests

Type	Description
GET	GET requests, retrieve resource representation/information only and not to modify it.
POST	POST APIs create new subordinate resources. For example, a file is subordinate to a directory containing it or a row is subordinate to a database table. In terms of REST, POST methods are used to create a new resource into the collection of resources.
PUT	PUT APIs are used to update existing resources (if a resource does not exist then API may decide whether to create a new resource or not).
DELETE	DELETE APIs are used to delete resources (identified by the Request-URI).

Requests

Each endpoint URL is built in the same way by the following structure:

```
http://<IP/HostName/TenantName>:<GWPORT>/avxapi/<Endpoint>?<gwsource>
```

The following section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Request Structure

All endpoints accept a request structure that should consist of JSON formatted data. To ensure the request is accepted, set the header **Content-Type: application/json**.

The following example shows a request to add a resource:

```
{
"payload": {
```

```
"name": "resource_1",
"description": "This is a sample resource."
}
}
```

Response Structure

The Content-Type of the response is typically determined by the Content-Type header, and for most endpoints, it will be application/json. All requests that reach the server, regardless of the response code, will retrieve a response body. A successful request will contain a body with the requested information, for example:

```
https://appviewxapi.com/avxapi/resource?gwsource=external
```

Returns the following JSON structure that a resource is added:

```
{
"response": "Resource added successfully",
"message": "Resource added successfully",
"appStatusCode": null,
"tags": null,
"headers": null
}
```

Description of Server Responses

HTTP Code	Response Message
200 OK	The request was successful (some API calls may return 201 or 202 instead).
400 Bad Request	The request is not understood or required parameters are missing.
401 Unauthorized	Authentication failed or the user doesn't have permissions for the requested operation.
409 Forbidden	Access denied.
404 Not Found	Resource not found.
429 Too many requests	The number of requests to the service has crossed the threshold.
503 Service unavailable	The client cannot communicate with the service.

HTTP Code	Response Message
504 Gateway timeout	The given request has exceeded the expected time.

URI Scheme

- **Host :** {url}
- **BasePath :** /avxapi
- **Schemes :** HTTPS
- **URL :** https://{url}/avxapi

Types of Accounts in AppViewX

There are two types of accounts in AppViewX:

- **User Accounts**: These are used by actual users.
- **Service Accounts**: These are used by system services such as web servers, automation tools, and so on.

AppViewX recommends using a Service Account for accessing APIs from automation tools. Service Accounts are supported with oAuth standard for a more secure and standard way of accessing APIs.

Note: AppViewX supports both User Account and Service Account for accessing APIs.

Authentication using a User Account

For accessing APIs, you can login via two types of accounts:

- User account

  A **user account** represents an individual person interacting with the application or the system. User accounts are used for accessing the system on behalf of a user.

  For accessing APIs with a user account, you need to get the session ID by providing a username and password in the login API. This session ID can then be used for accessing other APIs.

  Note: You can also use the username and password in all API calls instead of the sessionId. However, this is not recommended.

- Service account

A **service account** represents a non-human entity such as an application or a service. It is used for automated processes or system-to-system interactions without human intervention.

For accessing APIs with a service account, you need to get the Access Token by providing Client ID and Client Secret in get-service-token API. This Access Token can then be used for accessing other APIs.

Note: Access Token Validity is 30 minutes by default and it can be configured in **Settings** > **Authentication** > **oAuth Settings**.

For accessing APIs with a user account, the subsequent sections will help you with instructions to:

- Retrieve session ID using login API
- Using Session ID for further API calls

Retrieve session ID using login API

This API used to retrieve the session ID using the login API for secure authentication and access to system resources.

Before you begin

- Make sure you have valid login credentials (Username and Password) for accessing the system.
- You cannot use OAuth credentials (Client ID and Client Secret) for login.
- To access the APIs using the service token, use the API with the Service Account.

Request Structure

Endpoint	`/login`
Type	`POST`
Sample URL	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/login?&gwsource=external` To understand the elements of the sample URL, click here.
Headers	
Content-Type	application/json
Request timeout period	15 minutes

Input Parameters

	Description
username *Header*	(Mandatory) Use login name of the user. **Type:** String **Example:** "admin"
password *Header*	(Mandatory) Password for the username. **Type:** String **Example:** "AppViewX@123"
otp *Header*	(Mandatory only if MFA is enabled) If MFA is enabled, enter the OTP received on your registered email ID in the header. Multifactor authentication (MFA) is a security mechanism that requires users to provide two or more verification factors to gain access to a resource If MFA is enabled, and you try to login with only the username and password, you will get the following error upon execution of the API: **MFA is enabled. We have sent an OTP to your email ID: aaa********r@appviewx.com**. In this case, ensure that the OTP is included in the header and try logging in again. **Type:** String **Example:** "OTP : **609700**"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Type:** String **Example:** "application/json"
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external **Type:** String

Response Structure

- **Status Code:** 200 Ok
- **Message:** Login Successful
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	The response contains the attributes needed to retrieve the session ID.
message	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Name	Description
status	Indicates the overall status of the response. The values can be: • SUCCESS • FAILURE
appStatusCode	An application-specific status code, if applicable.
statusDescription	Description of the status, if available.
sessionId	Unique identifier for the session.
lockDownPeriod	Number of login attempts remaining.
termsAccepted	
passwordExpiryMsg	
emailId	

Status Codes

HTTP Code	appStatusCode	Response Message
200 OK	NA	Login successful

HTTP Code	appStatusCode	Response Message
400 Bad request	ACCT_AUTH_001	Username or password cannot be null or empty.
401 Unauthorized	ACC_AUTH_022	Login failed. Invalid credentials.
401 Unauthorized	ACC_AUTH_006	Login failed. Invalid credentials.

Sample Request/Response

Use Case

Log in to the application with a username and password.

Sample Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/login?&gwsource=external
```

Request Payload

```
{}
```

Sample Response

```
{
    "response": {
        "status": "SUCCESS",
        "appStatusCode": null,
        "statusDescription": null,
        "sessionId": "avx~~c73a4f56-f4ab-4cdf-aadf-6d90bf406077",
        "authCode": null,
        "lockDownPeriod": 15,
        "emailId": null,
        "termsAccepted": true,
        "passwordExpiryMsg": ""
    },
    "message": "Login successful.",
    "appStatusCode": null,
    "tags": null,
    "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What can you do next?

After the sessionID ris etrieved using the login API, you can now use the session ID for API calls.

Using Session ID for further API calls

The sessionID retrieved using the login API can be used in the header for making further API calls.

In this section, as an example, we are using the session ID with the API call for managing a cluster.

Before you begin

- Session ID is obtained from the login API.
- Ensure that the session ID is valid and has not expired.

Request Structure

Endpoint:	/kube-manage-cluster
Type:	PUT
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-manage-cluster?gwsource=external` To understand the elements of the sample URL, click here.
Headers	
Content-Type:	application/json

Input Parameters

Name	Description
sessionId *Header*	(Mandatory) Session Id received after login. **Type**: String **Constraint**: Required if username and password are not provided.
userName *Header*	(Mandatory only if **sessionID** is not provided) Username that is configured in AppViewX. **Type**: String
password *Header*	(Mandatory only if **sessionID** is not provided) Password of that user. **Type**: String
content-type *Header*	(Mandatory) Payload content-type with application/json value. **Type**: String **Constraint**: The value must be **application/json**.
gwkey *Query*	(Mandatory) Tenant Key. This is required only for multi-tenant installations and can be disregarded for other types of installations. **Type**: String
gwsource	(Mandatory) The source from which the request is triggered, e.g., external.

Input Parameters (continued)

Name	Description
Query	**Type**: String
Payload *Body*	Contains all the parameters to be sent in the request body for the put request. **Type**: Payload

Payload

Input Parameter

Name	Description
clusterName *List<String>*	List of cluster names available in the inventory that needs to be manged.

Response Structure

Name	Description
response *String*	The response contains the unmanage state of the cluster.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Manage a cluster.

Request URL

https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-manage-cluster?gwsource=kube

Sample Request

```
POST Content type: application/json Username: <> Password: <>

{

  "payload": {
```

```
  "clusterNames": [
    "casetting-cluster-appviewx-ca-server-ud-gp"
  ]
 }
}
```


Sample Response


```
{
 "response": null,
 "message": "Clusters are Managed",
 "appStatusCode": "success",
 "tags": null,
 "headers": null
}
```


Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL

- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Authentication using a Service Account

For accessing APIs, you can login via two types of accounts:

- User account

  A **user account** represents an individual person interacting with the application or the system. User accounts are used for accessing the system on behalf of a user.

  For accessing APIs with a user account, you need to get the session ID by providing a username and password in the login API. This session ID can then be used for accessing other APIs.

  > **Note:** You can also use the username and password in all API calls instead of the sessionId. However, this is not recommended.

- Service account

  A **service account** represents a non-human entity such as an application or a service. It is used for automated processes or system-to-system interactions without human intervention.

  For accessing APIs with a service account, you need to get the Access Token by providing Client ID and Client Secret in get-service-token API. This Access Token can then be used for accessing other APIs.

  > **Note:** Access Token Validity is 30 minutes by default and it can be configured in **Settings** > **Authentication** > **oAuth Settings**.

For accessing APIs with a service account, the subsequent sections will help you with instructions to:

- Retrieve Access Token using get-service-token API
- Using Access Token in the header for further API calls

Retrieve Access Token using get-service-token API

The API provides a streamlined process for retrieving service tokens related to account management tasks.

Before you begin

- Make sure you have valid login credentials for accessing the system.

Request Structure

Endpoint:	`/acctmgmt-get-service-token`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/acctmgmt-get-service-token?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json
Authentication:	Yes
Request timeout period	15 minutes

Input Parameters

	Description
Authorization *Header*	(Mandatory) Please form a string in this format <Client ID>:<Client Secret> and do base64 encoding. Then prepend a key 'Basic' before the encoded value. Final value should be "Basic <EncodedValue>". **Type:** *String* **Example:** "admin"
Content-Type *Header*	(Mandatory) The parameter should be set to `application/json` to specify the nature of the data in the payload. **Type:** *String* **Example:** "application/json"
grant_type *Payload*	(Mandatory) Payload Type should be "Form". The value of the param should be "Client_Credentials". **Type:** *Text*

Response Structure

- **Status Code:** 200 Ok
- **Message:** Successful
- **Headers:**
  - **Content-Type:** application/json

Response Parameters

Name	Description
response	The response contains the attributes needed to retrieve the access token.
message	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Status Codes

HTTP Code	appStatusCode	Response Message
200 OK	NA	Successful
400 Bad request	ACCT_SA_003	Service account is invalid/not found::[Service account not found in the database]
400 Bad request	OAUTH_CLNT_015	Client Password is incorrect::[Invalid Client credential]
400 Bad request	ACCT_SA_001	Invalid Request::[Invalid client Id or secret]
500 Internal Server Error	avx-common-011	Error while processing.

Sample Request/Response

Use Case

Retrieve Access Token.

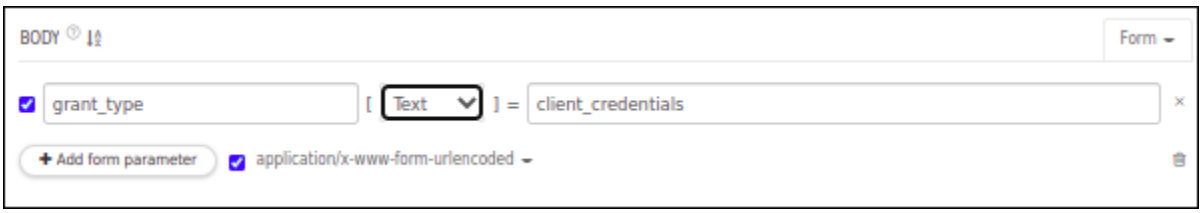
Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/acctmgmt-get-service-token?gwsource=external
```

Content-Type: application/x-www-form-urlencoded

Authorization: Basic
NTIxYzNhZDItZWE0ZS00NDdiLWE1MWItOTYyMWJiN2VhMTI2OjU1QVUjTk84JSpaaGd2TmZhWVVtdHZYMGRrRWhvZ

Request Payload



Sample Response


```
{
"response":
 "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwbGF0Zm9ybSIsImF1ZCI6ImF2eCIsImNsaWVudElkIjoiOTcwNzRlNDEtOGFmOS00NTZkLTlhNjQtZjBj
 NGJiOTA4MDQ4IiwiaXNzIjoiYXZ4IiwiZXhwIjoxNjUwMzY5MzY3LCJncmFudCB0eXBlIjoiY2xpZW50X2NyZWRlbnRpYWxzIn0.HZnkuUEjXIeqJWqpqiNWFHqIDI7
 GYf4cWx6VwbjGD_0",
"message": null,
"appStatusCode": null,
"tags":{},
"headers": null
}
```


Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Using Access Token in the header for further API calls

The access token retrieved using the get-service-token API can be used in the header for making further API calls.

In this section, as an example, we are using the access token with the API call for managing a cluster.

Before you begin

- Access Token is obtained from the get-service-token API.
- Ensure that the Access Token is valid and has not expired.

Request Structure

Endpoint:	`kube-manage-cluster`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-manage-cluster?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameters

Name	Description
Token *Header*	(Mandatory) Use token retrieved from login API. **Type:** *String* **Example:** eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwbGF0Zm9ybSIsImF1ZCI6ImF2eCIsImNsaWVudElkIjoiOTcwNzRlNDEtOGFmOS00NTZkLTlhNjQtZjBjNGJiOTA4MDQ4IiwiaXNzIjoiYXZ4IiwiZXhwIjoxNjUwMzY5MzY3LCJncmFudCB0eXBlIjoiY2xpZW50X2NyZWRlbnRpYWxzIn0.HZnkuUEjXIeqJWqpqiNWFHqIDI7GYf4cWx 6VwbjGD_0
gwsource *Query*	(Mandatory) Source from which the request is triggered. The values can be: • web • external **Type:** *String*
Payload *String*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
clusterName *List<String>*	List of cluster names available in the inventory that needs to be manged.

Response Structure

Name	Description
response *String*	The response contains the unmanage state of the cluster.
message *String*	Success message or failure description in case of error.
appStatusCode	Application specific status code for the response. Will be non-null for failure response.

Name	Description
String	
tags	More info in case of failure response.

Sample Request/Response

Use case: Manage a cluster.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-manage-cluster?gwsource=kube`

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
  "payload": {
    "clusterNames": [
      "casetting-cluster-appviewx-ca-server-ud-gp"
    ]
  }
}
```

Sample Response

```
{
  "response": null,
  "message": "Clusters are Managed",
  "appStatusCode": "success",
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Cluster Inventory

Get Cluster Details

The API to retrieve detailed information about a specific cluster given the cluster name.

Request Structure

Endpoint:	`/kube-get-cluster-details`
Type:	`GET`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-details?name=<cluster_name>&gwsource=<gwsource>`

	To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: `DataCenterA`
clusterName *String*	Name of the cluster.

Response Structure

Name	Description
response *String*	The response contains the cluster details.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Response

Sample Response

```
{
    "response": {
        "clusterID": "e0c40779-55d4-434c-aadc-17c6463ba8e9",
        "clusterName": "cluster-01",
        "certOrchestratorNamespace": "crypto-mesh",
        "keywords": [
            "cluster-01",
            "Self-Managed",
            "crypto-mesh",
            "default",
            "kube-node-lease",
            "kube-system",
            "kube-public",
        ],
        "state": "Managed",
        "status": "Unhealthy",
        "vendor": "Self-Managed",
        "discoveryDefaultGroupName": "Default",
        "createdBy": "admin",
        "createdDateTime": 1716800271978,
        "lastModifiedTime": 1716963662484,
        "lastUpdatedTime": "05/28/2024 10:59:36",
        "namespaces": [
            "crypto-mesh",
            "default",
            "kube-node-lease",
            "kube-system",
            "kube-public"
        ],
        "nodes": [
            {
                "hostName": "cluster-01",
                "type": "master",
                "version": "v1.28.3"
            }
        ],
        "totalNodes": [
```



```
        {
            "hostName": "cluster-01",
            "type": "master",
            "version": "v1.28.3"
        }
    ],
    "numberOfNodes": 1,
    "numberOfPods": 10,
    "appViewXAuthSecretName": "appviewx-auth",
    "appViewXAuthSecretNamespace": "crypto-mesh",
    "credentialType": "Basic Authentication",
    "appViewXURL": "https://{appviewx-fqdn}/"
    "clientId": "",
    "collectionName": "<collection_name>",
    "isDiscoveryEnabled": true,
    "isPrivateKeyDiscoveryRequired": true,
    "isInfraOrchestratorEnabled": true,
    "isAppViewXCSIProviderEnabled": true,
    "isSignerEnabled": true
  },
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```


Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Manage Clusters
- Unmanage Clusters
- Delete a Cluster

Delete a Cluster

The API offboards and deletes a cluster from the Cluster Inventory.

Request Structure

Endpoint:	`/kube-delete-cluster`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-delete-cluster?gwsource=kube` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: `DataCenterA`
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
clusterNames *String*	List of cluster names to offboard.
offboardComment *String*	A message to show in auditLog for offboarding clusters.

Response Structure

Name	Description
response *String*	The response contains the attributes needed for the cluster deletion.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.

Name	Description
tags	More info in case of failure response.

Sample Request/Response

Use case: Offboard a cluster from CA inventory.

Request URL

https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-delete-cluster?gwsource=kube

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
  "clusterNames": [
    "kubeplus"
  ],
  "offboardComment": "Offboarding clusters"
}
```

Sample Response

```
{
    "response": {
        "message": "Successfully triggered workflow",
        "messageType": "SUCCESS"
    },
    "message": null,
    "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.

- **HostName**: A human-readable label assigned to a device (host) on a network

  The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Easy OnBoarding
- Advanced Onboarding with Basic Authentication
- Advanced Onboarding for credential type OAuth

Get Installation Command for OnBoarding and Upgrade

The API generates commands for the following:

- Easy OnBoarding
- Advanced Onboarding with Basic Authentication
- Advanced Onboarding for credential type OAuth

Easy OnBoarding

The API generates cluster installation command for easy onboarding.

Request Structure

Endpoint:	`/kube-get-cluster-installation-command`
Type:	`POST`

Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: A1B2c3d4E5F6
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: DataCenterA
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
isEasyOnBoarding *Boolean*	(Optional) Generates the simple installation command. The values can be: • true • false
isUpgrade *Boolean*	(Optional) Generates upgrade command of cluster. The values can be: • true • false

Input Parameter (continued)

Name	Description
clusterName *String*	(Mandatory) Name of the cluster used in helm install/ upgrade command it is the cert orchestrator name. It can contain alphanumeric characters. Example: `appviewx-cluster`
serviceAccountName *String*	(Optional) When the credential type is OAUTH2.0, the user needs to choose the name of the service account associated with the OAuth 2.0 authentication. It can contain alphanumeric characters.
vendor *String*	(Mandatory) Name of the vendor where the cert orchestrator needs to run. It can contain alphanumeric characters. Example: `Self-Managed`
connectivityType *String*	(Optional) Type of the URL that can be either AppViewX URL or cloud connector machine URL. It can contain alphanumeric characters. Example: `AppViewX URL.`
connectivityURL *String*	(Mandatory) AppViewX node URL, the node through which users can onboard the cluster. It can contain alphanumeric characters. Example: `https://{appviewx-fqdn}`
credentialType *String*	(Mandatory) The type of credentials. The values can be: • Basic Authentication • Oauth
userName *String*	(Optional) Name of the user. If credential type is basic authentication, then username should not be null or blank.
isProvisionCertificatesToK8sSecrets *Boolean*	(Optional) Allows to enroll certificates to secrets. The values can be:

Input Parameter (continued)

Name	Description
	• true • false
isDiscoverCertificates *Boolean*	(Optional) Allows to discover certificates from secret. The values can be: • true • false
isDiscoverK8sInfraCertificates *Boolean*	(Optional) Discovering the infra certificates. The values can be: • true • false
isProvisionCertificatesToEphemeralVolumes *Boolean*	(Optional) Allows to enroll certificates to POD's volumes using appviewx-csi-provider. The values can be: • true • false
isEnableMTLSCertificatesForServiceMesh *Boolean*	(Optional) Allows to enroll certificates to istio service mesh. The values can be: • true • false
isPrivateKeyDiscovery *Boolean*	(Optional) Enables private key discovery. The values can be: • true • false
clusterNamespace *String*	(Mandatory) Installation namespace. The values must not be blank.
discoveryDefaultGroupName *String*	(Optional) Default group name for discovery when `isAutoCreateGroup` is false. The values must not be blank.

Input Parameter (continued)

Name	Description
secretName *String*	(Mandatory) The secret name of AppViewX authentication. The values must not be blank.
secretNamespace *String*	(Mandatory) The secret namespace of AppViewX authentication. The values must not be blank.
isAutoCreateGroup *Boolean*	(Optional) Enables to create the group name automatically for the given cluster/namespace. The values can be: • true • false

Response Structure

- **Status Code:** 202 OK
- **Message:** Successful
- **Headers:**
  - **Content-Type:** application/json

Name	Description
response *String*	The response contains the attributes needed to create namespace, add Helm repo, and install cert-orchestrator plugin for easy onboarding.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Generate cluster installation command for easy onboarding.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command?gwsource=external`

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
   "payload": {
      "isEasyOnBoarding": true,
      "clusterName": "appviewx-cluster",
      "vendor": "Self-Managed",
      "connectivityURL": "https://{appviewx-fqdn}/"
   }
}
```

Sample Response

```
{
 "response": "#Create Namespace for Cert-Orchestrator plugin\n\nkubectl create ns crypto-mesh\n\n#Add Helm Repo\n\nhelm repo add kube-plus-repo
 https://charts.appviewx.com\n\n\n#Install Cert-Orchestrator Plugin\n\nhelm install crypto-mesh kube-plus-repo/crypto-mesh \\\n--namespace crypto-mesh
 \\\n--version v1.3 \\\n--set certOrchestrator.global.clusterName=appviewx-cluster \\\n--set certOrchestrator.global.k8sVendor=Self-Managed",
 "message": "Constructed install command of cert-orchestrator for easyOnboarding",
 "appStatusCode": "success",
 "tags": null,
 "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Generate or Download the Secret YAML

Advanced Onboarding with Basic Authentication

The API generates cluster installation command for advanced onboarding using basic authentication.

Request Structure

Endpoint:	`/kube-get-cluster-installation-command`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource	(Mandatory) Source from which the request is triggered.

Input Parameter (continued)

Name	Description
Header	**Type**: String Example: DataCenterA
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
isEasyOnBoarding *Boolean*	(Optional) Generates the simple installation command. The values can be: • true • false
isUpgrade *Boolean*	(Optional) Generates upgrade command of cluster. The values can be: • true • false
clusterName *String*	(Mandatory) Name of the cluster used in helm install/ upgrade command it is the cert orchestrator name. It can contain alphanumeric characters. Example: appviewx-cluster
vendor *String*	(Mandatory) Name of the vendor where the cert orchestrator needs to run. It can contain alphanumeric characters. Example: Self-Managed
connectivityType *String*	(Optional) Type of the URL that can be either AppViewX URL or cloud connector machine URL. It can contain alphanumeric characters. Example: AppViewX URL.

Input Parameter (continued)

Name	Description
connectivityURL *String*	(Mandatory) AppViewX node URL, the node through which users can onboard the cluster. It can contain alphanumeric characters. Example: https://{appviewx-fqdn}
credentialType *String*	(Mandatory) The type of credentials. The values can be: • Basic Authentication • Oauth
userName *String*	(Optional) Name of the user. If credential type is basic authentication, then username should not be null or blank.
isProvisionCertificatesToK8sSecrets *Boolean*	(Optional) Allows to enroll certificates to secrets. The values can be: • true • false
isDiscoverCertificates *Boolean*	(Optional) Allows to discover certificates from secret. The values can be: • true • false
isDiscoverK8sInfraCertificates *Boolean*	(Optional) Discovering the infra certificates. The values can be: • true • false
isProvisionCertificatesToEphemeralVolumes *Boolean*	(Optional) Allows to enroll certificates to POD's volumes using appviewx-csi-provider. The values can be: • true • false
isEnableMTLSCertificatesForServiceMesh *Boolean*	(Optional) Allows to enroll certificates to istio service mesh. The values can be:

Input Parameter (continued)

Name	Description
	• true • false
isDiscoveryEnabled *Boolean*	(Optional) Enables discovery. The values can be: • true • false
isAutoCreateGroup *Boolean*	(Optional) Enables to create the group name automatically for the given cluster/namespace. The values can be: • true • false
isPrivateKeyDiscovery *Boolean*	(Optional) Enables private key discovery. The values can be: • true • false
clusterNamespace *String*	(Mandatory) Installation namespace. The values must not be blank.
discoveryDefaultGroupName *String*	(Optional) Default group name for discovery when `isAutoCreateGroup` is false. The values must not be blank.
secretName *String*	(Mandatory) The secret name of AppViewX authentication. The values must not be blank.
secretNamespace *String*	(Mandatory) The secret namespace of AppViewX authentication. The values must not be blank.

Response Structure

- **Status Code:** 202 OK
- **Message:** Successful
- **Headers:**
  - **Content-Type:** application/json

Name	Description
response *String*	The response contains the attributes needed to create namespace, add Helm repo, and install cert-orchestrator plugin for onboarding using basic authentication.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Generate cluster installation command for advanced onboarding using basic authentication.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command?gwsource=external`

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
    "payload": {
        "isEasyOnBoarding": false,
        "isUpgrade": false,
        "clusterName": "kubeplus",
        "vendor": "Self-Managed",
        "connectivityType": "AppViewX URL",
        "connectivityURL": "https://{appviewx-fqdn}/",
        "credentialType": "Basic Authentication",
        "userName": "user",
        "isProvisionCertificatesToK8sSecrets": true,
        "isDiscoverCertificates": true,
        "isDiscoverK8sInfraCertificates": true,
        "isProvisionCertificatestoEphemeralVolumes": true,
        "isEnableMTLSCertificatesForServiceMesh": true,
        "isDiscoveryEnabled": true,
        "isAutoCreateGroup": false,
        "isPrivatekeyDiscovery": false,
        "clusterNamespace": "crypto-mesh",
```

```
      "discoveryDefaultGroupName": "default"
   }
}
```


Sample Response


```
{
   "response": "# Create Namespace for Cert-Orchestrator plugin\n\nkubectl create ns crypto-mesh\n\n# Add Helm Repo\n\nhelm repo add
kube-plus-repo https://charts.appviewx.com\n\n# Create credentials to integrate Cert-Orchestrator with AppViewX.\n\nkubectl create secret generic
appviewx-auth -n crypto-mesh --from-literal=APPVIEWX_ENV_USER_NAME=user --from-literal=APPVIEWX_ENV_PASSWORD=[password]
--from-literal=APPVIEWX_ENV_URL=https://{appviewx-fqdn}/\n\n# Install Cert-Orchestrator Plugin\n\nhelm install crypto-mesh
kube-plus-repo/crypto-mesh \\\n--namespace crypto-mesh \\\n--version v1.3 \\\n--set certOrchestrator.global.clusterName=kubeplus
\\\n--set certOrchestrator.global.k8sVendor=Self-Managed \\\n--set certOrchestrator.namespace=crypto-mesh \\\n--set
certOrchestrator.discovery.credentialSecretName=appviewx-auth \\\n--set certOrchestrator.discovery.credentialSecretNamespace=crypto-mesh
\\\n--set appviewxInfraOrchestrator.enabled=True \\\n--set appviewxSigner.enabled=True \\\n--set certOrchestrator.discovery.enabled=True \\\n--set
certOrchestrator.discovery.appviewxGroupName=default",
   "message": "Constructed install command of cert-orchestrator for advanceOnboarding",
   "appStatusCode": "success",
   "tags": null,
   "headers": null
}
```


Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Generate or Download the Secret YAML

Advanced Onboarding for credential type OAuth

The API generates cluster installation command for advanced onboarding using credential type OAuth.

Request Structure

Endpoint:	`/kube-get-cluster-installation-command`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource	(Mandatory) Source from which the request is triggered.

Input Parameter (continued)

Name	Description
Header	**Type**: String Example: DataCenterA
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
isEasyOnBoarding *Boolean*	(Optional) Generates the simple installation command. The values can be: • true • false
isUpgrade *Boolean*	(Optional) Generates upgrade command of cluster. The values can be: • true • false
clusterName *String*	(Mandatory) Name of the cluster used in helm install/ upgrade command it is the cert orchestrator name. It can contain alphanumeric characters. Example: appviewx-cluster
serviceAccountName *String*	(Optional) When the credential type is OAUTH2.0, the user needs to choose the name of the service account associated with the OAuth 2.0 authentication. It can contain alphanumeric characters.
vendor *String*	(Mandatory) Name of the vendor where the cert orchestrator needs to run. It can contain alphanumeric characters. Example: Self-Managed

Input Parameter (continued)

Name	Description
connectivityType *String*	(Optional) Type of the URL that can be either AppViewX URL or cloud connector machine URL. It can contain alphanumeric characters. Example: `AppViewX URL.`
connectivityURL *String*	(Mandatory) AppViewX node URL, the node through which users can onboard the cluster. It can contain alphanumeric characters. Example: `https://{appviewx-fqdn}`
credentialType *String*	(Mandatory) The type of credentials. The values can be: • Basic Authentication • Oauth
userName *String*	(Optional) Name of the user. If credential type is basic authentication, then username should not be null or blank.
isProvisionCertificatesToK8sSecrets *Boolean*	(Optional) Allows to enroll certificates to secrets. The values can be: • true • false
isDiscoverCertificates *Boolean*	(Optional) Allows to discover certificates from secret. The values can be: • true • false
isDiscoverK8sInfraCertificates *Boolean*	(Optional) Discovering the infra certificates. The values can be: • true • false
isProvisionCertificatesToEphemeralVolumes *Boolean*	(Optional) Allows to enroll certificates to POD's volumes using appviewx-csi-provider. The values can be:

Input Parameter (continued)

Name	Description
	• true • false
isEnableMTLSCertificatesForServiceMesh *Boolean*	(Optional) Allows to enroll certificates to istio service mesh. The values can be: • true • false
isDiscoveryEnabled	(Optional) Enables discovery. The values can be: • true • false
isPrivateKeyDiscovery *Boolean*	(Optional) Enables private key discovery. The values can be: • true • false
clusterNamespace *String*	(Mandatory) Installation namespace. The values must not be blank.
discoveryDefaultGroupName *String*	(Optional) Default group name for discovery when `isAutoCreateGroup` is false. The values must not be blank.
isAutoCreateGroup *Boolean*	(Optional) Enables to create the group name automatically for the given cluster/namespace. The values can be: • true • false

Response Structure

Name	Description
response *String*	The response contains the attributes needed to create namespace, add Helm repo, and install cert-orchestrator plugin for onboarding using basic authentication.
message	Success message or failure description in case of error.

Name	Description
String	
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Generate cluster installation command for advanced onboarding using basic authentication.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command?gwsource=external`

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
   "payload": {
      "isEasyOnBoarding": false,
      "isUpgrade": false,
      "clusterName": "kubeplus",
      "serviceAccountName": "service",
      "vendor": "Self-Managed",
      "connectivityType": "AppViewX URL",
      "connectivityURL": "https://{appviewx-fqdn}/",
      "credentialType": "OAuth2.0",
      "isProvisonCertificatesToK8sSecrets": true,
      "isDiscoverCertificates": true,
      "isDiscoverK8sInfraCertificates": true,
      "isProvisionCertificatestoEphemeralVolumes": true,
      "isEnableMTLSCertificatesForServiceMesh": true,
      "isDiscoveryEnabled": true,
      "isAutoCreateGroup": false,
      "isPrivatekeyDiscovery": false,
      "clusterNamespace": "crypto-mesh",
      "discoveryDefaultGroupName": "groupName"
   }
}
```

Sample Response

```
{
   "response": "#Create Namespace for Cert-Orchestrator plugin\n\nkubectl create ns crypto-mesh\n\n#Add Helm Repo\n\nhelm repo
add kube-plus-repo https://charts.appviewx.com\n\n#Create credentials to integrate Cert-Orchestrator with AppViewX.\n\nkubectl
create secret generic appviewx-auth -n crypto-mesh --from-literal=CLIENT_ID=[Client ID] --from-literal=CLIENT_SECRET=[Client
Secret] --from-literal=APPVIEWX_ENV_URL=https://{appviewx-fqdn}/\n\n\n#Install Cert-Orchestrator Plugin\n\nhelm install crypto-mesh
kube-plus-repo/crypto-mesh \\\n--namespace crypto-mesh \\\n--version v1.3 \\\n--set certOrchestrator.global.clusterName=kubeplus
\\\n--set certOrchestrator.global.k8sVendor=Self-Managed \\\n--set certOrchestrator.namespace=crypto-mesh \\\n--set
certOrchestrator.discovery.credentialSecretName=appviewx-auth \\\n--set certOrchestrator.discovery.credentialSecretNamespace=crypto-mesh
\\\n--set appviewxInfraOrchestrator.enabled=True \\\n--set appviewxSigner.enabled=True \\\n--set certOrchestrator.discovery.enabled=True \\\n--set
certOrchestrator.discovery.appviewxGroupName=groupName",
   "message": "Constructed install command of cert-orchestrator for advanceOnboarding",
   "appStatusCode": "success",
   "tags": null,
   "headers": null
}
```


Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Generate or Download the Secret YAML

Generate or Download the Secret YAML

This API is used to generate or download the secret YAML for the following:

- Easy Onboarding
- Service Account Name
- Cluster Name

Easy Onboarding

The API generates command to generate or download the secret YAML for easy onboarding.

Request Structure

Endpoint:	/kube-get-secret-yaml
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-secret-yaml?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: `DataCenterA`
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
isDefaultServiceAccountDownload *Boolean*	(Optional) Downloads the default service account credentials. The values can be: • true • false
authenticationURL *String*	(Mandatory) This URL is referenced in the secret YAML content, and the orchestrator uses this URL to communicate with AppViewX.

Response Structure

Response

Name	Description
name	Name of the secret.
namespace	Namespace in which the secret is created.
type	Type of secret.
data	Encoded data fields containing sensitive information such as client ID, client secret, and AppViewX environment URL.

Name	Description
response *String*	The response contains the attributes needed to create namespace, add Helm repo, and install cert-orchestrator plugin for easy onboarding.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Generate command to generate or download the secret YAML for easy onboarding.

Request URL

https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-secret-yaml?gwsource=external

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
   "payload": {
      "isDefaultServiceAccountDownload": true,
      "authenticationURL": "url"
   }
}
```

Sample Response

```
---
apiVersion: "v1"
kind: "Secret"
metadata:
name: "appviewx-auth"
namespace: "crypto-mesh"
type: "Opaque"
data:
CLIENT_ID: "ZTE0ZmRlOGMtMTg0ZC00ZmRkLTk1MTItNGQzYzZmYjI3NmQz"
CLIENT_SECRET: "ZVE2clB1QkdIOCpnTTFWUmMzeVZ2VHZQbUNwI3pBYzY="
APPVIEWX_ENV_URL: "dXJs"
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Upgrade Cluster Command

Service Account Name

The API generates command to generate or download the secret YAML using service account name.

Request Structure

Endpoint:	/kube-get-secret-yaml
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-secret-yaml?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters.

Input Parameter (continued)

Name	Description
	Type: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: `DataCenterA`
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
isDefaultServiceAccountDownload *Boolean*	(Optional) Downloads the default service account credentials. The values can be: • true • false
serviceAccountName *String*	(Optional) If isDefaultServiceAccountDownload is set to false, the specified ServiceAccountName is used for downloading the service account credentials.
authenticationURL *String*	(Mandatory) This URL is referenced in the secret YAML content, and the orchestrator uses this URL to communicate with AppViewX.

Response Structure

Response

Name	Description
name	Name of the secret.
namespace	Namespace in which the secret is created.

Response (continued)

Name	Description
type	Type of secret.
data	Encoded data fields containing sensitive information such as client ID, client secret, and AppViewX environment URL.

Name	Description
response *String*	The response contains the attributes needed to create namespace, add Helm repo, and install cert-orchestrator plugin for easy onboarding.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Generate command to generate or download the secret YAML using service account name.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-secret-yaml?gwsource=external`

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
   "payload": {
      "isDefaultServiceAccountDownload": false,
      "serviceAccountName": "kube-svc-account",
      "authenticationURL": "url"
   }
}
```

Sample Response

```
---
apiVersion: "v1"
kind: "Secret"
```

```
metadata:

name: "appviewx-auth"

namespace: "crypto-mesh"

type: "Opaque"

data:

CLIENT_ID: "ZTE0ZmRlOGMtMTg0ZC00ZmRkLTk1MTItNGQzYzZmYjI3NmQz"

CLIENT_SECRET: "ZVE2clB1QkdIOCpnTTFWUmMzeVZ2VHZQbUNwI3pBYzY="

APPVIEWX_ENV_URL: "dXJs"
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Upgrade Cluster Command

Cluster Name

The API generates command to generate or download the secret YAML using cluster name.

Request Structure

Endpoint:	/kube-get-secret-yaml
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-secret-yaml?gwsource=external To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: A1B2c3d4E5F6
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: DataCenterA
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
isDefaultServiceAccountDownload *Boolean*	(Optional) Downloads the default service account credentials. The values can be:

Input Parameter (continued)

Name	Description
	• true • false
clusterName *String*	(Optional) Retrieves or downloads the secret using the data from cluster information. Any string matches regex "^[A-Za-z0-9-_.*|!\\s][A-Za-z0-9-_.* |!]*$\|^$"
authenticationURL *String*	(Mandatory) This URL is referenced in the secret YAML content, and the orchestrator uses this URL to communicate with AppViewX.

Response Structure

Response

Name	Description
name	Name of the secret.
namespace	Namespace in which the secret is created.
type	Type of secret.
data	Encoded data fields containing sensitive information such as client ID, client secret, and AppViewX environment URL.

Name	Description
response *String*	The response contains the attributes needed to create namespace, add Helm repo, and install cert-orchestrator plugin for easy onboarding.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Generate command to generate or download the secret YAML using cluster name.

Request URL

```
https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-secret-yaml?gwsource=external
```

Sample Request

```
Content type: application/json Username: <> Password: <>
{
   "payload": {
      "isDefaultServiceAccountDownload": false,
      "serviceAccountName": "kube-svc-account",
      "authenticationURL": "url"
   }
}
```

Sample Response

```
---
apiVersion: "v1"
kind: "Secret"
metadata:
name: "appviewx-auth"
namespace: "crypto-mesh"
type: "Opaque"
data:
CLIENT_ID: "ZTE0ZmRlOGMtMTg0ZC00ZmRkLTk1MTItNGQzYzZmYjI3NmQz"
CLIENT_SECRET: "ZVE2clB1QkdIOCpnTTFWUmMzeVZ2VHZQbUNwI3pBYzY="
APPVIEWX_ENV_URL: "dXJs"
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.

- **HostName**: A human-readable label assigned to a device (host) on a network

  The hostname will be included in the endpoint URL for an on-prem deployment.

- **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

  The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Upgrade Cluster Command

Upgrade Cluster Command

This API used to upgrade cluster command.

Request Structure

Endpoint:	/kube-get-cluster-installation-command
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command?gwsource=kube To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: `DataCenterA`
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Payload Details

Name	Description
hook *Object*	Contains all the hook params.
input *Object*	Tenant Key

Hook Details

Name	Description
name *String*	(Mandatory) Name of the hook. **Possible values:** Cluster

Input Details

Name	Description
action	(Mandatory) Specific action.

Input Details (continued)

Name	Description
String	**Possible values:** upgrade_commands
payload *payload*	Contains all the payload params.

Payload Parameter

Name	Description
clusterName *String*	(Mandatory) Name of the Cluster.
purpose *String*	(Mandatory) Features to be enabled in the cert-orchestrator deployed in the cluster. Possible Values: • Provision Certificates to K8s Secrets • Discover Certificates • Provision Certificates to Ephemeral Volumes • Discover K8's Infra Certificates • Enable mTLS Certificates for Service Mesh

Response Structure

Name	Description
response *String*	The response contains the upgrade status.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Upgrade command for a cluster.

Request URL

https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-get-cluster-installation-command ?gwsource=kube

Sample Request

POST Content type: application/json Username: <> Password: <>

```
{
    "payload": {
        "hook": {
            "name": "Cluster"
        },
        "input": {
            "action": "upgrade_commands",
            "payload": {
                "clusterName": "testing.appviewx.net",
                "purpose": [
                    "Provision Certificates to K8s Secrets",
                    "Discover Certificates",
                    "Provision Certificates to Ephemeral Volumes"
                ]
            }
        }
    }
}
```

Sample Response

```
{
  "response": {
    "status": "Success",
    "output": {
      "status": "Success",
      "response": "\"Update Command\"\nhelm upgrade crypto-mesh crypto-mesh/crypto-mesh --version v1.1 --n"
    },
    "type": "script"
  },
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Cluster Inventory

Manage Clusters

The API manages the cluster in inventory, which are in pending approval state.

- Manage Clusters
- Unmanage Clusters

Manage Clusters

The API will manage the cluster in inventory which are in pending approval state.

Request Structure

Endpoint:	/kube-manage-cluster
Type:	POST
Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-manage-cluster?gwsource=kube To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: A1B2c3d4E5F6
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: DataCenterA
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
clusterName *List<String>*	List of cluster names available in the inventory that needs to be manged.

Response Structure

Name	Description
response *String*	The response contains the unmanage state of the cluster.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Manage a cluster.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-manage-cluster?gwsource=kube`

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
  "payload": {
    "clusterNames": [
      "casetting-cluster-appviewx-ca-server-ud-gp"
    ]
  }
}
```

Sample Response

```
{
  "response": null,
  "message": "Clusters are Managed",
  "appStatusCode": "success",
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Unmanage Clusters

Unmanage Clusters

The API will unmanage the cluster in inventory which are in pending approval state.

Request Structure

Endpoint:	`/kube-unmanage-cluster`
Type:	`POST`

Sample URL:	https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-unmanage-cluster?gwsource=kube To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: A1B2c3d4E5F6
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: DataCenterA
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
clusterName *List<String>*	List of cluster names available in the inventory that needs to be unmanged.

Response Structure

Name	Description
response *String*	The response contains the unmanage state of the cluster.
message	Success message or failure description in case of error.

Name	Description
String	
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Unmanage a cluster.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-unmanage-cluster?gwsource=kube`

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
  "payload": {
    "clusterNames": [
      "casetting-cluster-appviewx-ca-server-ud-gp"
    ]
  }
}
```

Sample Response

```
{
  "response": null,
  "message": "Clusters are Unmanaged",
  "appStatusCode": "success",
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Manage Clusters

Certificate Actions

- CaSetting Action (Push/Revoke)
- Certificate Action (Push/Revoke)

CaSetting Action (Push/Revoke)

The APIs for push and revoke the Issuer CA from AppViewX to end cluster are:

CA Setting Push

The API pushes the issuer CA from AppViewX to end cluster.

Request Structure

Endpoint:	`/kube-cluster-casetting-push`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-cluster-casetting-push?gwsource=kube` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: `DataCenterA`
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
clusterName *String*	Name of the cluster associated with the CA Setting template.
kubeCaSettingName *String*	Name of the CA Setting template created by user.
selectedNamespaces *String*	Namespace associated to the CA Setting template **Note:** Namespace value will be blank “” if the CA Setting template has Cluster Wide Policy. Otherwise actual namespace value.

Response Structure

Name	Description
response *String*	The response contains the attributes needed for the CA settings push.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Push or revoke the CA from AppViewX to end cluster.

Request URL

`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-cluster-casetting-push?gwsource=kube`

Sample Request

```
POST Content type: application/json Username: <> Password: <>

{

   "payload": [
```

```
    {
      "clusterName": "poc-n4",
      "kubeCaSettingName": "test-ca-setting-cluster",
      "selectedNamespaces": ""
    }
  ]
}
```

Sample Response

```
{
  "response": {
    "messageType": "SUCCESS",
    "message": "Issuer CA deployed to cluster successfully",
    "status": "Success"
  },
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.

- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**

- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- CA Setting Revoke

CA Setting Revoke

The API revokes the issuer CA from AppViewX to end cluster.

Request Structure

Endpoint:	`/kube-cluster-casetting-revoke`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-cluster-casetting-revoke?gwsource=kube` To understand the elements of the sample URL, click here.
Headers:	
Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource	(Mandatory) Source from which the request is triggered.

Input Parameter (continued)

Name	Description
Header	**Type**: String Example: DataCenterA
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Input Parameter

Name	Description
clusterName *String*	Name of the cluster associated with the CA Setting template.
kubeCaSettingName *String*	Name of the CA Setting template created by user.
selectedNamespaces *String*	Namespace associated to the CA Setting template **Note:** Namespace value will be blank “” if the CA Setting template has Cluster Wide Policy. Otherwise actual namespace value.

Response Structure

Name	Description
response *String*	The response contains the attributes needed for the CA settings revoke.
message *String*	Success message or failure description in case of error.
appStatusCode *String*	Application specific status code for the response. Will be non-null for failure response.
tags	More info in case of failure response.

Sample Request/Response

Use case: Push or revoke the CA from AppViewX to end cluster.

Request URL

https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-cluster-casetting-revoke?gwsource=kube

Sample Request

```
POST Content type: application/json Username: <> Password: <>
{
   "payload": [
      {
         "clusterName": "poc-n4",
         "kubeCaSettingName": "test-ca-setting",
         "selectedNamespaces": "absecon"
      }
   ]
}
```

Sample Response

```
{
   "response": {
      "messageType": "SUCCESS",
      "message": "Issuer CA revoked from cluster successfully",
      "status": "Success"
   },
   "message": null,
   "appStatusCode": null,
   "tags": null,
   "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- CA Setting Push

Certificate Action (Push/Revoke)

The API will push or revoke the Certificate from AppViewX to end cluster.

Request Structure

Endpoint:	`/kube-cert-action`
Type:	`POST`
Sample URL:	`https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-cert-action?gwsource=external` To understand the elements of the sample URL, click here.
Headers:	

Content-Type:	application/json

Input Parameter

Name	Description
sessionId *Header*	(Mandatory) A unique identifier assigned to a user's session upon successful authentication. The session ID remains valid until it expires, and it can contain alphanumeric characters. **Type**: String **Constraints**: The session ID is used when username and password are not provided. Example: `A1B2c3d4E5F6`
gwsource *Header*	(Mandatory) Source from which the request is triggered. **Type**: String Example: `DataCenterA`
payload *Body*	(Mandatory) Input data for request body in application/json format. For payload details, see Payload section.

Payload

Name	Description
action *String*	Certificate action to be performed within the cluster. **Constraints**: The actions are push and delete the certificate in cluster
certName *String*	Name of the certificate.
caSettingType *String*	CA type of the issuer CA. **Constraints**: The values are CA Setting Cluster and CA Setting.
caSettingName *String*	Name of the issuer CA.
clusterName *String*	Name of cluster in which the certificate-related action needs to be performed.
namespace *String*	Namespace where the CA setting must be pushed.

Sample Request/Response

Use case: Revoke the certificate from AppViewX to end cluster.

Request URL

https://<IP/HostName/TenantName>:<GWPORT>/avxapi/kube-cert-action?gwsource=kube

Sample Request

```
{
  "payload": {
    "certs": [
      {
        "certName": "cert-default-casetting-default-selfsigned",
        "caSettingType": "CA Setting",
        "caSettingName": "casetting-default-selfsigned",
        "clusterName": "kubeplus-05",
        "namespace": "default"
      }
    ],
    "action": "delete"
  }
}
```

Sample Response

```
{
  "response": {
    "messageType": "SUCCESS",
    "message": "Certificate revoked from secret successfully",
    "status": "Success"
  },
  "message": null,
  "appStatusCode": null,
  "tags": null,
  "headers": null
}
```

Reference

Understanding the sample URL: This section provides an explanation of each component of the sample URL structure used in API requests. For quick reference, this section is referenced in all the API topics as **Reference** in this guide.

- **IP/HostName/TenantName**: Replace with the actual IP address, hostname, or tenant name based on the specific configuration in AppViewX.
  - **IP**: A unique identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication

    The IP address will be included in the endpoint URL for an on-prem deployment.
  - **HostName**: A human-readable label assigned to a device (host) on a network

    The hostname will be included in the endpoint URL for an on-prem deployment.
  - **TenantName**: An identifier label for a tenant given to indicate which tenant's data the API request will access/modify

    The tenant name will be included in the endpoint URL for a SaaS deployment.
- **GWPORT**: AppViewX gateway port

  A gateway port refers to a network port through which data is sent and received to communicate with a gateway in an on-prem deployment.

  Example: **31443**
- **avxapi**: Path parameter value (static) that is part of the endpoint's URL
- **Endpoint**: Endpoint of the API, for example: **execute-hook**
- **gwsource**: Source or origin of a gateway, for example: **external**.

What's New

- Upgrade Cluster Command

Chapter 3: KUBE+ Upgrade Guide

Introduction

When migrating the AppViewX Compute cluster, it is necessary to upgrade the Cloud Connector and Cert Orchestrator to enable the latest KUBE+ features.

Upgrade the Cloud Connector by selecting the upgrade option on the Cloud Connector page.

Due to changes in the API that Cert Orchestrator uses to communicate with AppViewX, Cert Orchestrator should also be upgraded after upgrading the compute cluster.

Steps for Cert-Orchestrator Migration from v1.2 to v1.3

Here are the steps for upgrading AppViewX KUBE+ to leverage the enhanced features introduced in the AppViewX 2023.1.0 FPs release:

1. Update the crypto-mesh helm repo by executing the command `helm repo update <REPO_NAME>`.
2. Get v1.3 helm chart by executing the command `helm pull <REPO_NAME>/crypto-mesh --version v1.3`.

   Note: This command will download the helm chart for crypto-mesh v1.3 and save it as **`crypto-mesh-v1.3.tgz`** in the current folder.

3. Extract the contents of the **`crypto-mesh-v1.3.tgz`** file by executing the command `tar -xvf crypto-mesh-v1.3.tgz`.

   Note: This command will extract the contents of the tarball (**`tgz`** file) and create a folder named 'crypto-mesh' with the extracted files in the current directory.

4. Apply CRDs of v1.3 from the above folder by executing the command `kubectl apply -f ./crypto-mesh/crds/cert-orchestrator/`
5. Upgrade cert-orchestrator from v1.2 to v1.3 using "helm upgrade" by executing the following command:

   `helm upgrade <HELM_NAME> <REPO_NAME>/crypto-mesh --version v1.3 --namespace`

```
<NAMESPACE> --set
    certOrchestrator.discovery.credentialSecretName=appviewx-auth --set
    certOrchestrator.discovery.credentialSecretNamespace=<NAMESPACE> --set
    certOrchestrator.discovery.isGroupAutoGenerate=<true/false> --set
    certOrchestrator.global.clusterName=<CLUSTER_NAME> --set
    certOrchestrator.global.k8sVendor=<VENDOR>
```

Make sure to replace the placeholders with the actual values:

a. <HELM_NAME>: Existing Helm name for cert-orchestrator. To get the existing helm name, execute the command `helm ls -A | grep crypto-mesh-v1.2`.

b. <REPO_NAME>: Name of the Helm repository.

c. <NAMESPACE>: Namespace where cert-orchestrator is installed. To get the existing namespace, execute the command `helm ls -A | grep crypto-mesh-v1.2`.

.

d. <CLUSTER_NAME>: Existing cluster name. To get the existing cluster name, execute the command `kubectl get deployment cert-orchestrator -n <NAMESPACE> -o yaml | grep cluster-name`

e. <VENDOR>: Vendor type (Self-Managed/EKS/AKS/GKE/OpenShift).

Chapter 4:KUBE+ as an EKS Addon - AWS Marketplace

AppViewX KUBE+ for Workload Certificate Management: An AWS EKS Add-On

Amazon Elastic Kubernetes Service (EKS) is a managed container service to run and scale Kubernetes applications in the Amazon Web services (AWS) cloud. AppViewX provides an EKS Add-on available in the AWS marketplace that allows you to seamlessly deploy the KUBE+ solution to Amazon EKS clusters to efficiently manage certificate management for your container workloads.

For more information, read the official AWS documentation at Amazon EKS add-ons.

Benefits

The AppViewX KUBE+ Amazon EKS Add-on:

1. Provides simplified installation, configuration and management of x509 certificates on the Amazon EKS clusters.
2. Includes the latest security patches, bug fixes, and are validated by AWS to work with Amazon EKS.

Install the EKS Add-on

To install the EKS Add-on and manage the x509 certificates follow these steps:

- Step 1: Subscribe to the EKS Add-on in the AWS Marketplace
- Step 2: Ensure access to a valid AppViewX KUBE+ Subscription
- Step 3: Connect your EKS Cluster to AppViewX

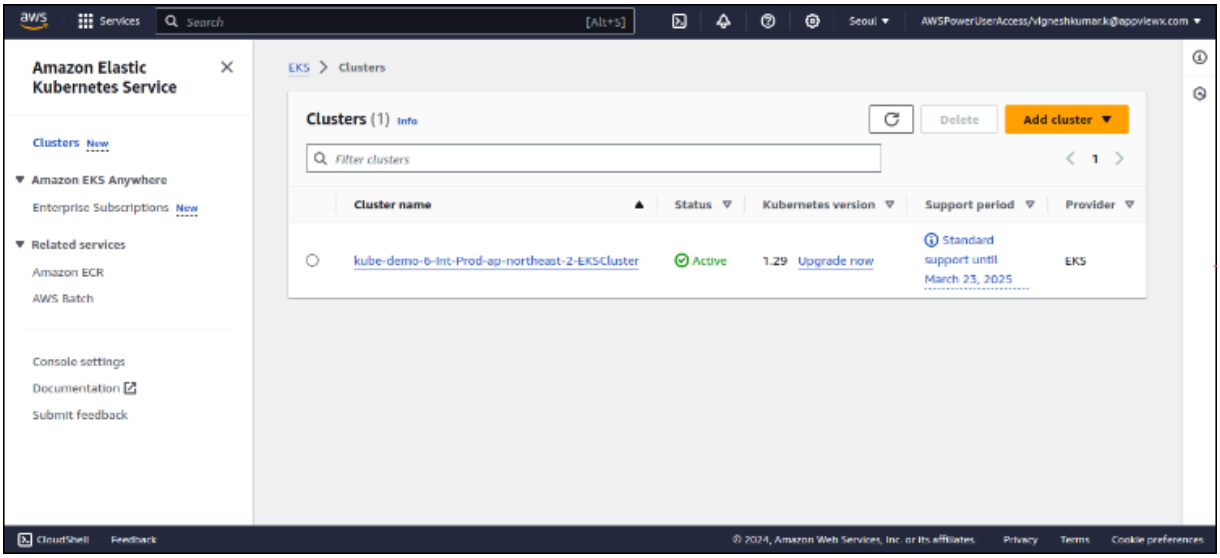
Step 1: Subscribe to the EKS Add-on in the AWS Marketplace

In your AWS Marketplace, ensure the following:

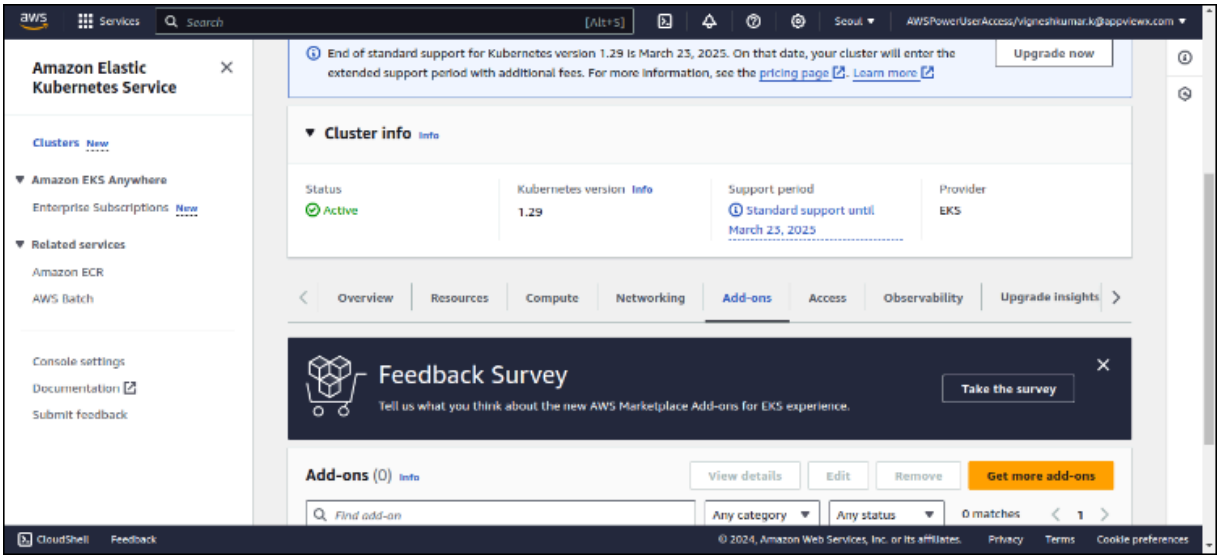
- You have sufficient permissions in your AWS account to enable this Add-on.
- Complete the subscription process in the AWS console. Go to the AWS Marketplace Page to add the AppViewX KUBE+ - EKS Add-on to your AWS account.

Below steps will navigate you through the add-on subscription process.

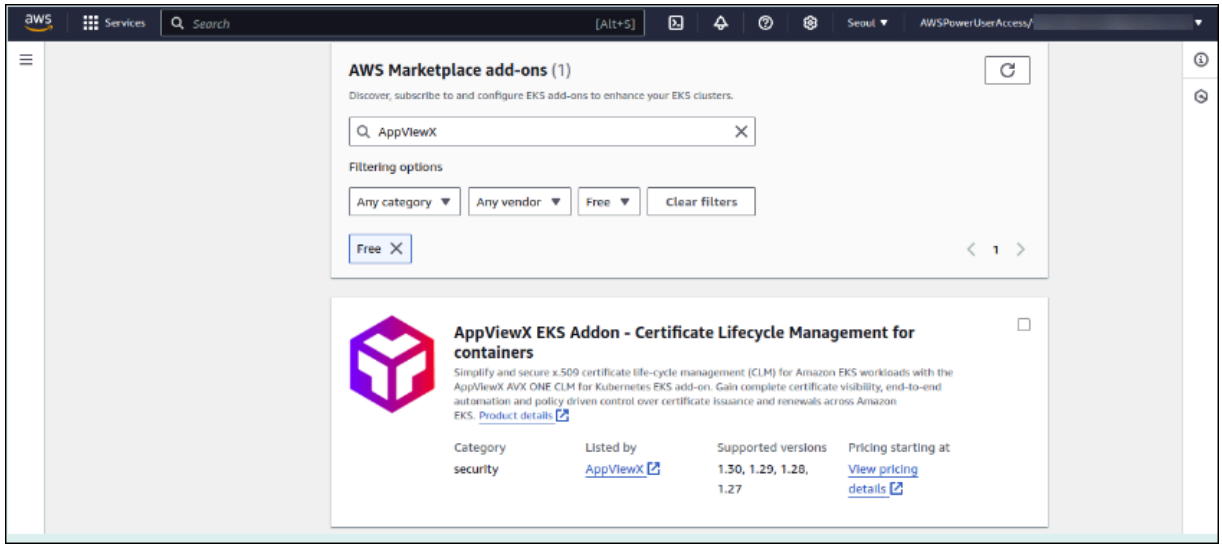
1. Accessing the EKS console - In your AWS account navigate to the **EKS Console** > **Clusters** and select the cluster where the AppViewX add-on is to be deployed.



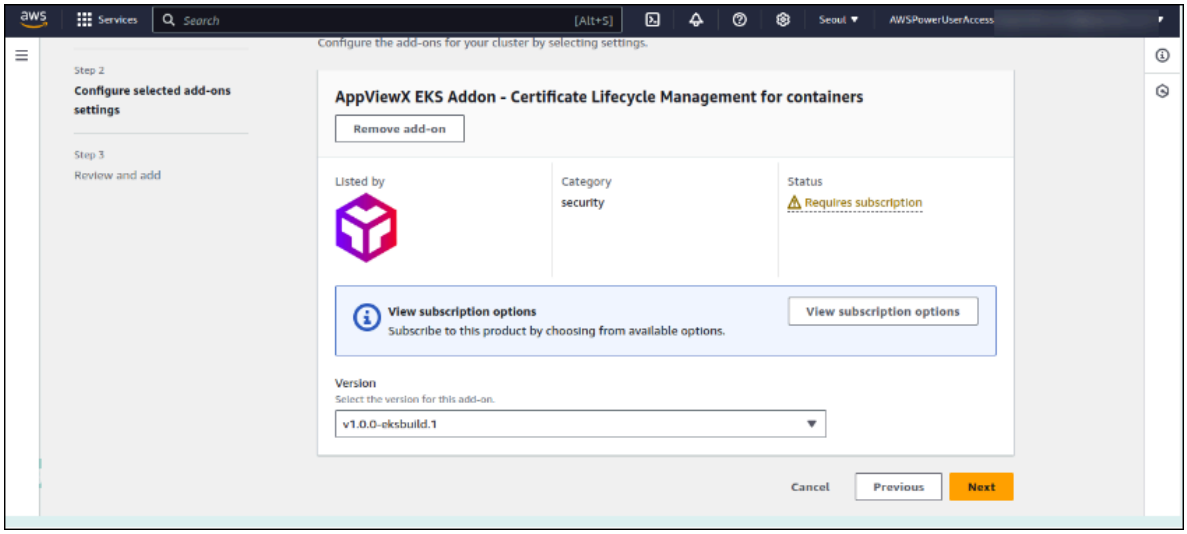
2. Subscribing for Marketplace Add-on - In the **Cluster info** section, navigate to the **Add-ons** tab and click the **Get more add-ons** button.



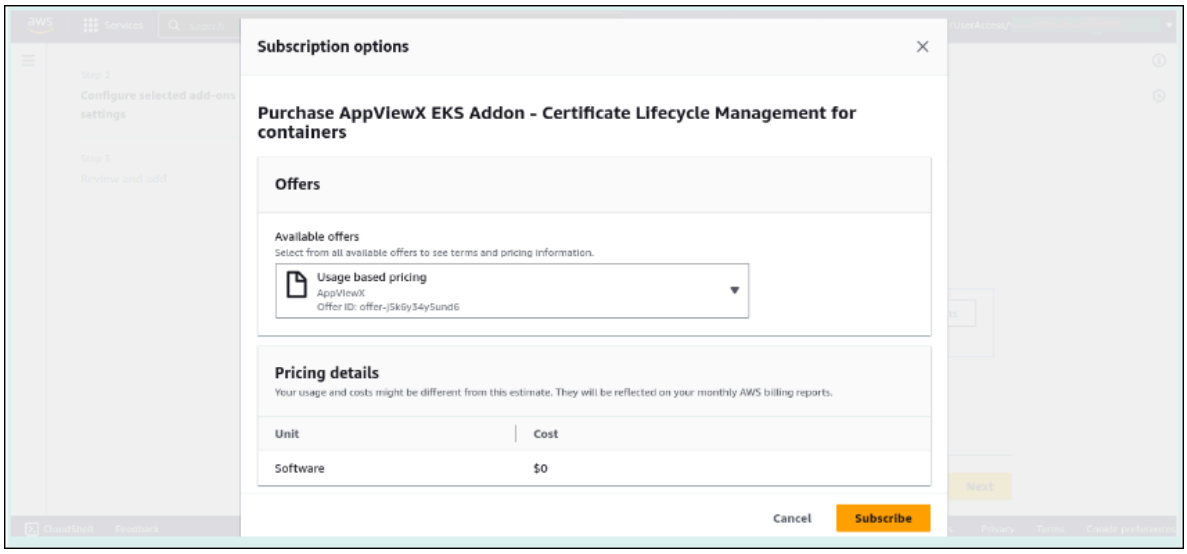
3. In the add-ons section, under the **AWS Marketplace add-ons**, do the following steps:
   a. Search for AppViewX.
   b. Select the **AppViewX EKS Addon - Certificate Lifecycle Management for containers** checkbox.
   c. Click **Next**.



4. Choosing the Add-on and making the marketplace subscription:
   a. Select the add-on version click on **View Subscription**.

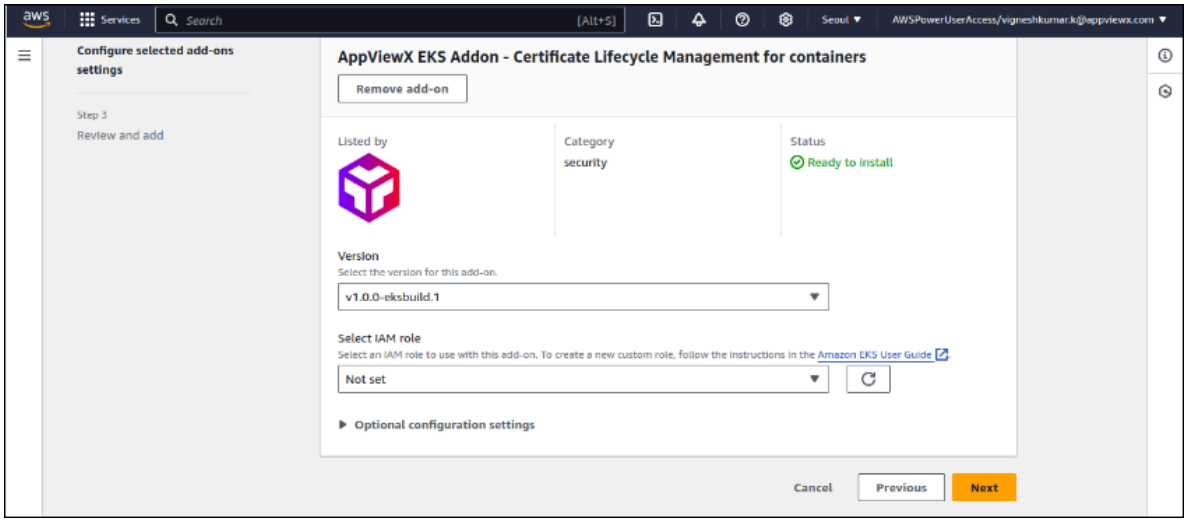


   b. Verify the subscription options, and then click **Subscribe**.

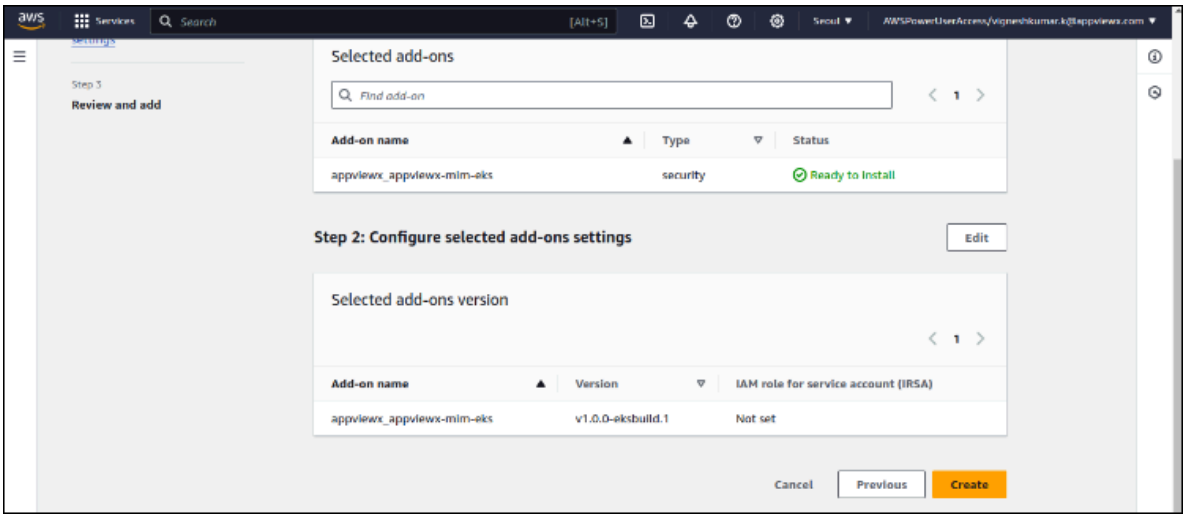


5. Continuing with add-on configuration & installation

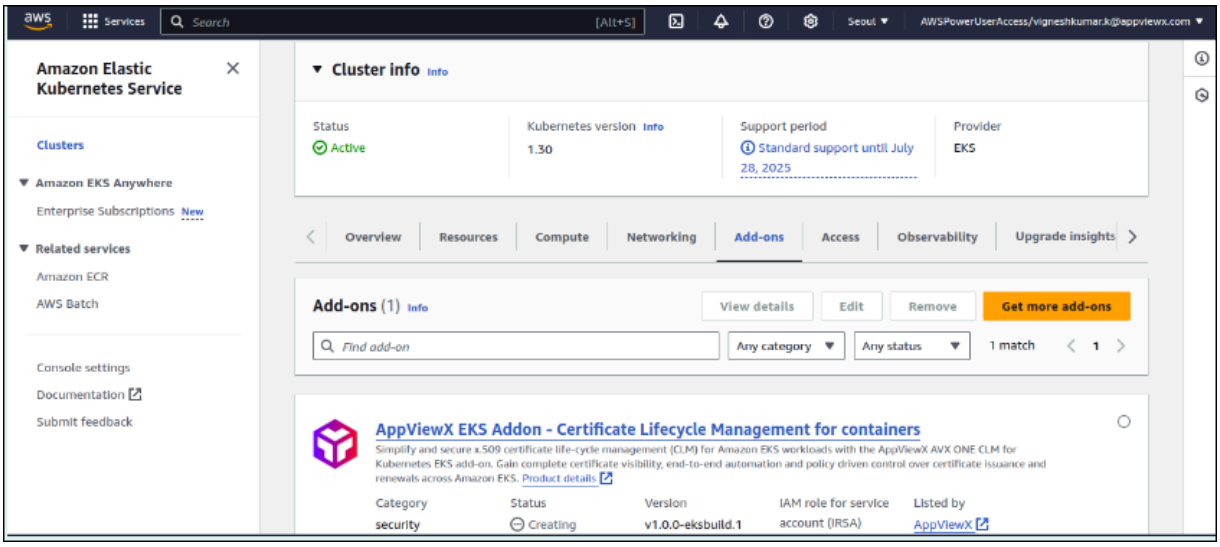
   a. Once the subscription status is **Ready to Install** continue select the version and click **Next**.



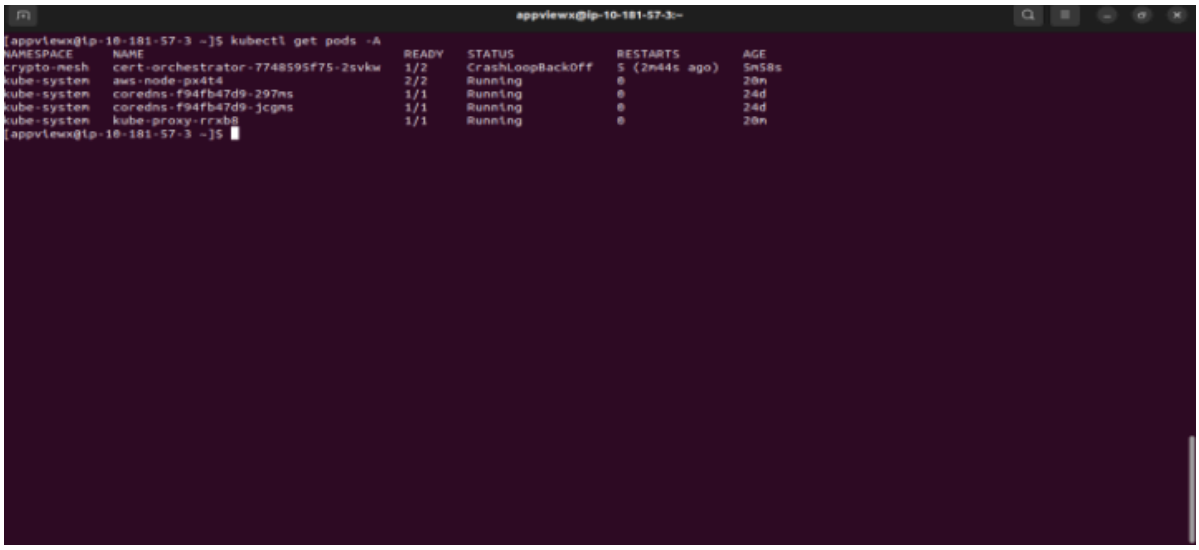
   b. In the next page, review the subscription and the add-on configuration, and then click **Create**.



6. Verifying the add-on installation at the cluster end - After selecting the **Create** action, the KUBE+ add-on is automatically deployed to the clusters, and the status on the cluster add-on page updates to **Creating**.



7. To verify the add-on pod running status, log in to the cluster and verify the install.
   - Log in to bastion / cluster and with necessary access permissions, execute the kubectl commands to verify the pod status.
   - To verify if the pod is running, execute the command `kubectl get pods -n crypto-mesh`.



Note: The result of the command provided above is expected to have a **CrashLoopBackoff** status.

The administrator/user should download the credentials from a valid AppViewX Subscription to connect the add-on to an AppViewX KUBE+ environment.

Step 2: Ensure access to a valid AppViewX KUBE+ Subscription

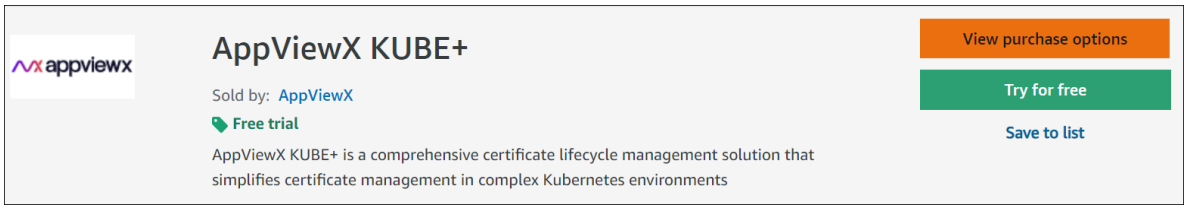
For users evaluating the AppViewX KUBE+ solution as a Service (SaaS), which enables turnkey Certificate Lifecycle Management on the container workloads, AppViewX enables multiple channels to onboard you for a free trial / paid subscription of the product:

- via the AWS Marketplace
- via request to AppViewX
- Alternatively existing AppViewX customers ensure to have KUBE+ license subscription

1. Via AWS Marketplace

   a. Navigate to the Marketplace.

      The **AppViewX KUBE+** page is displayed.



2. Sign into your account or create your account if you are new to AWS Marketplace.
3. Click **Continue to Subscribe**.

> **Note:** The marketplace subscription steps for KUBE+ product of AppViewX are provided here.

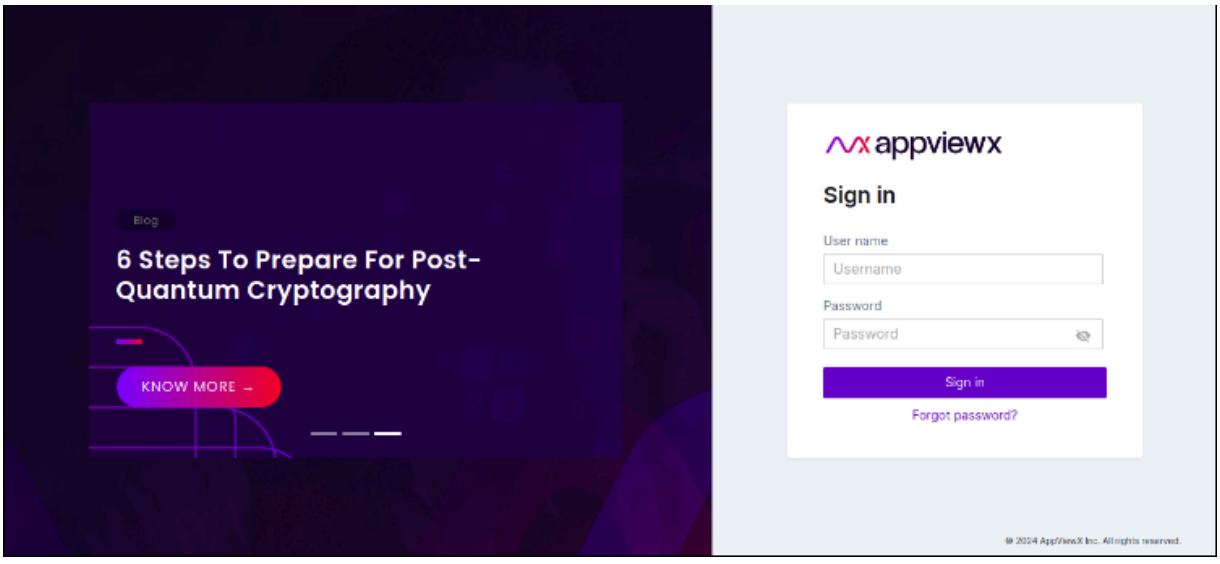
4. Via request to AppViewX - For a SaaS instance from appviewx for free trial or paid subscription reach sales@appviewx.com

Step 3: Connect your EKS cluster to AppViewX

To connect EKS cluster to AppViewX, ensure the following:

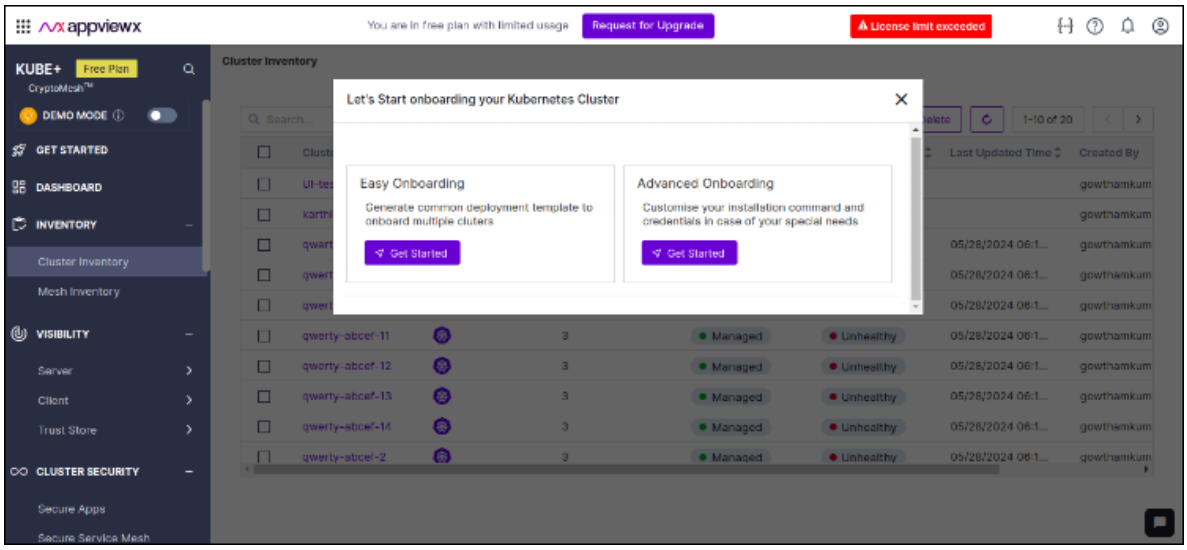
- You have access to the AppViewX (available as) SaaS, Onprem or Managed K8s based deployments.
- You have sufficient Permission to access the KUBE+ feature sets.

1. Accessing the AppViewX console - Access your AppViewX tenant/deployment with the login credentials. Go to **Menu** > **KUBE+**.

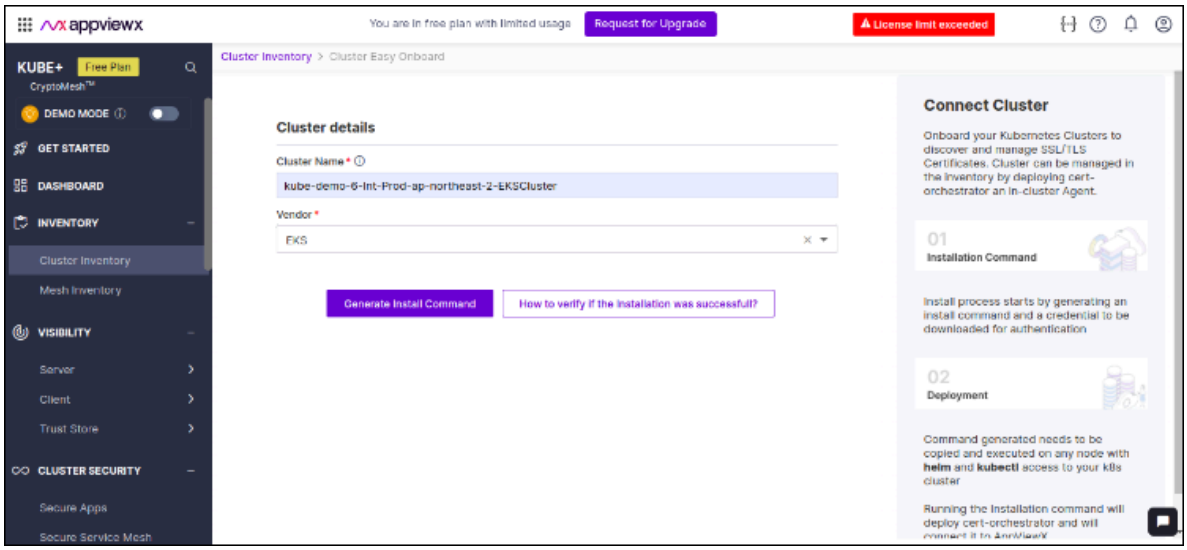


2. Generate Credentials or Access Token for connectivity:

a. Generate authentication credentials for connecting the EKS add-on to AppViewX. Go to **Menu** > **KUBE+** > **Cluster Inventory** > **Connect Cluster** > **Onboard** > **Easy Onboarding**.



b. Provide the cluster name and select the vendor and click **Generate Install Command**.



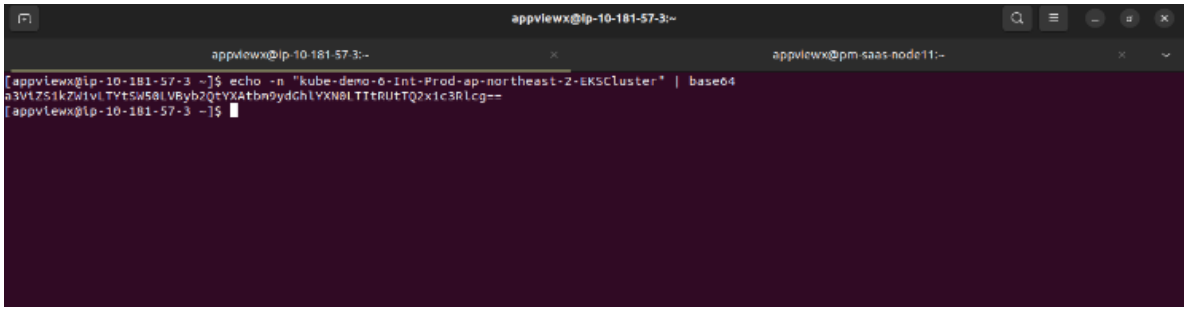
c. Download the authentication credentials as YAML by clicking the **Download Credentials** and exit back to the inventory.

3. Deploy credentials in the EKS cluster:

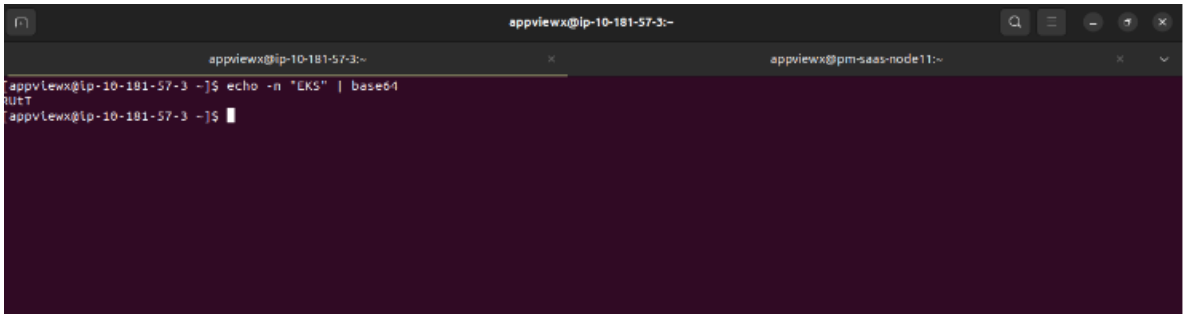
a. Copy the YAML to the Bastion host or to the cluster for deploying the credentials in the cluster. Before deployment of credentials ensure the **Cluster Name** and **Vendor Name** is added to the credential YAML.

b. Add **Cluster Name** and **Vendor Name** to the YAML converting the Cluster Name and Vendor Name as a base64 encoded text.

c. Encoding the cluster name.

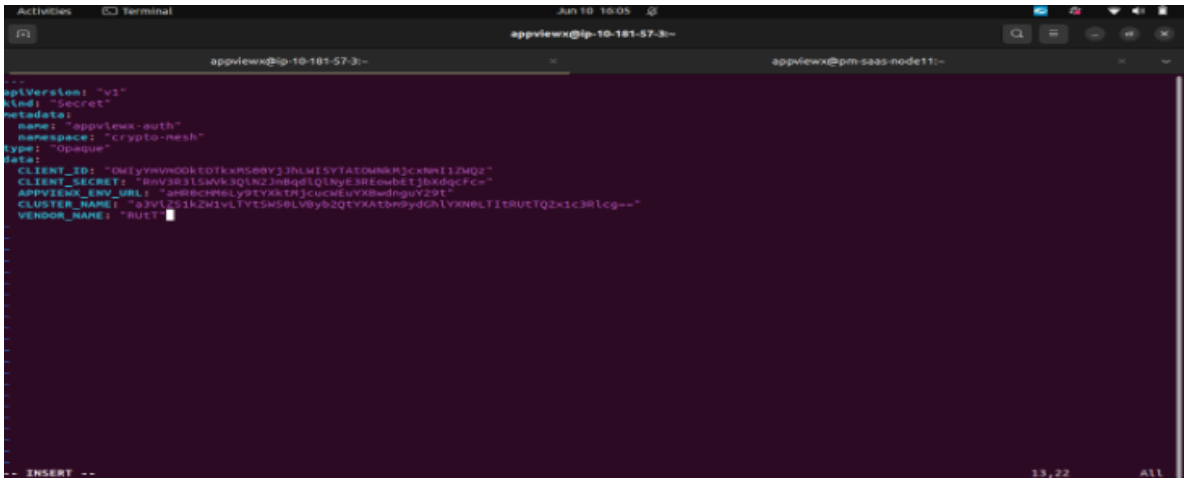


d. Encoding the vendor name.



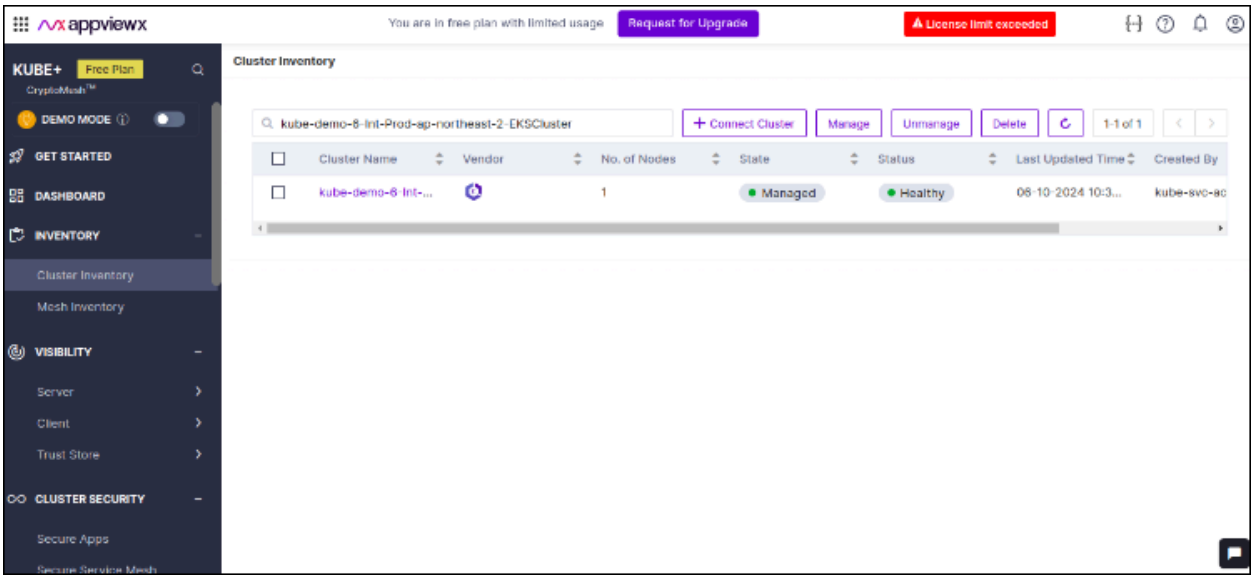
e. Adding the above to the YAML:

- Edit the secret.yaml downloaded to the local machine, cluster, or bastion.
- Add the **Cluster Name** and **Vendor Name** from the above derived output.
- Save the YAML file.



Step 4: Deploy credentials and manage EKS cluster in AppViewX

Deploy the above saved YAML to the cluster with the command `kubectl apply -f secret.yaml` and once the connection is established the cluster is automatically managed in AppViewX KUBE+ Cluster Inventory.



Manage SSL/TLS certificates in your EKS

With the above mentioned steps you should be able to successfully deploy the AppViewX KUBE+ EKS addon and manage the cluster in the AppViewX Cluster Inventory.

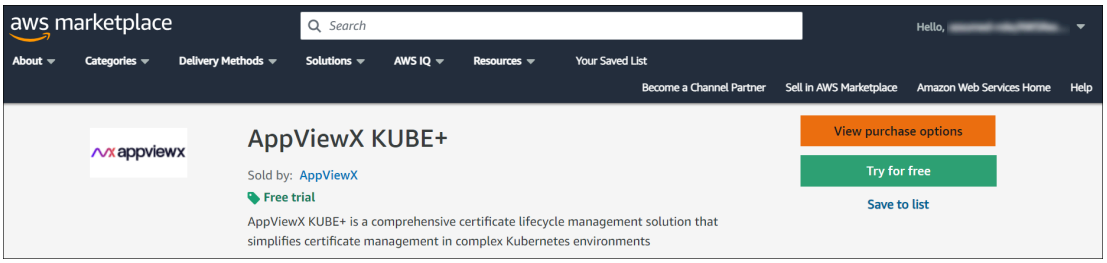
Once the cluster is managed, you can refer to the AppViewX KUBE+ documentation on how to use KUBE+ for certificate management operations (discover, monitor and enroll certificates) for your EKS cluster.

Signing Up for the Free Trial via the AWS Marketplace

To get started with the KUBE+ SaaS free trial, you can sign up via the AWS Marketplace and set up the SaaS by following the steps given below:

Step 1: Accessing the AWS Marketplace Sign Up Page

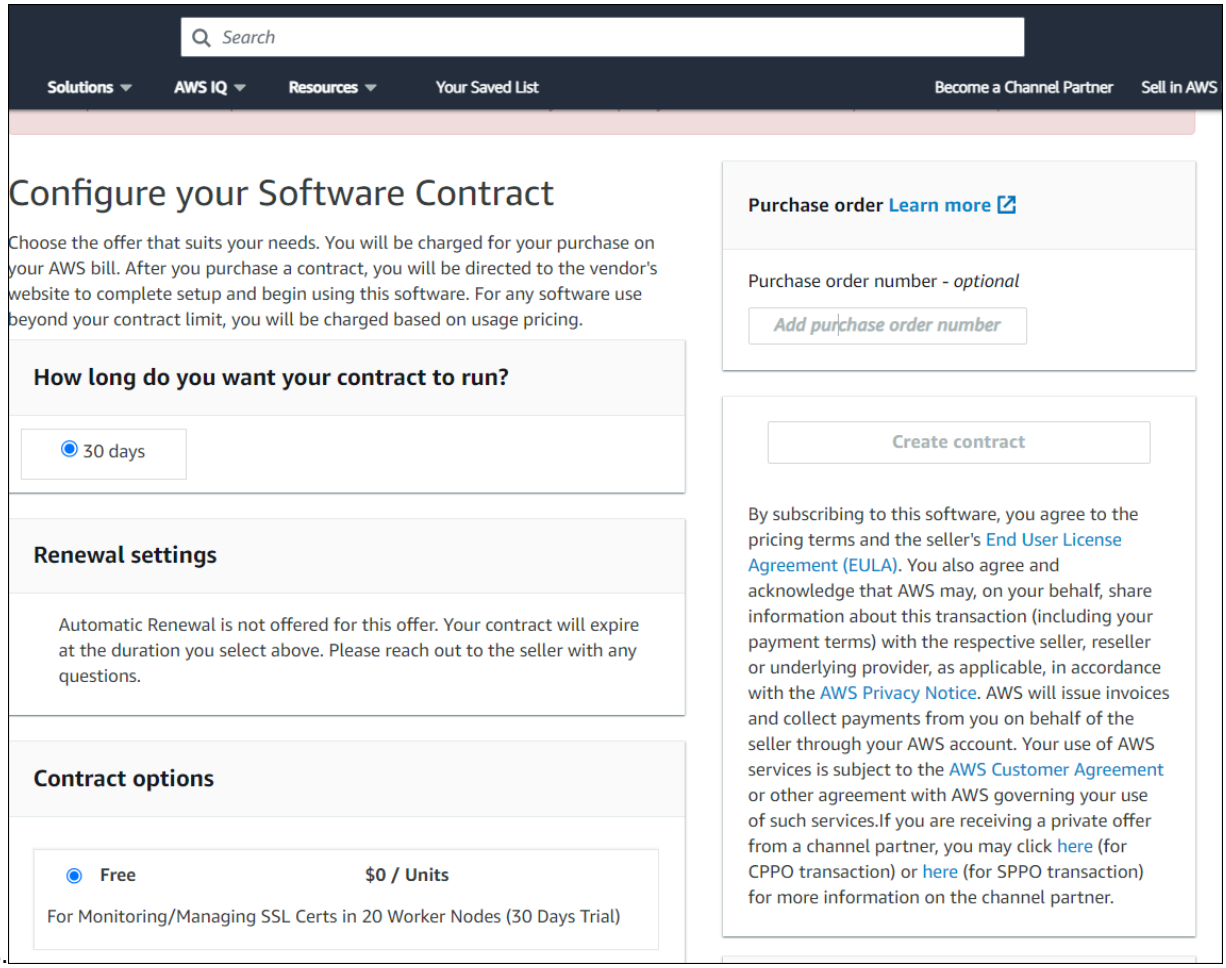
1. Navigate to the AWS Marketplace page.
2. Sign into your account or create your account if you are new to AWS Marketplace.
3. Search for AppViewX KUBE+ in the search bar and select it from the search result.
   The **AppViewX KUBE+** page is displayed.



4. In case you have a private offer from AppViewX, it will be listed on top of the next page. Make sure you select the private offer and not the public offers.

   In case you are here for the first time and do not have a private offer listed, select **Try for free** option.

5. Select the desired **Contract options**.



6. Click **Create contract** to create the contract for AppViewX KUBE+.

   A confirmation dialog box is displayed.

7. In the confirmation dialog box, click **Setup your account** to complete the signup.

   You will be redirected to the AppViewX SaaS registration page.

Step 2: Filling the Sign Up Form

1. To get started with your free trail, enter the following details:

Field	Description
First Name*	Enter your first name.
Last name*	Enter your last name.
Business Email*	Enter your business email address.

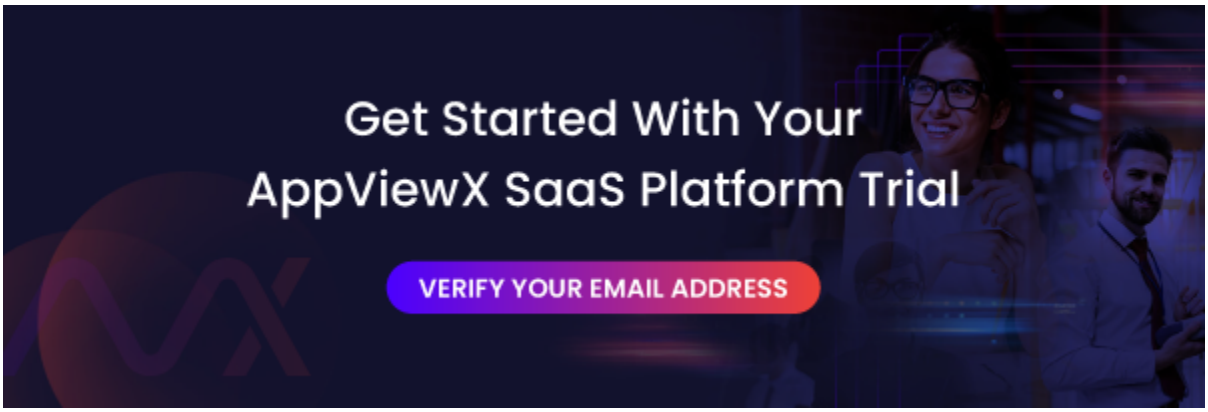
Field	Description
Company Name*	Enter your company name.
Enter Custom Domain*	By default, the company name is auto-filled. Enter a custom domain if you want to.
Select Service Region*	The service region is where your SaaS account will be set up and localized. You cannot migrate data between regions. Select from one of the service regions: • US (Americas) • EMEA • APAC
Select Country*	Select the country from the dropdown list.

Note:
- Fields marked with the asterisk (*) symbol are mandatory.
- If you are creating a free or trial account, there are email restrictions put in place for security reasons. Email addresses from Gmail.com, Outlook.com, Yahoo.com, and other personalized email addresses are restricted and may not be used for trial account creation purposes.

2. From the **What are you trying to solve** list, select the corresponding checkboxes for your requirements.
3. To acknowledge that you have read and reviewed AppViewX's Terms of Service and their Privacy Policy, select the **By checking this box, I acknowledge...** checkbox.
4. Click **Get Started.** The message, *Thank you for signing up for the free trial! You will receive an email from us shortly,* is displayed.

Step 3: Verifying your Email

On clicking **Get Started,** you will get a verification email to your registered email address. Click **Verify Email Address** to get your SaaS account set-up.



Note:

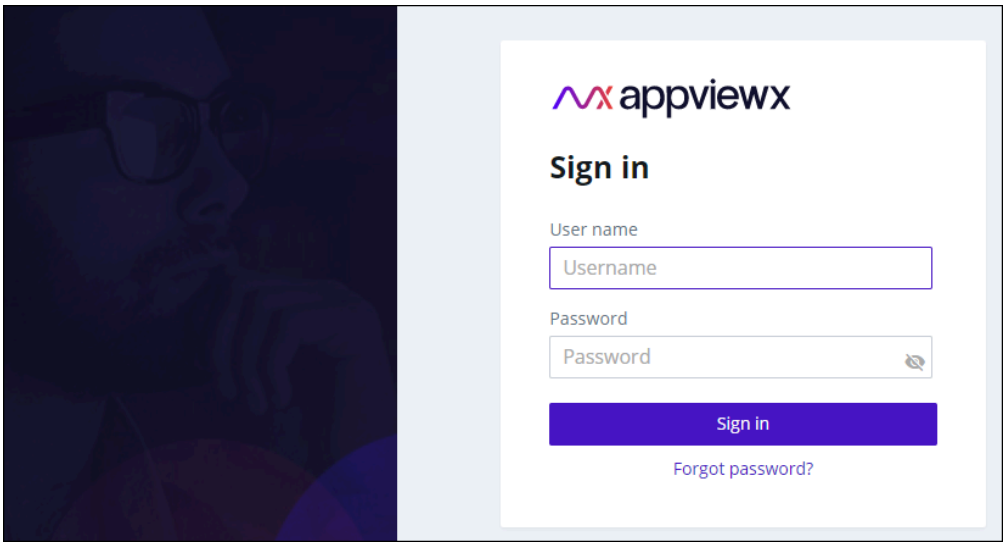
- If you do not see the email in your inbox, then check the Junk/Spam folder. Whitelist the email address so you receive all AppViewX emails in your inbox.
- Confirm your email address within 48hours.

Wait for a couple of minutes until your email address is successfully verified.

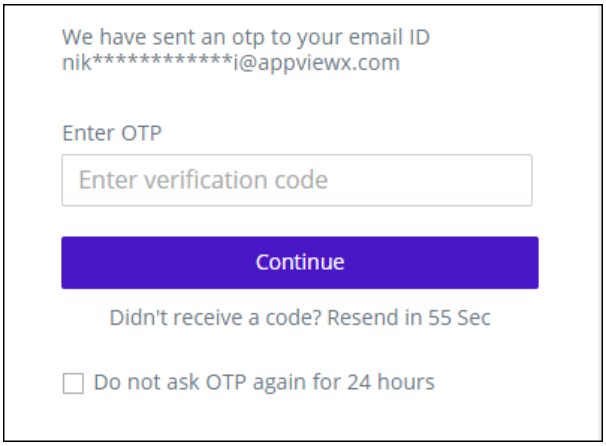
Step 4: Logging in to your SaaS Account

Based on the details entered, you will receive a welcome email on your registered email ID. The email includes your login URL and temporary credentials.

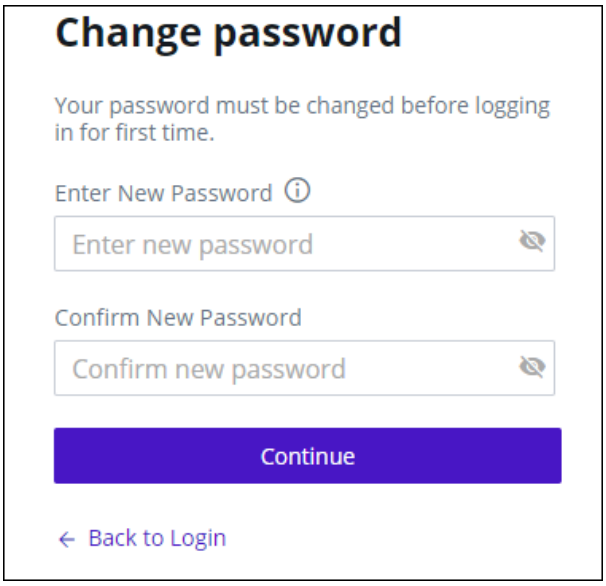
1. Navigate to the login URL.
   The login page is displayed.



2. From the welcome email, login using the credentials provided.
   On successful login, the OTP verification screen is displayed. You will receive the OTP on your registered email address.



3. Enter the OTP received.
   On entering the correct OTP, the **Change password** screen is displayed.



4. Enter and reenter your new password in the **Enter New Password** and **Confirm New Password** fields respectively.

> **Note:** The password must:
> - Have at least one uppercase character
> - Have at least one lowercase character
> - Have one special character such as ~!@#$^&*_-+=|()
> - Have minimum of 6 characters and maximum of 24 characters
> - Not contain user name
> - Not contain more than 3 same characters continuously, for example, aaa
> - Not contain blank space

5. Click **Continue**.

   A message notifying successful password change is displayed.

   You will be redirected to the login page again.

6. Sign in with your new credentials.
7. n the **OTP Verification** screen, enter the OTP received on your registered email and click **Continue**.
8. On the **Terms of Service** screen, select the **I accept the terms and conditions** checkbox and click **Continue**.

   The AppViewX Platform landing page is displayed.

9. To try a product, click **Try Now** to start your 30-day trial of the product.

You will be redirected to the **GET STARTED** page of the selected product.

> **Note:** The 30-day trial period starts from the day you receive the welcome email. The trial period can be extended by 60 more days (which makes the trial duration 90 days). For more details on how you can extend your trial, please reach out to AppViewX Support.

Step 5: Setting up the AppViewX Cloud Connector

The AppViewX Cloud Connector can be set up in two different ways:

- Using the Automated Script

  For instructions on setting up the AppViewX Cloud Consumer using the Automated Script, click here.

- Using AppViewX User Interface

  For instructions on setting up the AppViewX Cloud Consumer using AppViewX User Interface, click here.

To understand the difference between the two methods, click here.

Step 6: Getting Started with AppViewX SaaS

To simplify your interaction with the product's features, AppViewX offers exhaustive documentation in the form of the following guides:

- AppViewX Cloud Connector User Guide
- KUBE+ User Guide

You can access the complete AppViewX documentation here.