

# Integrate Kubernetes Secrets-Agent with Password Safe

The Kubernetes (K8S) Secrets-Agent integration for Password Safe enables the injection of secrets from Password Safe into K8S pods.

Pods can be configured to retrieve secrets from Password Safe before being applied to their primary application. The application consuming the secrets does not need to have any knowledge of Password Safe to use the secret at run time.

Permissions for access to secrets in Password Safe can be granted to specific accounts within BeyondInsight.



**Note:** This Secrets Agent version only works with Password Safe version 23.1.0 and later releases.

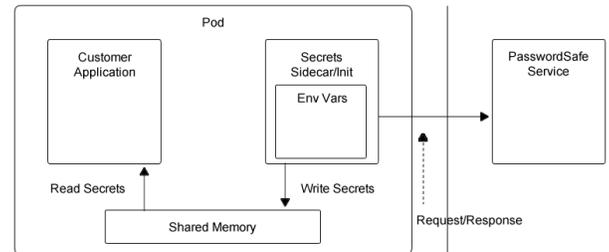
## Overview

Applications can opt in to Password Safe secret retrieval by adding the secrets-agent to their Kubernetes manifests as an initContainer, or a sidecar container. The secrets-agent retrieves secrets and makes them available to the target consumer without requiring that the consumer be aware of Password Safe.

To support this feature, your BeyondInsight instance must be configured with an API registration.

At run time, secrets are retrieved from Password Safe by Kubernetes pods, pictured in the figure below:

1. On startup, the initContainer authenticates to Password Safe.
2. Secrets-agent retrieves secrets from Password Safe and writes them to a shared volume.
3. The end-user application reads the secrets from the shared volume.



## Configure an Application for Secret Retrieval

For an application to opt in to Password Safe secret retrieval, each of the following must be in place:

1. A Password Safe API registration must be configured.
2. The application's manifest must add the secrets-agent container and a Shared Volume.

The below sections first provide a guide to setting up the required authorization to allow a pod's secrets-agent to retrieve secrets. Then secondly, show the pod manifest modifications required to add the secret-retrieval behavior to the pod at run time.

## Permissions - Authorize Access to the Target Secret

The secrets-agent must be provided an API key and a user account in order to access its target secrets in Password Safe. This is accomplished through the following steps:

### Setup

1. Create an API registration in BeyondInsight (does not require a user password).
2. Create or use an existing Secrets Safe group.
3. Create or use an existing BeyondInsight user.
4. Add API registration to the group.
5. Add the user to the group.

### Secrets Safe Setup

Add the Secrets Safe feature to the group.

### Managed Accounts Setup

1. Create or use an existing access policy that has the **View Password Auto Approve** option set.
2. Add the **All Managed Accounts** Smart Group to the BeyondInsight group.
3. Add the access policy to the **All Managed Accounts** Smart Group role, and ensure that both requestor and approver are set.
4. Create or use an existing managed system.
5. Create or use an existing managed account associated with the managed system.
6. Configure the managed account with the **API Enabled** and **Max Concurrent Requests Unlimited** options selected.

## Application Manifest Additions

The final step in injecting secrets into pods is to modify the manifest of the target application to include the resources that retrieve secrets and write them to the pod filesystem.

Below is an example manifest for a deployment that retrieves the secrets at paths `rootFolder/childFolder1/secretTitle` and `rootFolder/` from a Password Safe instance. The secret-retrieval initContainer runs prior to the main application starting, retrieves the target secrets, and writes their contents to files on the shared volume at `/usr/src/app/secrets_files`:

```
apiVersion: v1
kind: Pod
metadata:
  name: passwordsafe-integration
spec:
  volumes:
    - name: secrets
      emptyDir:
        medium: Memory
  initContainers:
```

```
- name: secrets-agent
  image: secrets-agent:latest
  volumeMounts:
    - name: secrets
      mountPath: /usr/src/app/secrets_files
  ports:
    - containerPort: 8000
      name: secrets-agent
  imagePullPolicy: Never
  resources:
    limits:
      memory: "400Mi"
  env:
    - name: SECRETS_PATH
      value: "/usr/src/app/secrets_files"
    - name: BT_API_URL
      value: "https://example.com:443/BeyondTrust/api/public/v3"
    - name: BT_API_KEY
      value: "<API-KEY>;runas=username;"
    - name: SECRETS_LIST
      value: "rootFolder/childFolder1/secretTitle"
    - name: FOLDER_LIST
      value: "rootFolder/childFolder2/"
    - name: MANAGED_ACCOUNTS_LIST
      value: "Server2016Standard/serveruser1"
    - name: BT_VERIFY_CA
      value: "True"
```

Below is an example manifest for a deployment that retrieves the secrets at paths *rootFolder/childFolder1/secretTitle* and *rootFolder/* from a Password Safe instance. The secret-retrieval sidecar runs alongside the main application, retrieves the target secrets, and writes their contents to files on the shared volume at */usr/src/app/secrets\_files*:

```
apiVersion: v1
kind: Pod
metadata:
  name: passwordsafe-integration
spec:
  volumes:
    - name: secrets
      emptyDir:
        medium: Memory
  containers:
    - name: secrets-agent-sidecar
      image: secrets-agent:latest
      volumeMounts:
        - name: secrets
          mountPath: /usr/src/app/secrets_files
      ports:
        - containerPort: 8000
          name: secrets-agent
      imagePullPolicy: Never
      resources:
        limits:
          memory: "400Mi"
      env:
```

```
- name: SECRETS_PATH
  value: "/usr/src/app/secrets_files"
- name: BT_API_URL
  value: "https://example.com:443/BeyondTrust/api/public/v3"
- name: BT_API_KEY
  value: "<API-KEY>;runas=username;"
- name: SECRETS_LIST
  value: "rootFolder/childFolder1/secretTitle"
- name: FOLDER_LIST
  value: "rootFolder/childFolder2/"
- name: MANAGED_ACCOUNTS_LIST
  value: "Server2016Standard/serveruser1"
- name: POLLING_WAIT_BETWEEN_REQUESTS_MINUTES
  value: "20"
- name: BT_VERIFY_CA
  value: "True"
```

After saving the above pod manifest to a `.yaml` file named `secret-retrieval-example.yaml`, apply it to the cluster using `kubectl apply -f secret-retrieval-example.yaml`. This creates the pod on the cluster.

Observe the `initContainer` image pulled using `kubectl`. Verify that the target secret contents are injected into the directory at `/usr/src/app/secrets_files` using `kubectl exec -it <pod-name> sh` to start an interactive shell session inside the running pod. From there, navigate to the `/usr/src/app/secrets_files` directory and inspect the contents of the files in the `secrets_files` folder.

## Usage

Secrets-agent usage is controlled using environment variables. Add the following environment variables to your K8s `yaml` file:

### Environment Variables

#### 1. SECRETS\_PATH

- *Optional* sidecar or `initContainer` environment variable.
- Defaults to: `/usr/src/app/secrets_files`
- The name and path of the folder on the volume to store secrets and their metadata in.
- Example usage:

```
SECRETS_PATH="/usr/src/app/secrets_files"
```

#### 2. BT\_API\_URL

- *Mandatory* sidecar or `initContainer` environment variable.
- The URL for retrieving secrets.
- Example usage:

```
BT_API_URL=https://PasswordSafeInstance.com:443/BeyondTrust/api/public/v3
```

### 3. BT\_API\_KEY

- *Mandatory* sidecar or initContainer environment variable.
- The registered API key.
- Example usage:

```
BT_API_KEY="<API-KEY>;runas=username;"
```

### 4. SECRET\_LIST

- *Optional* sidecar or initContainer environment variable.
- A comma-delimited list of Secrets Safe *path/SecretTitle*.
- Example usage:

```
rootFolder/childFolder1/secretTitle
```

### 5. FOLDER\_LIST

- *Optional* sidecar or initContainer environment variable.
- A comma-delimited list of Secrets Safe folder paths.



**Note:** Only secrets at the level specified are shown. Usage of this variable does not traverse subfolders for secrets.

- Example usage:

```
rootFolder/childFolder2
```

### 6. MANAGED\_ACCOUNTS\_LIST

- *Optional* sidecar or initContainer environment variable.
- A comma-delimited list of system name/managed account name.
- Example usage:

```
MANAGED_ACCOUNTS=server2019/accountName1,server2019/accountName2
```

### 7. BT\_VERIFY\_CA

- *Optional* sidecar or initContainer environment variable.
- Instructs the secrets-agent to not verify the Password Safe certificate authority. If the environment variable is not specified or BT\_VERIFY\_CA=False, the CA will not be verified.
- Example Usage:

```
MANAGED_ACCOUNTS=server2019/accountName1,server2019/accountName2
```

## 8. POLLING\_WAIT\_BETWEEN\_REQUESTS\_MINUTES

- *Mandatory* sidecar-only environment variable.
- When running secrets-agent as a sidecar, you can specify how long to wait between subsequent secrets requests. The recommended wait time is 20 minutes. The minimum wait is 5 minutes.
- If not specified or POLLING\_WAIT\_BETWEEN\_REQUESTS\_MINUTES=0, there will be no polling.



**Note:** If this variable is used for your `initContainer` configuration, the `initContainer` will never complete.

- Example usage:

```
POLLING_WAIT_BETWEEN_REQUESTS_MINUTES=20
```

## 9. BT\_CLIENT\_CERTIFICATE\_PATH

- *Optional* sidecar or `initContainer` environment variable.
- The path to a persistent volume with the client certificate file. If a path is empty, a client certificate will not be used.
- Example usage:

```
BT_CLIENT_CERTIFICATE_PATH="/usr/src/app/certificate/certificate.pfx"
```

## 10. BT\_CLIENT\_CERTIFICATE\_PASSWORD

- *Optional* sidecar or `initContainer` environment variable.
- The client certificate password.
- Example Usage:

```
BT_CLIENT_CERTIFICATE_PASSWORD=password
```

The `initContainer` environment variable list from the manifest example above can be expanded to include these options for a client certificate file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv
spec:
  storageClassName: standard
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/home/docker/"
```

```
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  name: pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 200Mi
  storageClassName: standard
  volumeName: pv
```

```
apiVersion: v1
kind: Pod
metadata:
  name: passwordsafe-integration
spec:
  volumes:
    - name: secrets
      emptyDir:
        medium: Memory
    - name: certificate
      persistentVolumeClaim:
        claimName: pvc
  containers:
    - name: secrets-agent-sidecar
      image: secrets-agent:latest
      volumeMounts:
        - name: secrets
          mountPath: /usr/src/app/secrets_files
        - name: certificate
          mountPath: /usr/src/app/certificate
      ports:
        - containerPort: 8000
          name: secrets-agent
      imagePullPolicy: Never
      resources:
        limits:
          memory: "400Mi"
      env:
        - name: SECRETS_PATH
          value: "/usr/src/app/secrets_files"
        - name: BT_API_URL
          value: "https://example.com:443/BeyondTrust/api/public/v3"
        - name: BT_API_KEY
          value: "<API-KEY>;runas=username;"
        - name: SECRETS_LIST
          value: "rootFolder/childFolder1/secretTitle"
        - name: FOLDER_LIST
          value: "rootFolder/childFolder2"
        - name: MANAGED_ACCOUNTS_LIST
          value: "Server2016Standard/serveruser1"
        - name: POLLING_WAIT_BETWEEN_REQUESTS_MINUTES
          value: "20"
        - name: BT_VERIFY_CA
          value: "True"
```

```
- name: BT_CLIENT_CERTIFICATE_PATH  
  value: "/usr/src/app/certificate/certificate.pfx"  
- name: BT_CLIENT_CERTIFICATE_PASSWORD  
  value: "*****"
```