PROTEGRITY

Protegrity Application Protector On-Premises Immutable Policy User Guide 9.1.0.0

Created on: Aug 8, 2024

Notice

Copyright

Table of Contents

Chapter 1

Introduction to this Guide

This Protegrity Application Protector On-Premises Immutable Policy User Guide discusses about the Immutable Application Protector and its uses. The Guide also provides detailed information and examples of libraries and deployment architectures for all flavors of the Protegrity Immutable Application Protector (IAP).

The purpose of this guide is to help you understand and implement the APIs in your application. Developers who use this guide should be familiar with the operation and management of Protegrity Data Security Platform and Protegrity Immutable Application Protector (IAP).

This guide includes IAP API libraries and definitions and code samples. *Hello World* examples are provided as well to assist in explaining the required steps and requirements for the Protegrity IAP API.

The code samples in this guide represent a small portion of code and are designed to show the basic logic and programming constructions needed to use the Protegrity IAP API effectively.

For more information about IAP API libraries and definitions, code samples, and error codes, refer to the section *Application Protector* in the *Protegrity APIs, UDFs, and Commands Reference Guide 9.1.0.0*.

1.1 Sections contained in this Guide

The guide is broadly divided into the following sections:

- *Section 1 Introduction to this Guide* defines the purpose and scope for this guide. In addition, it explains how information is organized in this guide.
- *Section 2 Protegrity Security Operations* introduces you to the concepts of the Immutable Application Protector, discusses the business problem and solution, and the architecture and workflow of this protector..
- *Section 3 Installing Immutable Application Protector (IAP) C* describes the installation and uninstallation process of the IAP C on various platforms.
- *Section 4 Installing Immutable Application Protector (IAP) Java* describes the installation and uninstallation process of the IAP Java on various platforms.
- *Section 5 Installing Immutable Application Protector (IAP) Python* describes the installation and uninstallation process of the IAP Python on various platforms.
- *Section 6 Installing Immutable Application Protector (IAP) Go* describes the installation and uninstallation process of the IAP Go on various platforms.
- *Section 7 Running IAP - Example* describes how to run the Immutable Application Protector using sample applications.

1.2 Accessing the Protegrity documentation suite

This section describes the methods to access the *Protegrity Documentation Suite* using the *My.Protegrity* portal.

1.2.1 Viewing product documentation

The **Product Documentation** section under **Resources** is a repository for Protegrity product documentation. The documentation for the latest product release is displayed first. The documentation is available in the HTML format and can be viewed using your browser. You can also view and download the *.pdf* files of the required product documentation.

1. Log in to the *My.Protegrity* portal.
2. Click **Resources** > **Product Documentation**.
3. Click a product version.
   The documentation appears.

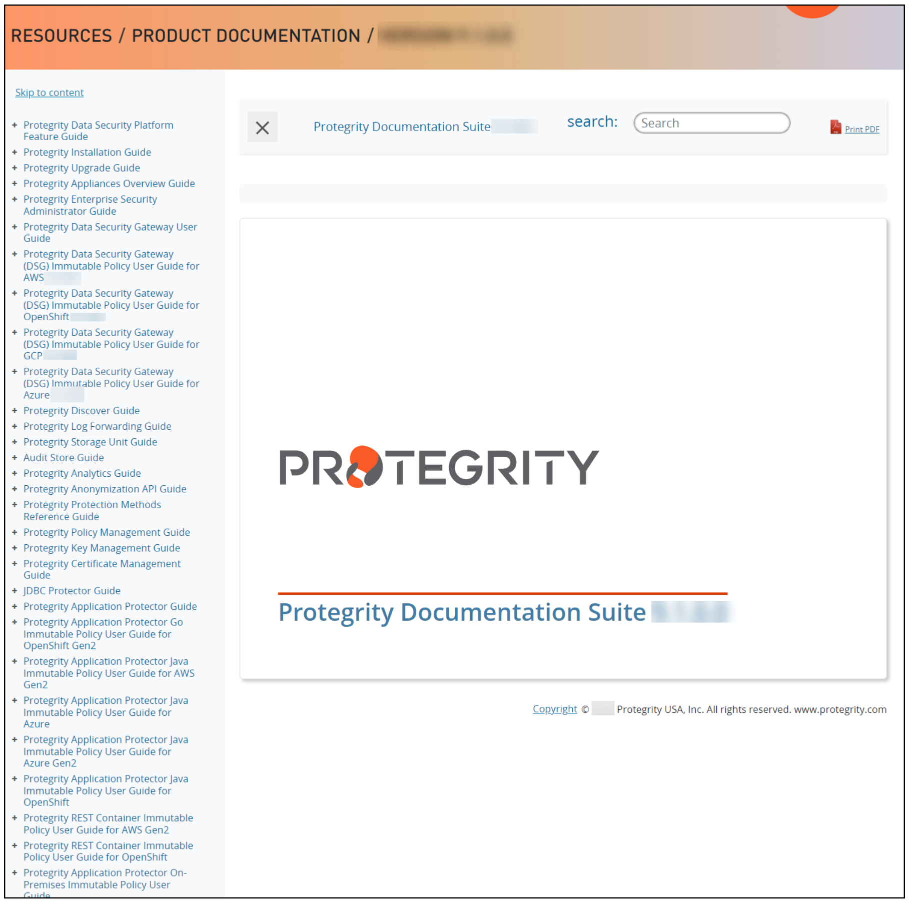

RESOURCES / PRODUCT DOCUMENTATION / [redacted]
Skip to content
Protegrity Data Security Platform Feature Guide
Protegrity Installation Guide
Protegrity Upgrade Guide
Protegrity Appliances Overview Guide
Protegrity Enterprise Security Administrator Guide
Protegrity Data Security Gateway User Guide
Protegrity Data Security Gateway (DSG) Immutable Policy User Guide for AWS [redacted]
Protegrity Data Security Gateway (DSG) Immutable Policy User Guide for OpenShift [redacted]
Protegrity Data Security Gateway (DSG) Immutable Policy User Guide for GCP [redacted]
Protegrity Data Security Gateway (DSG) Immutable Policy User Guide for Azure [redacted]
Protegrity Discover Guide
Protegrity Log Forwarding Guide
Protegrity Storage Unit Guide
Audit Store Guide
Protegrity Analytics Guide
Protegrity Anonymization API Guide
Protegrity Protection Methods Reference Guide
Protegrity Policy Management Guide
Protegrity Key Management Guide
Protegrity Certificate Management Guide
JDBC Protector Guide
Protegrity Application Protector Guide
Protegrity Application Protector Go Immutable Policy User Guide for OpenShift Gen2
Protegrity Application Protector Java Immutable Policy User Guide for AWS Gen2
Protegrity Application Protector Java Immutable Policy User Guide for Azure
Protegrity Application Protector Java Immutable Policy User Guide for Azure Gen2
Protegrity Application Protector Java Immutable Policy User Guide for OpenShift
Protegrity REST Container Immutable Policy User Guide for AWS Gen2
Protegrity REST Container Immutable Policy User Guide for OpenShift
Protegrity Application Protector On-Premises Immutable Policy User Guide
Protegrity Documentation Suite [redacted]
search:
Search
Print PDF
PROTEGRITY
Protegrity Documentation Suite [redacted]
Copyright © [redacted] Protegrity USA, Inc. All rights reserved. www.protegrity.com

Figure 1-1: Documentation

4. Expand and click the link for the required documentation.
5. If required, then enter text in the **Search** field to search for keywords in the documentation.

   The search is dynamic, and filters results while you type the text.
6. Click the **Print PDF** icon from the upper-right corner of the page.
   The page with links for viewing and downloading the guides appears. You can view and print the guides that you require.

1.2.2 Downloading product documentation

This section explains the procedure to download the product documentation from the *My.Protegrity* portal.

1. Click **Product Management** > **Explore Products**.
2. Select **Product Documentation**.
   The **Explore Products** page is displayed. You can view the product documentation of various Protegrity products as per their releases, containing an overview and other guidelines to use these products at ease.
3. Click **View Products** to advance to the product listing screen.
4. Click the **View** icon (👁) from the **Action** column for the row marked **On-Prem** in the **Target Platform Details** column.
   If you want to filter the list, then use the filters for: **OS**, **Target Platform**, and **Search** fields.
5. Click the icon for the action that you want to perform.

Chapter 2

Protegrity Security Operations

This section describes the Protegrity security operations based on shared memory.

2.1 Business Problem

This section identifies the problems a company faces while protecting data:

- Protegrity customers need a data security product that can be configured on a primary system and then it can be applied on various on-premise systems.
- An independent data security product that does not need to be connected to a centralised administrative system like the ESA to perform data security operations.
- A data security product that simplifies the deployment and implementation of protecting on-premises sensitive data.

2.2 Protegrity Solution

The Protegrity solution leverages shared memory capabilities to deliver a Protegrity data security product with the following characteristics:

- *Immutable Deployment*: You can run the Application Protector C, Java, Python, and Go without the PEP server, as connectivity to the ESA is not required for the policy consuming machine.
- *On-premises solution*: It enables you to protect the sensitive data present on various on-premise systems without the need to be connected to the administrator system.
- Use the following components to fetch the encrypted policy and decrypt it:
  - *REST Client Utility* – It fetches the policies from the datastore present on the ESA and creates an encrypted policy package on the local system.
  - *Import Utility* – It decrypts the policy package and publishes the decrypted policies to the shared memory.

2.3 Architecture and Components for Gen1

This section describes the architecture, the individual components, and the workflow of the Protegrity Immutable Application Protector (IAP) On-premises solution for Gen1.

Note:
This is applicable only for the Windows IAP deployment.

The IAP On-premises deployment for Gen1 consists of the following stages:

- *Stage 1*: Running the IMP Creator script on the Policy Package Creator Machine to fetch policies from the ESA and create a policy package on your local system in an encrypted format.
- *Stage 2*: Installing the IAP on a Policy Consuming Machine, which uses the policy package created in stage 1.

The following describes the architecture diagram and the steps for Stage 1 - Creating a policy package on the Policy Package Creator Machine.

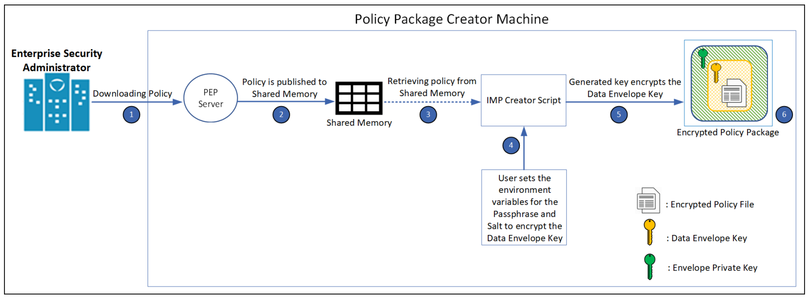

Policy Package Creator Machine
Enterprise Security Administrator
Downloading Policy
1
PEP Server
Policy is published to Shared Memory
2
Shared Memory
Retrieving policy from Shared Memory
3
IMP Creator Script
4
User sets the environment variables for the Passphrase and Salt to encrypt the Data Envelope Key
Generated key encrypts the Data Envelope Key
5
6
Encrypted Policy Package
: Encrypted Policy File
: Data Envelope Key
: Envelope Private Key

Figure 2-1: Stage 1: Creating a policy package on the Policy Package Creator Machine

The following steps describe the workflow of creating a policy package on the Policy Package Creator Machine:

1. The PEP server fetches the policy from the ESA.
2. The policy fetched from the ESA is published to the shared memory.
3. The policy is retrieved from the shared memory.
4. The user sets the environment variables for the passphrase and salt.

    Caution:
    The data in the salt and passphrase files is in clear text. This is a known security issue. Customers must ensure that these files are not exposed to unauthorized users. If unauthorised users get access to the salt and passphrase files, then they can decrypt the policy package.

5. The envelope private key generated from the passphrase and salt files encrypts the data envelope key.
6. An encrypted policy package is created at the local path specified by the user.

The following describes the architecture diagram and the steps for Stage 2 - Installing IAP On-premises on the Policy Consuming Machine.

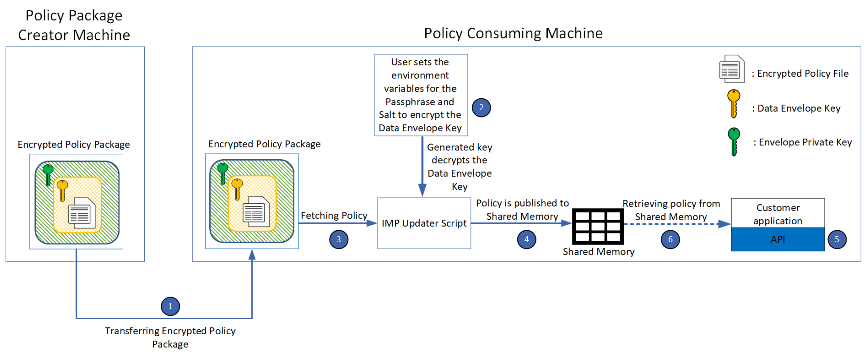

Policy Package Creator Machine
Policy Consuming Machine
Encrypted Policy Package
Encrypted Policy Package
User sets the environment variables for the Passphrase and Salt to encrypt the Data Envelope Key
2
Generated key decrypts the Data Envelope Key
Fetching Policy
3
IMP Updater Script
Policy is published to Shared Memory
4
Shared Memory
Retrieving policy from Shared Memory
6
Customer application
API
5
: Encrypted Policy File
: Data Envelope Key
: Envelope Private Key
1
Transferring Encrypted Policy Package

Figure 2-2: Stage 2: Installing IAP On-premises on the Policy Consuming Machine.

The following steps describe the workflow of installing the IAP On-premises on the Policy Consuming Machine.

1. The user copies the encrypted policy package from the Policy Package Creator Machine to the Policy Consuming Machine.
2. The user sets the environment variables for the passphrase and salt. The envelope private key generated from the passphrase and salt files encrypts the data envelope key.

   Caution:

   The data in the passphrase and salt files is in clear text. This is a known security issue. Customers must ensure that these files are not exposed to unauthorized users. If unauthorised users get access to the passphrase and salt files, then they can decrypt the policy package.

   Note:

   Ensure that you use the same passphrase and salt on the Policy Package Creator Machine and the Policy Consuming Machine.

3. The IMP Updater script fetches the policy from the encrypted policy package.
4. The policy fetched from the encrypted policy package is published to the shared memory.
5. The IAP retrieves the policy from the shared memory.
6. The client application sends a request to the IAP APIs for protecting, unprotecting, and reprotecting data.

2.4 Architecture and Components for Gen2

This section describes the architecture, the individual components, and the workflow of the Protegrity Immutable Application Protector (IAP) On-premises solution.

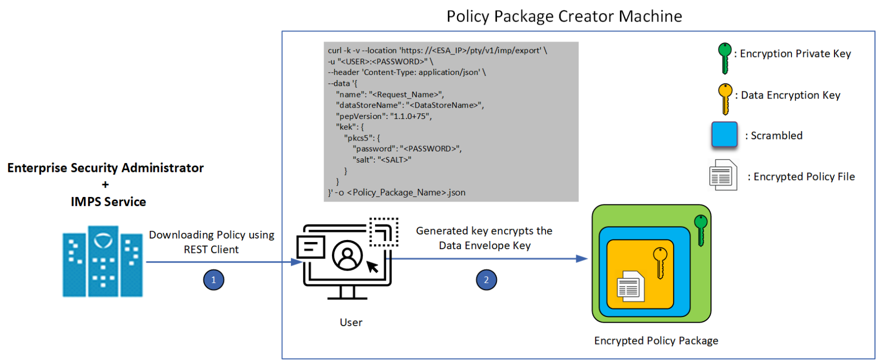
Note:

This is applicable only for the Linux IAP deployment.

The IAP On-premises deployment consists of the following two stages:

- *Stage 1*: Running the REST Client command on the machine that is creating the Policy Package to fetch policies from the ESA and create a policy package on your local system in an encrypted format.
- *Stage 2*: Installing the IAP on the machine that is consuming the Policy, which uses the policy package created in stage 1.

The following describes the architecture diagram and the steps for Stage 1 - Creating the policy package on the Policy Package Creator Machine.



```
curl -k -v --location 'https: //<ESA_IP>/pty/v1/imp/export' \
-u "<USER>:<PASSWORD>" \
--header 'Content-Type: application/json' \
--data '{
  "name": "<Request_Name>",
  "dataStoreName": "<DataStoreName>",
  "pepVersion": "1.1.0+75",
  "kek": {
    "pkcs5": {
      "password": "<PASSWORD>",
      "salt": "<SALT>"
    }
  }
}' -o <Policy_Package_Name>.json
```

Policy Package Creator Machine
: Encryption Private Key
: Data Encryption Key
: Scrambled
: Encrypted Policy File
Enterprise Security Administrator
+
IMPS Service
Downloading Policy using REST Client
1
User
Generated key encrypts the Data Envelope Key
2
Encrypted Policy Package

Figure 2-3: Stage 1: Creating policy package on Policy Package Creator Machine

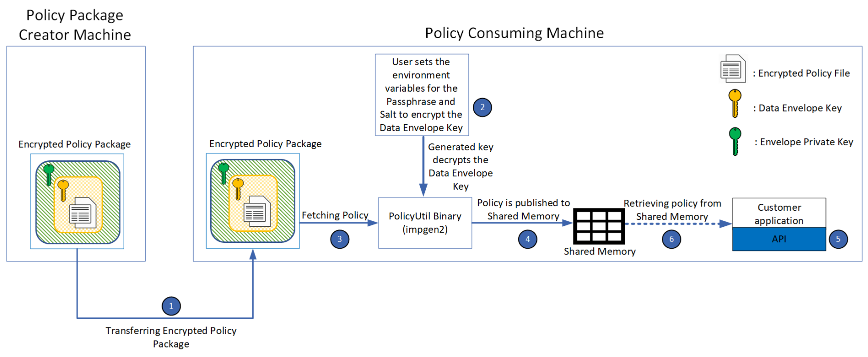
On the policy package creator machine, the user must provide the following details in the REST Client command.

1. Username and password of the user who has the *Export IMP* permissions.
2. The ESA IP address.
3. Datastore name containing the required policy(s).
4. PEP version.
5. Password and salt.
6. Policy package name.

Note: The PEP version passed in the REST command must match the PEP version that the ESA supports. This can be cross checked from the *manifest.json* file, which contains the PEP version.

Caution: This is a known security issue that the passphrase, salt, and user password is in clear text in the REST Client command. You must ensure that these fields are passed to the REST command in a secured way. If an unauthorized user gets access to these fields, then they can decrypt the policy package.

The following describes the architecture diagram and the steps for Stage 2 - Installing IAP On-premises on a Policy Consuming Machine.



Policy Package Creator Machine
Policy Consuming Machine
User sets the environment variables for the Passphrase and Salt to encrypt the Data Envelope Key
2
: Encrypted Policy File
: Data Envelope Key
: Envelope Private Key
Encrypted Policy Package
Encrypted Policy Package
Generated key decrypts the Data Envelope Key
Fetching Policy
3
PolicyUtil Binary (impgen2)
Policy is published to Shared Memory
4
Shared Memory
Retrieving policy from Shared Memory
6
Customer application
API
5
1
Transferring Encrypted Policy Package

Figure 2-4: Stage 2: Installing IAP On-premises on a Policy Consuming Machine

The following steps describe the workflow of installing the IAP On-premises on a Policy Consuming Machine.

1. The user copies the encrypted policy package from the Policy Package Creator Machine to the Policy Consuming Machine.
2. The user installs the IAP package.
3. The user can either set the environment variables for the passphrase and salt or set the path to the file that contains the passphrase and salt.

   Caution:

> This is a known security issue that data in passphrase and salt files are in clear text. The customer must ensure that these files are not exposed to unauthorized users. If an unauthorised user gets access to the passphrase and salt files, then they can decrypt the policy package.

> **Note:**
> Ensure that you use the same passphrase and salt on the Policy Package Creator machine and the Policy Consuming Machine.

4. The PolicyUtil Binary, impgen2, decrypts the policy package using the passphrase and salt and publishes the policy on the the shared memory.
5. The IAP retrieves the policy from shared memory.
6. The client application sends a request to the IAP APIs for protecting, unprotecting, and reprotecting data.

2.5 Limitation

Audit logs will be redirected to standard output.

Chapter 3

Installing Immutable Application Protector (IAP) C

The Immutable Application Protector C provides APIs for performing security operations.

All standard protection techniques offered by Protegrity are applicable to the Immutable Application Protector C.

3.1 Setting up Immutable Application Protector (IAP) C on Linux or Unix

This section describes how to set up the IAP C on a Linux or Unix platform.

3.1.1 Verifying the Prerequisites

This section describes the prerequisites for installing the IAP C on a Linux or Unix platform.

Before you begin
Ensure that the following prerequisites are met before installing the IAP C on a Linux or Unix platform:

- Glibc, version 2.4 or later, is installed.
- GCC, version 3.2.3 or later, is installed.
- The ESA is installed, configured, and running.
- The IP address or host name of the ESA is noted.

3.1.2 Downloading the Immutable Application Protector C Installation Package

This section describes the steps to download the IAP C installation package on a Linux or Unix platform.

To setup the IAP C on the Linux or Unix platform:

1. Download the *IAP_Linux-ALL-<bit-version>_x86-<bit-version>_GCC-3.2.3_<version>.tgz* installation package to any location on the machine where you want to install the protector.
2. Create a directory on your Policy Package Creator Machine where you want to install the protector.
3. Extract the IAP C installation package using the following command.

```
tar -xvf IAP_Linux-ALL-<bit-version>_x86-<bit-version>_GCC-3.2.3_<version>.tgz
```

   The following setup files are extracted:

   - *PolicyUtilSetup_Linux_x64_<version>.sh*

- *XCDevIMSetup_Linux_x64_<version>.sh*
- *manifest.json*

3.1.3 Installing the Immutable Application Protector C on Policy Consuming Machine

This section describes how to install the IAP C on a Linux or Unix platform.

➤ To install the IAP C on the Linu or Unix platform:

Run the following command to install the IAP C.

```
./XCDevIMSetup_Linux_<bit-version>_<version>.sh
```

The library directory path for the IAP C is the `<installation_directory>/defiance_xc` directory.

3.1.4 Running the REST Client utility on the Policy Package Creator Machine

This section describes how to run the cURL utility on the Policy Package Machine.

➤ To run the cURL utility:

1. Run the following command to know the PEP version supported by the ESA:
   curl -k -v -u "<user>:<password>" -X GET https://<ESA_IP_ADDRESS>/pty/v1/imp/version

   Sample output:

```
{
  "version" : "<version>",
  "buildVersion" : "<build version>",
  "pepVersions" : [ "<pepVersions supported by the ESA>" ]
}
```

   Note: Ensure that the password complexity meets OWASP recommendations. For more information about the password requirements, refer to section *Password Strength*.

2. Run the following command to fetch the policy dump from the ESA:

```
curl -k -v --location 'https: //<ESA_IP>/pty/v1/imp/export' \
-u "<USER>:<PASSWORD>" \
--header 'Content-Type: application/json' \
--data '{
    "name": "<Request_Name>",
    "dataStoreName": "<DataStoreName>",
    "pepVersion": "<pepVersion>",
    "emptyString":"NULL",
    "caseSensitive":"YES",
    "output":"STDOUT",
    "kek": {
        "pkcs5": {
            "password": "<PASSWORD>",
            "salt": "<SALT>"
        }
    }
}' -o <Policy_Package_Name>.json
```

> **Note:** Ensure that the output parameter is set to *STDOUT* to avoid the loss of logs.

> **Note:** Ensure that the value passed for the *pepVersion* parameter in the command matches with the *pepVersion* parameter in the *manifest.json* file.

> **Note:** The values passed for the parameters in the command must match the corresponding values in the ESA.

> **Note:** It is recommended that the *<USER>* exporting the policy should have minimum privileges. The least privilege that is required is the *Export IMP* permissions on the ESA.

An encrypted policy package file is generated at the local path defined by the user.

3. Ensure that you have copied the encrypted policy package to the Policy Package Consuming Machine.

3.1.5 Running the PolicyUtil Binary on the Policy Consuming Machine

This section describes how to run the *import* utility on the Policy Package Consuming Machine.

➤ To run the *import* utility:

1. Run the following command to install the PolicyUtil setup, which has the *impgen2* binary.

```
./PolicyUtilSetup_Linux_x64_<version>.sh
```

2. The following directory is created in the `/opt/protegrity` directory.
   `/opt/protegrity/imp-policy-util`
3. Navigate to the `/opt/protegrity/imp-policy-util/bin/impgen2` directory available on the Policy Package Consuming Machine.
4. Run the *impgen2* binary by using the following command.

```
./impgen2 import -passphrase <password> -salt <salt> -policy-file <file_path>/
<policy_name>.json -pepversion <pepversion>
```

> **Note:** The IMP package PEP version must match the protector PEP version.

> **Note:**
> Passphrase and salt can either be given as clear texts or as environment variables.
>
> For example: *PTY_SALT* or *PTY_SALT_PATH*.
>
> If a file is used for storing passphrase and salt, then ensure that there is no new line in it. For more information regarding the binary, run `./impgen2 help`.

The *PolicyUtil* binary repopulates the shared memory on the Policy Package Consuming Machine.

5. Run the following command to view the populated shared memory on the Policy Package Consuming Machine.

   ```
   ipcs -a
   ```

   After the shared memory is populated, the protector can be used to perform the protect, unprotect, and reprotect operations.

3.1.6 Verifying the Policy Health on the Policy Package Consuming Machine

This section describes how to verify the policy health on the Policy Package Consuming Machine.

To verify the policy health:

1. Navigate to the `/opt/protegrity/imp-policy-util/bin/` directory.
2. Run the following command to verify the policy health.

   ```
   ./impgen2 health
   ```

   The following output is expected.

   ```
   health check status: healthy
   ```

3.1.7 Uninstalling PolicyUtil Binary from Policy Package Consuming Machine

This section describes how to uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine.

To uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine:

1. Navigate to the `/opt/protegrity/imp-policy-util` directory.
2. Remove the `/imp-policy-util` directory.

3.1.8 Uninstalling Immutable Application Protector C from Linux or Unix

This section describes how to uninstall the IAP C from a Linux or Unix platfrom.

To uninstall the IAP C from the Linux or Unix platform:

1. Navigate to the installation directory `/opt/protegrity/defiance_xc/bin`.
2. Remove the `/bin` directory.

The IAP C is uninstalled.

3.1.9 Running IAP C - Example

This section provides examples on how to use the IAP C protect, unprotect, and reprotect APIs. It is assumed that policies are made available to the Policy Consuming Machine in a decrypted form and the IAP C has been successfully installed.

For more information on running the IAP C sample application, refer to the section *Running IAP - Example*.

3.2 Setting up Immutable Application Protector (IAP) C on Windows

This section describes how to set up the IAP C on a Windows platform.

3.2.1 Verifying the Prerequisites

Ensure that the following prerequisites are met:

- The Windows operating system, 32-bit or 64-bit is installed, configured, and running
- The ESA appliance is installed, configured, and running
- The IP address or host name of the ESA is noted
- Visual Studio, version 2016 or later, is installed
- Python, version 3.10.0 or later, is installed
- Install the cryptography package using the following command.

  ```
  pip install cryptography
  ```

- Install the password strength package using the following command.

  ```
  pip install password-strength
  ```

3.2.2 Downloading the Immutable Application Protector (IAP) C Installation Package

This section describes how to download the IAP C installation package on a Windows platform.

➤ To download the IAP C installation package on the Windows platform:

1. Create a directory on your machine where you want to install the protector.
2. Download the *IAP_WIN-ALL-<bit-version>_<architecture>_VS-2K16_<version>.zip* installation package.
3. Extract the files from the *IAP_WIN-ALL-<bit-version>_<architecture>_VS-2K16_<version>.zip* installation package.

   The following files are extracted:

   - *XCDevIMSetup_Windows_<bit-version>_<version>.exe*
   - *LogforwarderSetup_Windows_x64_<version>.exe*
   - *PepServerSetup_Windows_<bit-version>_<version>.exe*
   - *PtyShmCatSetup_Windows_<bit-version>_<version>.exe*
   - *PtyShmPutSetup_Windows_<bit-version>_<version>.exe*

   Note: The Log Forwarder is unsupported by the IAP C on the 32-bit Windows deployments.

3.2.3 Installing Log Forwarder on the Policy Package Consuming Machine

This section describes how to install the Log Forwarder on the Policy Package Consuming Machine on a Windows platform using the Windows Wizard and through the Silent mode of installation.

Note:

Installation of the Log Forwarder component is optional for the IAP C on the 64-bit Windows deployments.

If you want to forward the logs to a Security Information and Event Management (SIEM), then install the Log Forwarder on the Policy Package Consuming Machine.

Note: The Log Forwarder is unsupported by the IAP C on the 32-bit Windows deployments.

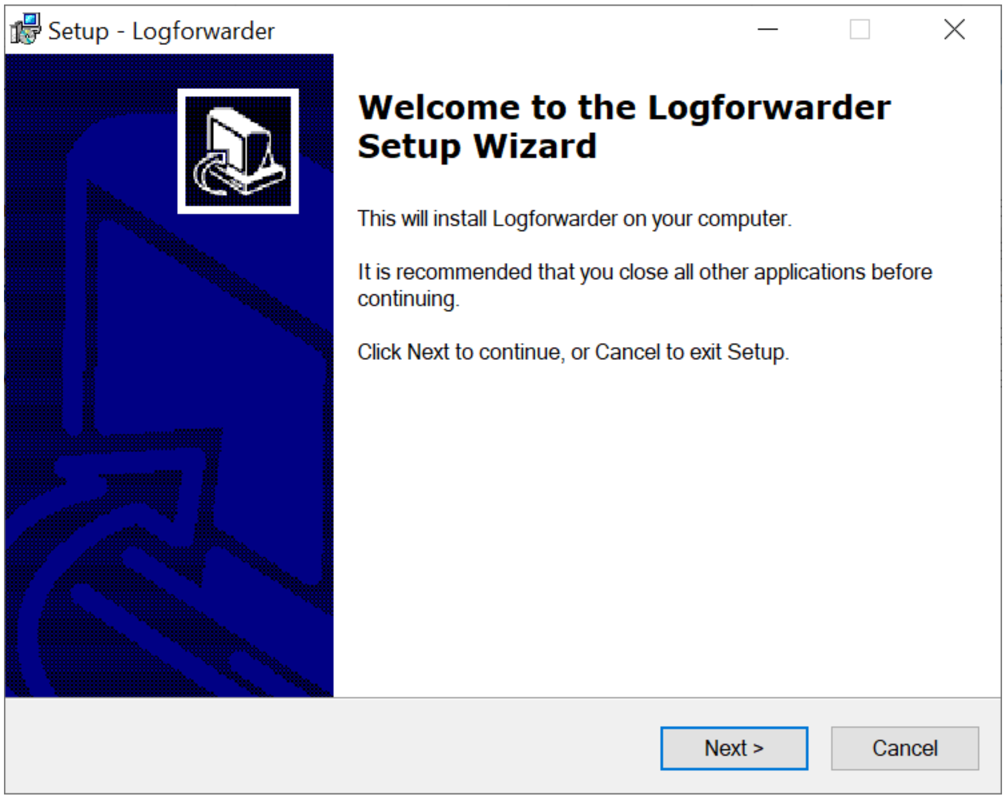
3.2.3.1 Using Windows Wizard

This section describes how to install the Log Forwarder on the Policy Package Consuming Machine on a Windows platform using the Windows Wizard.

To install the Log Forwarder on the Policy Package Consuming Machine using the Windows Wizard:

1. Run the *LogforwarderSetup_Windows_x64_<version>.exe* installer from the created directory.

   The **Welcome to the Log Forwarder Setup Wizard** screen appears.



Welcome to the Logforwarder Setup Wizard

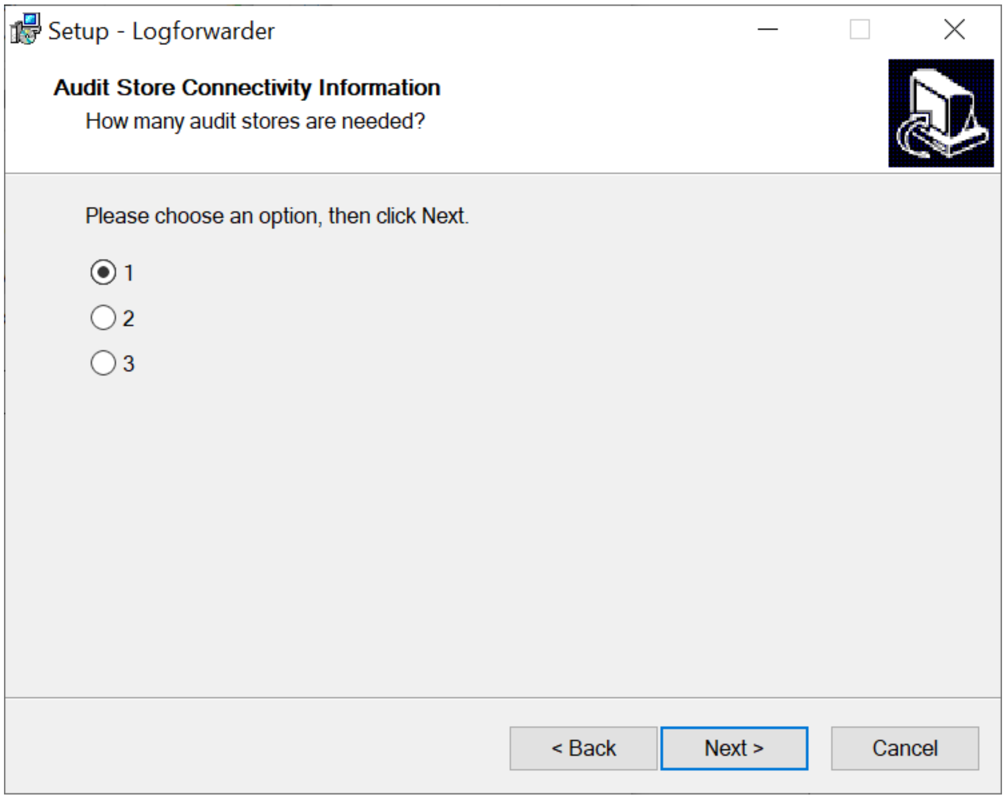
This will install Logforwarder on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

2. Click **Next**.

   The **Audit Store Connectivity Information** screen appears to select the number of audit stores that are needed.

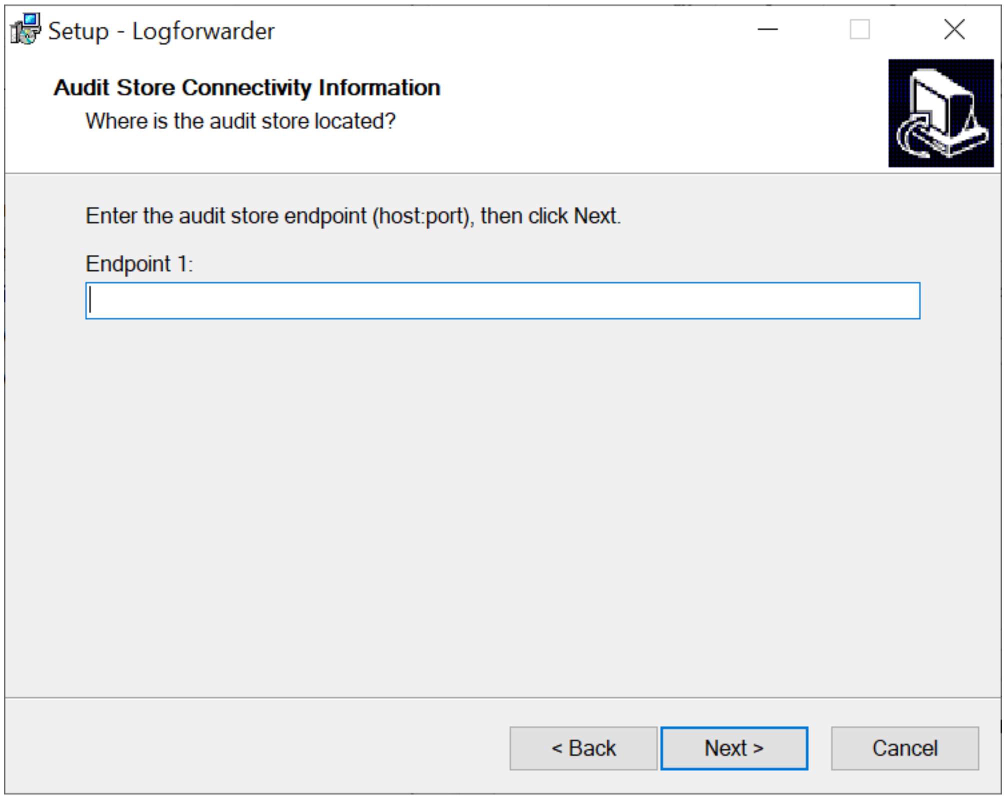


Audit Store Connectivity Information
How many audit stores are needed?
Please choose an option, then click Next.
[✓ 1] 2 3

3. Select the number of Audit Stores as required.

   The maximum number of Audit Stores that can be configured is 3.
4. Click **Next**.

   The **Audit Connectivity Information** screen appears to enter the location of the audit store.

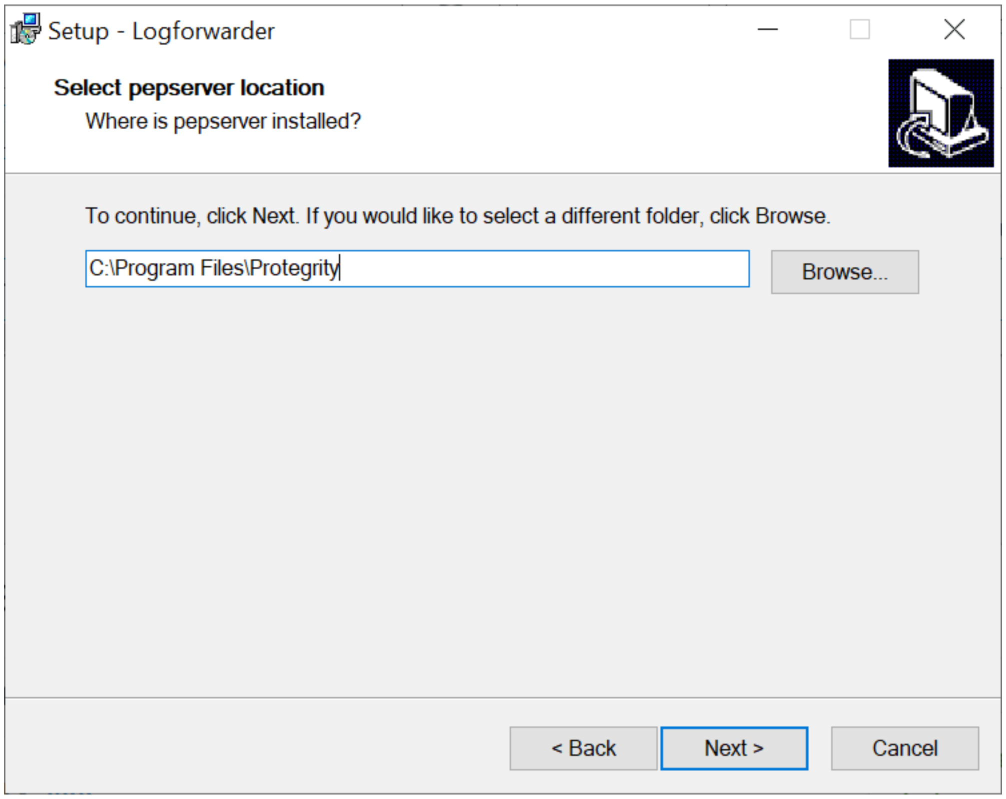


Audit Store Connectivity Information
Where is the audit store located?
Enter the audit store endpoint (host:port), then click Next.
Endpoint 1:

5. Enter the Audit Store endpoint as *Hostname/IPAddress:port* in the **Endpoint 1** field.

   Note:
   The default port number for the Audit Store is *9200*.

6. Click **Next**.
   The **Select pepserver location** screen appears.

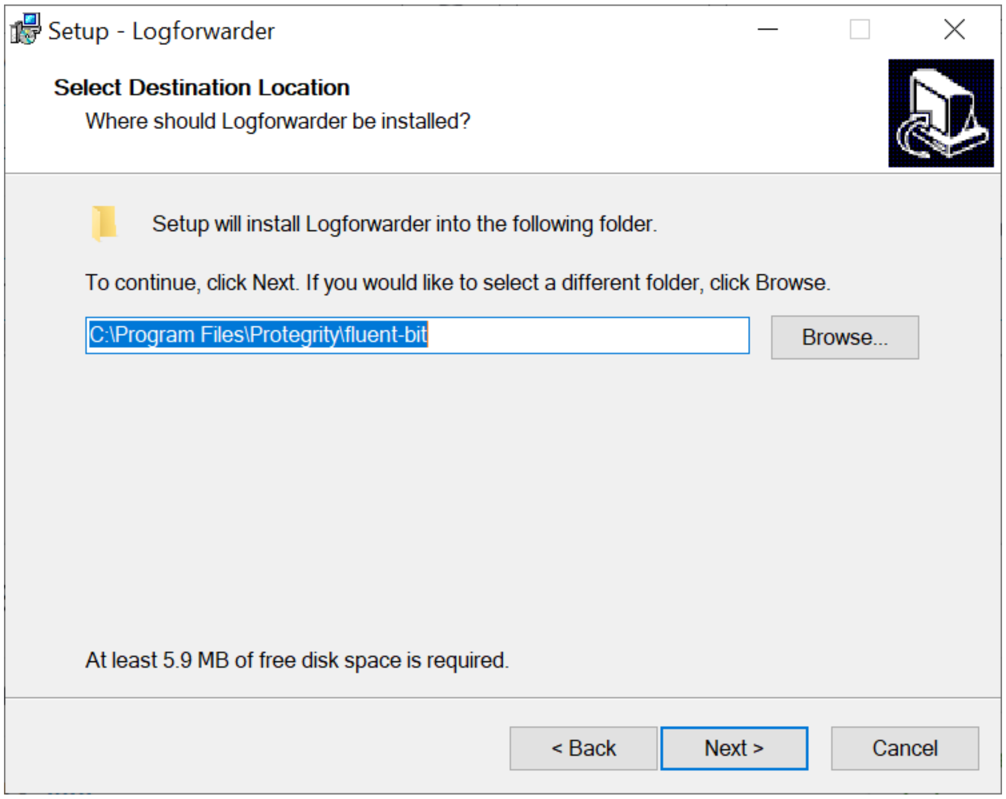

Setup - Logforwarder

Select pepserver location
Where is pepserver installed?

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity

7. Click **Next**.
   The **Select Destination Location** screen appears.


Setup - Logforwarder

Select Destination Location

Where should Logforwarder be installed?

Setup will install Logforwarder into the following folder.

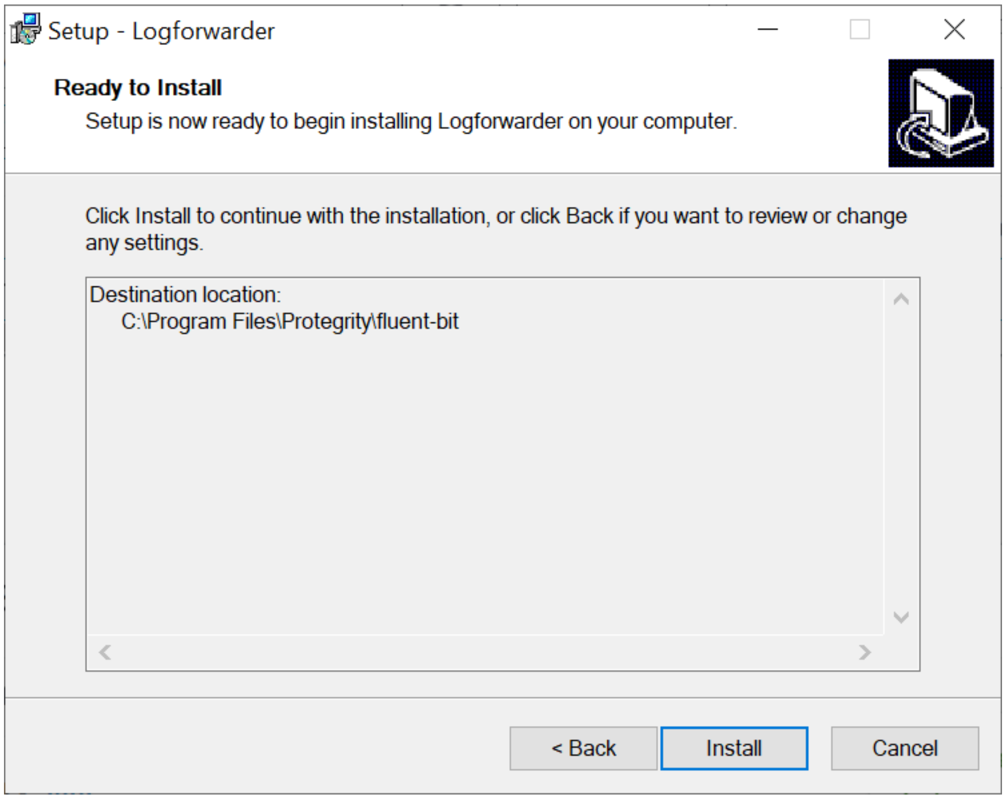
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\fluent-bit

At least 5.9 MB of free disk space is required.

8. Set the installation directory for the Log Forwarder to *`C:\Program Files\Protegrity\fluent-bit`*.
9. Click **Next**.

   The **Ready to Install** screen appears.


Setup - Logforwarder

Ready to Install

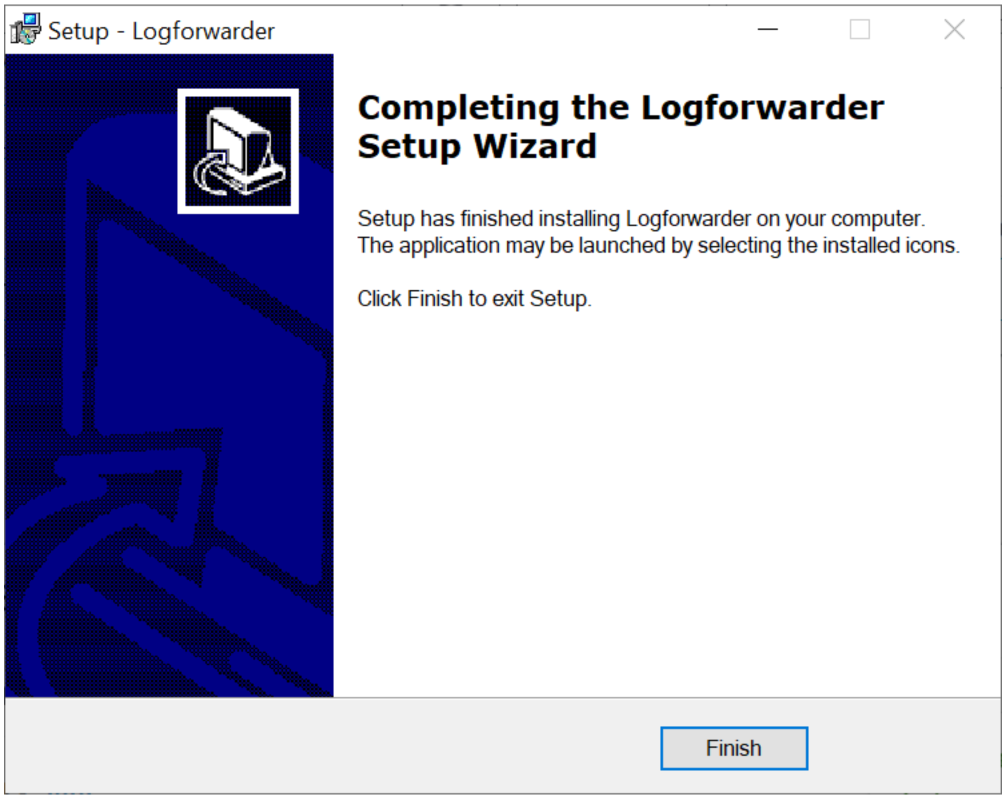
Setup is now ready to begin installing Logforwarder on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\Protegrity\fluent-bit

10. Click **Install**.

    The **Completing the Logforwarder Setup Wizard** screen appears.


Setup - Logforwarder

Completing the Logforwarder Setup Wizard

Setup has finished installing Logforwarder on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

11. From the **Completing the Logforwarder Setup Wizard** screen, click **Finish** to complete the installation and exit.

    The directories are created under the installation directory that was defined, and the installation files are installed in these directories.

The Log Forwarder is installed successfully.

3.2.3.2 Using Silent Mode of Installation

This section describes how to install the Log Forwarder on the Policy Package Consuming Machine on a Windows platform through the Silent mode of installation.

You can also execute the Log Forwarder installer without any manual intervention, which is also known as the Silent mode of installation. The following parameters must be provided to execute the installer in the Silent mode.

Table 3-1: Parameter List for Silent Installation

Parameter	**Description**
-endpoint1, -endpoint2, -endpoint3	Audit Store IP address and the port number where the Log Forwarder listens for logs. **Note:** The default port number is *9200*. **Note:** The parameters *-endpoint2* and *-endpoint3* are optional.
-dir	Installation directory of the Log Forwarder, which is optional. If the installation directory is not specified, then the installation path is the default directory, which is the `C:\Program Files\Protegrity\fluent-bit` directory.

Parameter	Description
-pepdir	Installation directory of the PEP server, which is optional. If the installation directory is not specified, then the installation path is the default directory, which is the `C:\Program Files\Protegrity` directory.

At the command prompt, type the following command from the installer directory.

```
.\LogforwarderSetup_Windows_x64.exe -endpoint1 <ip address:port number> [-endpoint2 <ip
address:port number>]
[-endpoint3 <ip address and port number>]
```

If you want to install the Log Forwarder and the PEP server in a directory other than the default directory, then you can add the *-dir* parameter to the command to specify the Log Forwarder installation directory and the *-pepdir* parameter to the command to specify the PEP server installation directory.

The following snippet displays a sample command.

```
.\LogforwarderSetup_Windows_x64.exe -endpoint1 <ip address:port number> [-endpoint2 <ip
address:port number>]
[-endpoint3 <ip address and port number>] -dir <Log Forwarder installation directory> -pepdir
<PEP server installation directory>
```

3.2.4 Installing PEP server on the Policy Package Creator Machine

This section describes how to install the PEP server on the Policy Package Creator Machine on a Windows platform using the Windows Wizard and through the Silent mode of installation.

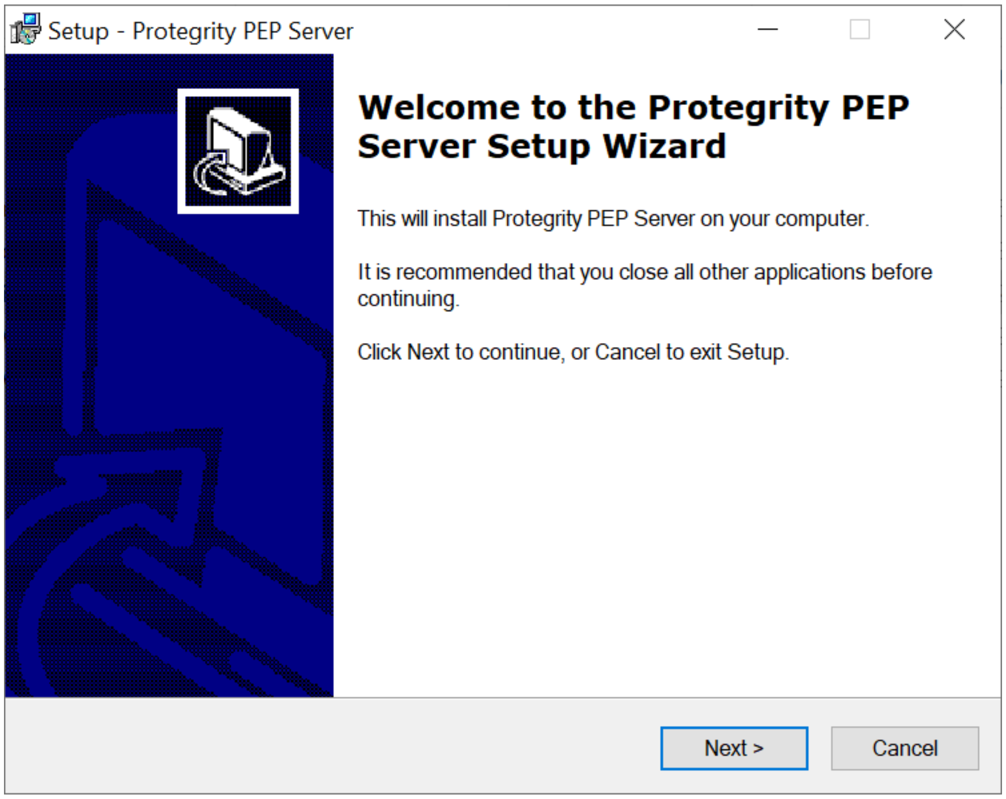
3.2.4.1 Using Windows Wizard

This section describes how to install the PEP server on the Policy Package Creator Machine on a Windows platform using the Windows Wizard.

➤ To install the PEP server on the Policy Package Creator Machine using the Windows Wizard:

1. Run the *PepServerSetup_Windows_<bit-version>_<version>.exe* file from its installed directory.

   The **Welcome to the Protegrity PEP Server Setup Wizard** appears.


Setup - Protegrity PEP Server

Welcome to the Protegrity PEP Server Setup Wizard

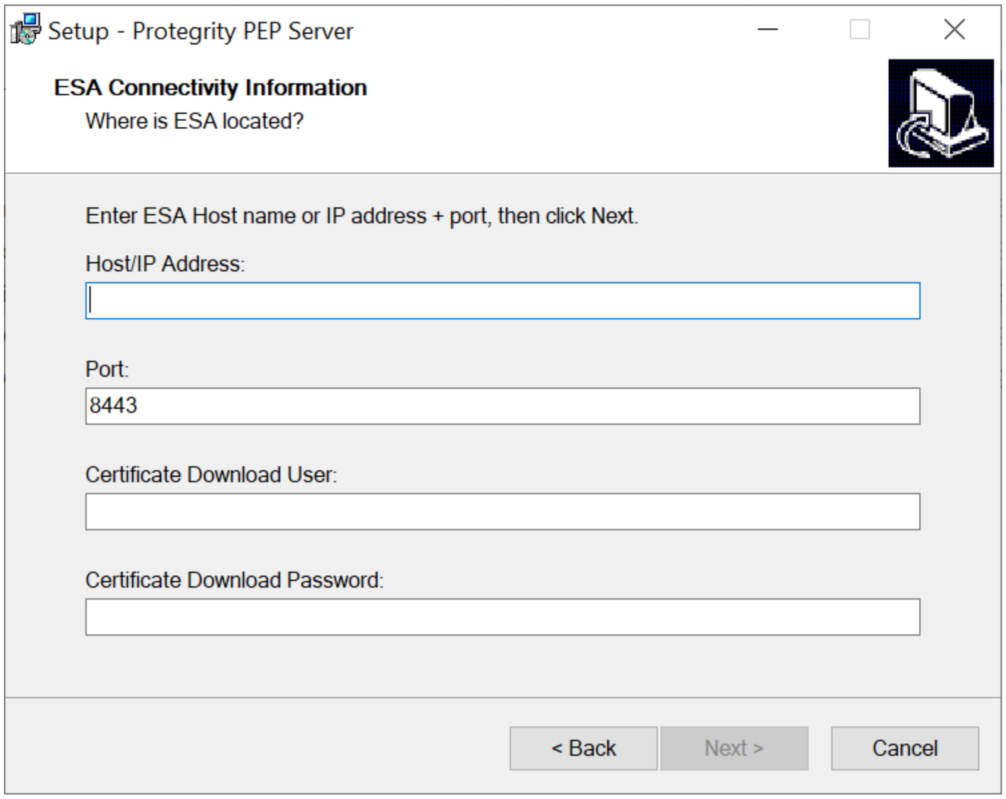
This will install Protegrity PEP Server on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

2. Click **Next**.

   The **ESA Connectivity Information** screen appears.


Setup - Protegrity PEP Server

ESA Connectivity Information

Where is ESA located?

Enter ESA Host name or IP address + port, then click Next.

Host/IP Address:

Port: 8443

Certificate Download User:

Certificate Download Password:

3. Enter the ESA Host name or IP Address in the **Host/IP Address** field.

4. Enter the ESA host listening port details in the **Port** field.

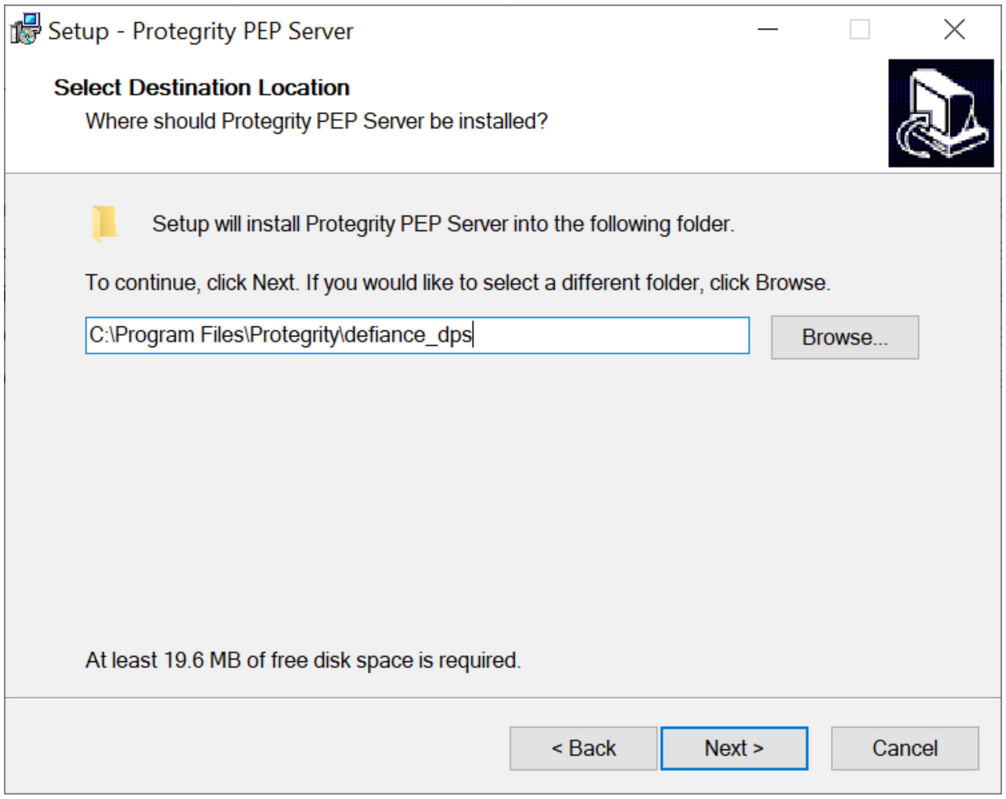
   > **Note:** Ensure that the ESA is up and running with the *HubController* service in running status to enable the automatic downloading of certificates.

5. Enter the **Certificate Download User**.

   > **Note:** It is recommended to use *admin* as the user.

6. Enter the **Certificate Download Password**.
7. Click **Next**.

   The **Select Destination Location** screen appears.



Setup - Protegrity PEP Server

Select Destination Location

Where should Protegrity PEP Server be installed?

Setup will install Protegrity PEP Server into the following folder.

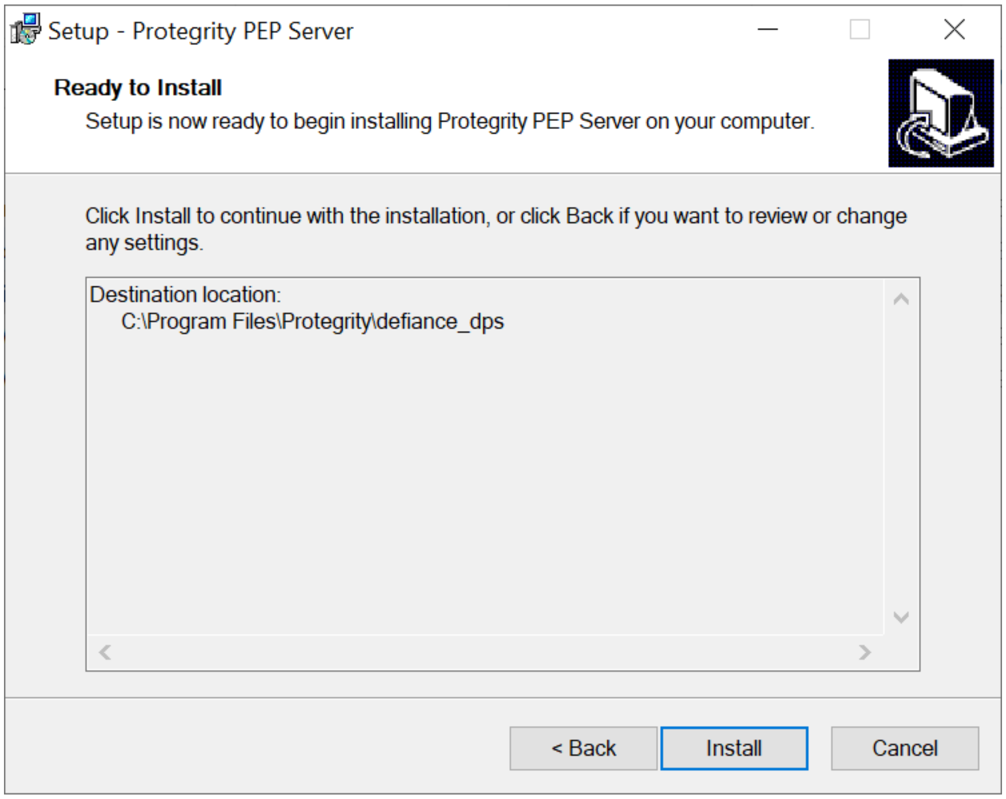
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\defiance_dps

At least 19.6 MB of free disk space is required.

8. Set the installation directory for the PEP server to `C:\Program Files\Protegrity\defiance_dps`.
9. Click **Next**.

   The **Ready to Install** screen appears.



Ready to Install

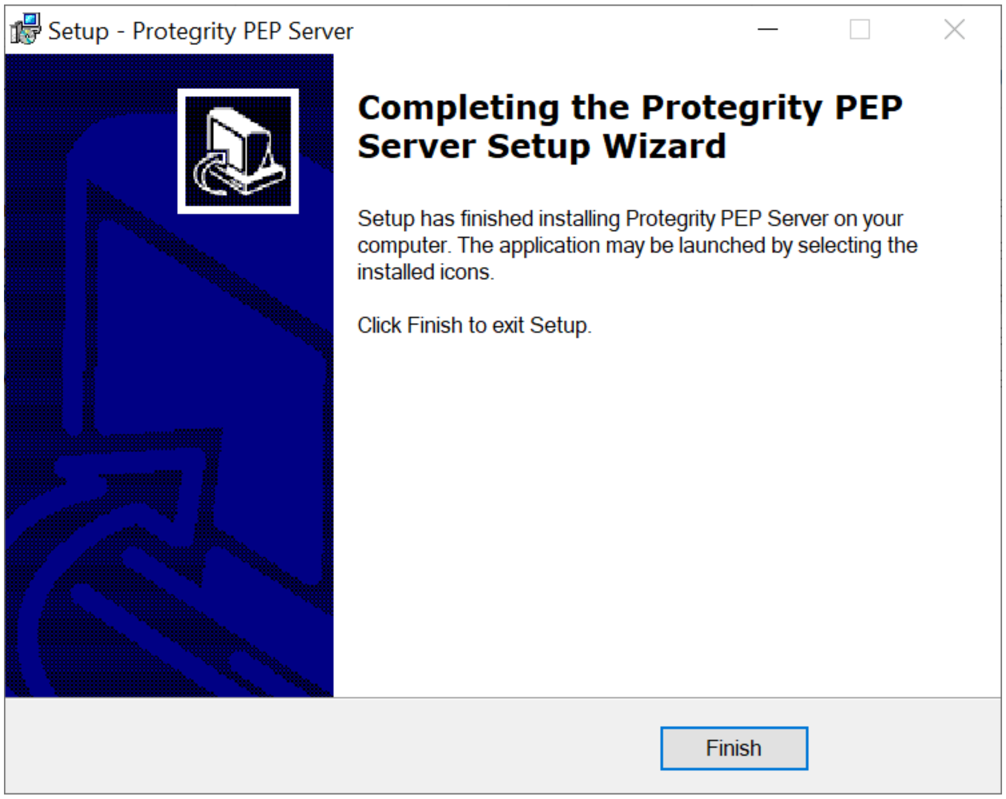
Setup is now ready to begin installing Protegrity PEP Server on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\Protegrity\defiance_dps

10. Click **Install**.

    The **Completing the Protegrity PEP Server Setup Wizard** screen appears.



Completing the Protegrity PEP Server Setup Wizard

Setup has finished installing Protegrity PEP Server on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

11. From the **Completing the Protegrity PEP Server Setup Wizard** screen, click **Finish** to complete the installation and exit.

The directories are created under the installation directory that was defined and the installation files are installed in these directories.

> **Note:**
>
> If you want to manually install the certificates to the `C:\Program Files\Protegrity\defiance_dps\data` directory of the PEP server, then navigate to the `C:\Program Files\Protegrity\defiance_dps\bin` directory, and run the following command:
>
> ***`GetCertificates -u admin <Admin Username> [-h <ESA host name or IP address>] [-p portno] [-d directory]`***
>
> This initiates secure communication between the PEP server and the ESA.
>
> Enter the password for the user *admin*.

The PEP server is installed successfully.

3.2.4.2 Using Silent Mode of Installation

This section describes how to install the PEP server on the Policy Creator Machine on a Windows platform through the Silent mode of installation.

You can also execute the PEP server installer without any manual intervention, which is also known as the Silent mode of installation. The following parameters must be provided to execute the installer in Silent mode.

Table 3-2: Parameter List for Silent Installation

Parameter	**Description**
-esa	Specifies the ESA IP address.
-esaport	Specifies the ESA port, which is optional. The default value is *8443*.
-certuser	Specifies the ESA user to download certificates.
-certpw	Specifies the ESA user password to download certificates.
-dir	Specifies the installation directory, which is optional. If the installation directory is not specified, then the installation path is the default directory, which is the `C:\Program Files\Protegrity\defiance_dps` directory.

At the command prompt, type the following command from the installer directory.

```
PepServerSetup_Windows_<bit-version>.exe -certuser <username> -esa <esaIP> -certpw <password>
```

If you want to install the PEP server in a directory other than the default directory, then you can add the *-dir* parameter to the command to specify the directory. The following command displays a sample snippet.

```
PepServerSetup_Windows_<bit-version>.exe -certuser <username> -esa <esaIP> -certpw <password>
-dir <installation-directory-path>
```

3.2.5 Creating Passphrase and Salt on the Policy Package Creator Machine and Policy Package Consuming Machine

The passphrase and the salt are used to generate a key. The *PtyShmCat* utility uses this key to encrypt the policy. This is used to implement Passphrase Based Encryption (PBE). The salt is used as an initialization vector for generating the key.

To create the passphrase and the salt:

1. Create two text files, *passphrase.txt* and *salt.txt* to store the passphrase and salt respectively.

   Important: Ensure that you use the same passphrase and salt on the Policy Package Creator Machine and the Policy Package Consuming Machine.

   Caution: This is a known security issue that data in passphrase and salt files are in clear text. You must ensure that these files are not exposed to unauthorized users. If an unauthorised user gets access to the passphrase and salt files, then they can decrypt the policy package.

   Note: Ensure that the *passphrase.txt* and *salt.txt* files do not contain a new line in it.

2. Ensure that you have installed the cryptographic and password strength package.

   For more information about installing the cryptographic package and password strength package, refer to the section *Verifying the Prerequisites*.

3. Set the following System Environment variables using the following commands.

   a. ***PTYSALT=<path of salt file>***

      For example:

      ```
      set PTYSALT=C:\Program Files\Protegrity\shmutilities\salt.txt
      ```

   b. ***PTYPASSPHRASE=<path of passphrase file>***

      For example:

      ```
      set PTYPASSPHRASE=C:\Program Files\Protegrity\shmutilities\passphrase.txt
      ```

4. Ensure that the password set in the *passphrase.txt* and *salt.txt* files must consist of the following:

   - A minimum total length of 12 characters
   - A minimum of 1 upper case letter
   - A minimum of 2 special characters
   - A minimum of 2 non-letter characters such as a digital or a special character
   - A minimum of 30 entropy bits

     Note: Entropy bits define the randomness of the password and the strength defines the complexity of the password.

   - A password that has 0.5 strength

     For more information about the password strength, refer to section *Password Strength*.

3.2.6 Running the PtyShmCat Utility on the Policy Package Creator Machine

This section describes how to run the *PtyShmCat* utility on the Policy Package Creator Machine on a Windows platform.

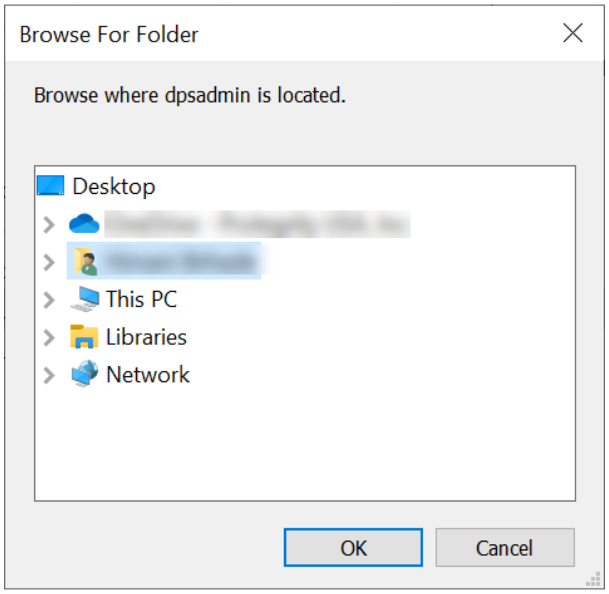
Note:

If you do not install the Log Forwarder, then ensure that the log output parameter in the *pepserver.cfg* file, which is present at `C:\Program Files\Protegrity\defiance_dps\data`, is set to *stdout* before creating the policy package.

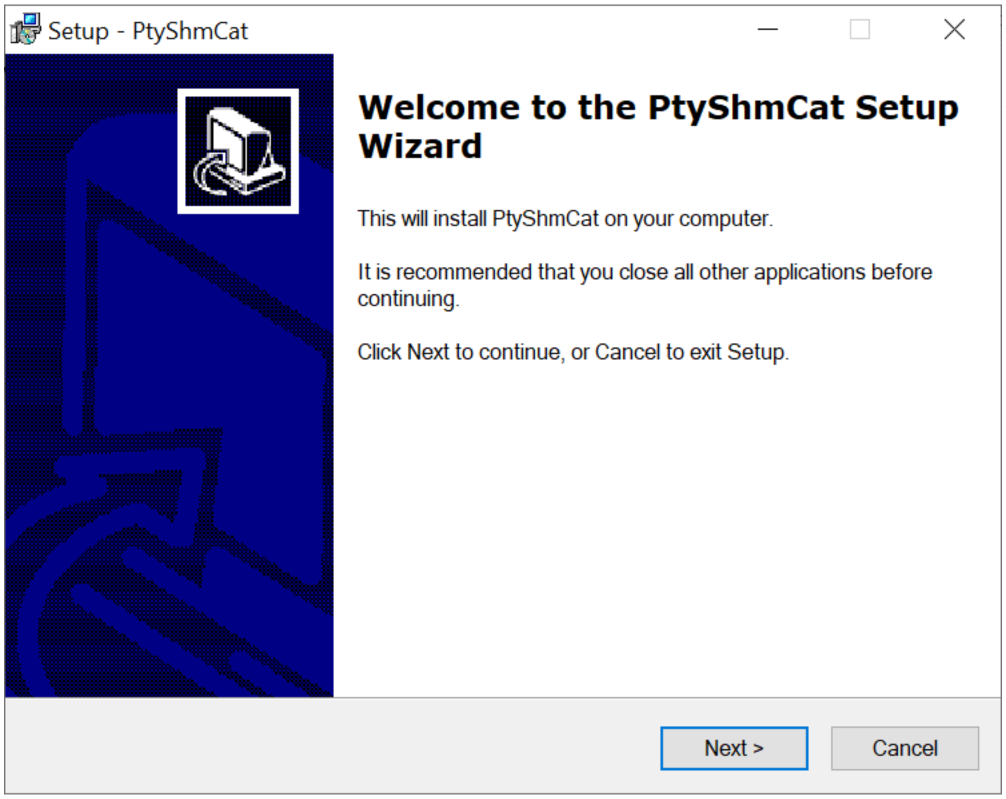
Note:
You must restart the PEP server from the **Services** after updating the configurations in the *pepserver.cfg* file for the changes to take effect.

To run the *PtyShmCat* utility on the Windows platform:

1. Install the *PtyShmCat* utility by running the following setup file.
   `PtyShmCatSetup_Windows_<bit-version>_<version>.exe`
   The **Browse For Folder** screen appears.



2. Select the path where the dpsadmin is located.
3. Click **OK**.
   The **Welcome to the PtyShmCat Setup Wizard** screen appears.



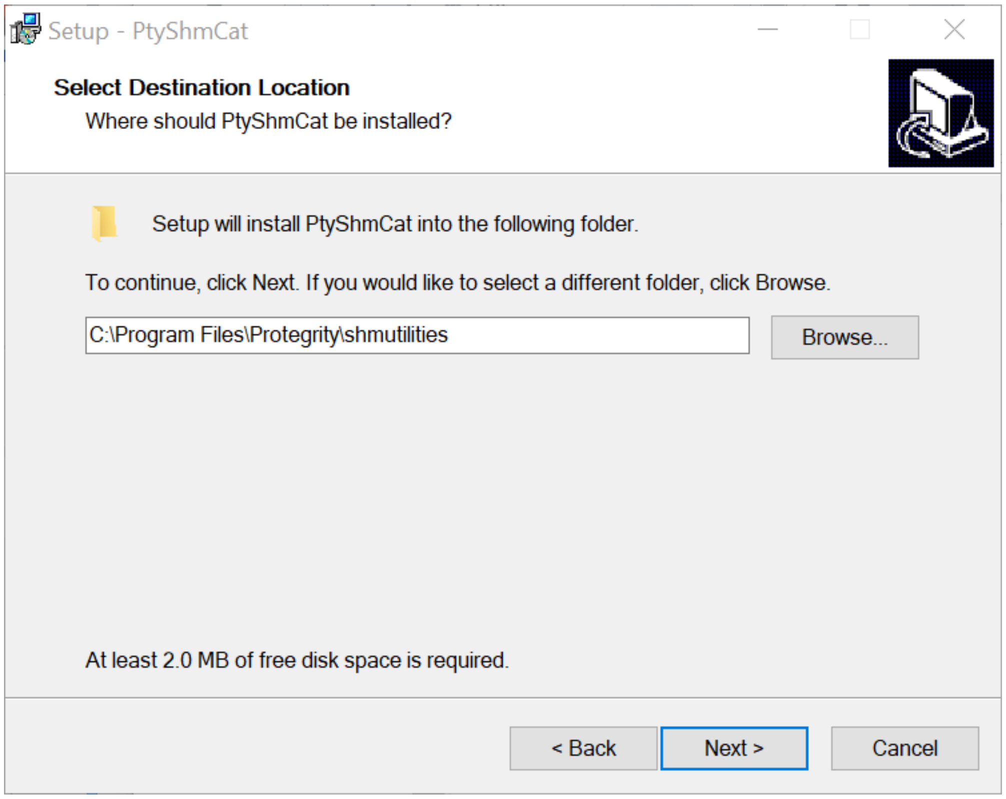
Welcome to the PtyShmCat Setup Wizard

This will install PtyShmCat on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

4. Click **Next**.
   The **Select Destination Location** screen appears.



Select Destination Location

Where should PtyShmCat be installed?

Setup will install PtyShmCat into the following folder.

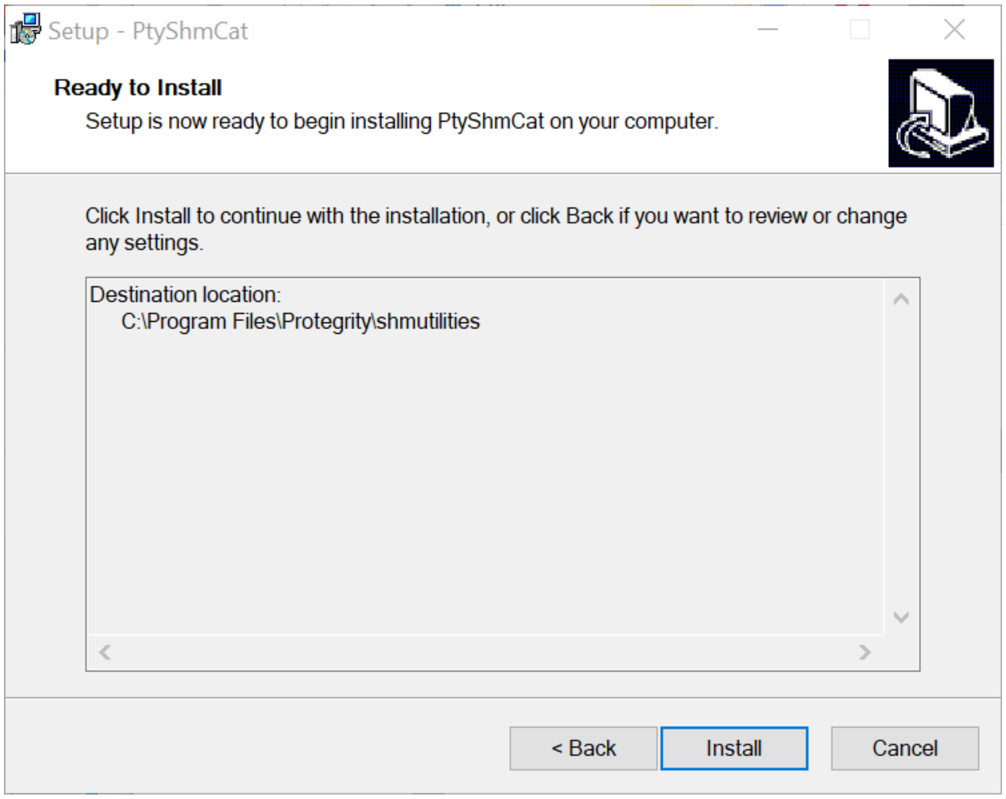
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\shmutilities

At least 2.0 MB of free disk space is required.

5. Select the installation directory for the *PtyShmCat* utility to `C:\Program Files\Protegrity\shmutilities`.
6. Click **Next**.

The **Ready to Install** screen appears.



Ready to Install

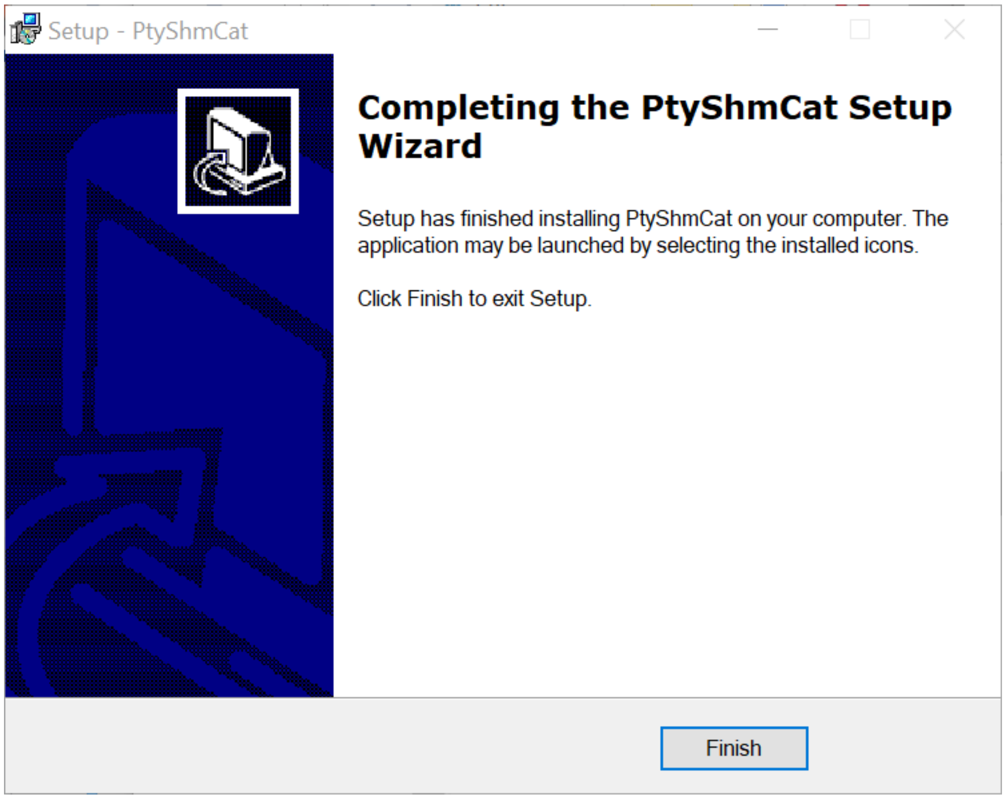
Setup is now ready to begin installing PtyShmCat on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\Protegrity\shmutilities

7. Click **Install**.

The **Completing the PtyShmCat Setup Wizard** screen appears.



Completing the PtyShmCat Setup Wizard

Setup has finished installing PtyShmCat on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

8. From the **Completing the PtyShmCat Setup Wizard** screen, click **Finish** to complete the installation and exit.

The directories are created under the installation directory that was defined, and the installation files are installed in these directories.

9. Run the *imp_creator.py* script by using the following command.

```
python imp_creator.py <policy package name> --local_path="<path to store policy package>"
```

An encrypted policy package is generated at the local path defined by the user.

Note:
Ensure that you have copied the encrypted policy package to the Policy Package Consuming Machine.

The *PtyShmCat* utility is installed successfully.

3.2.7 Running the PtyShmPut Utility on the Policy Package Consuming Machine

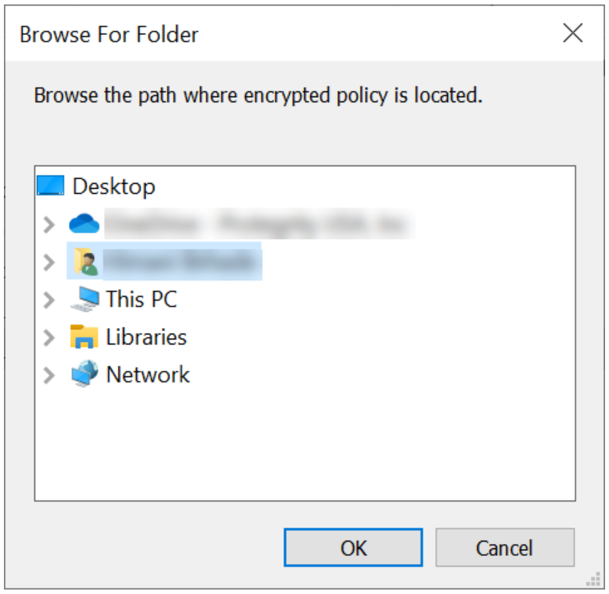
This section describes how to run the *PtyShmPut* utility on the Policy Package Consuming Machine on a Windows platform.

To run the *PtyShmPut* utility on the Windows platform:

1. Install the *PtyShmPut* utility by running the following setup file.

```
PtyShmPutSetup_Windows_<bit-version>_<version>.exe
```

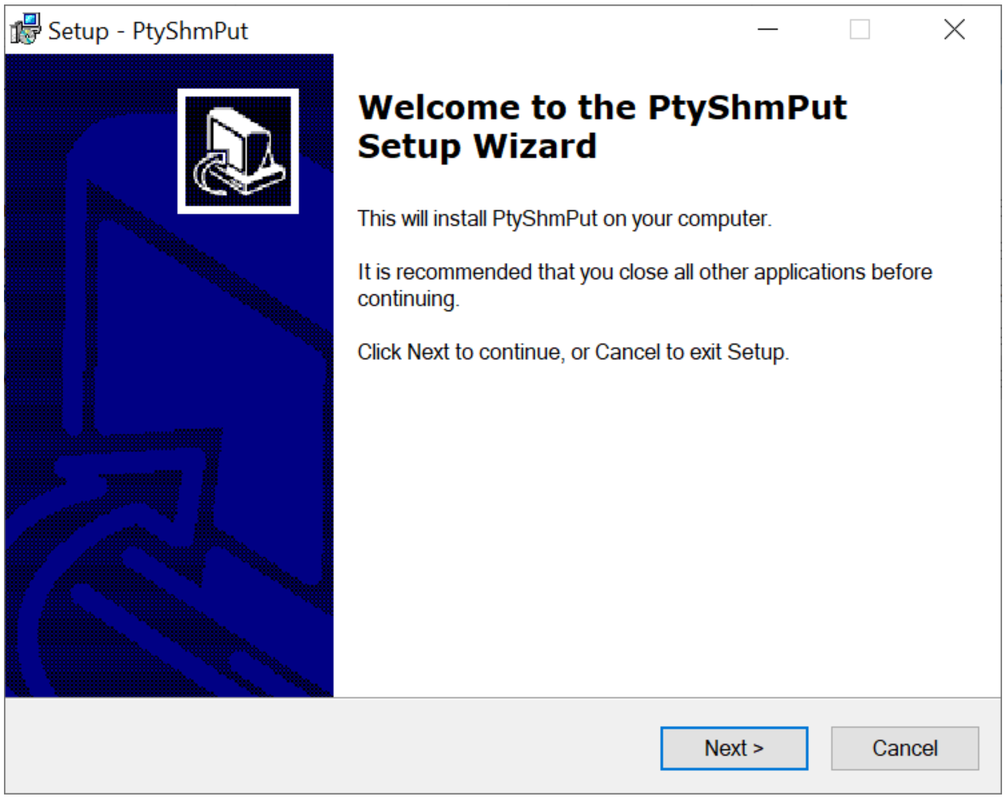
The **Browse For Folder** screen appears.


Browse For Folder

Browse the path where encrypted policy is located.

2. Select the path where the encrypted policy path is located.
3. Click **OK**.

The **Welcome to the PtyShmPut Setup Wizard** screen appears.



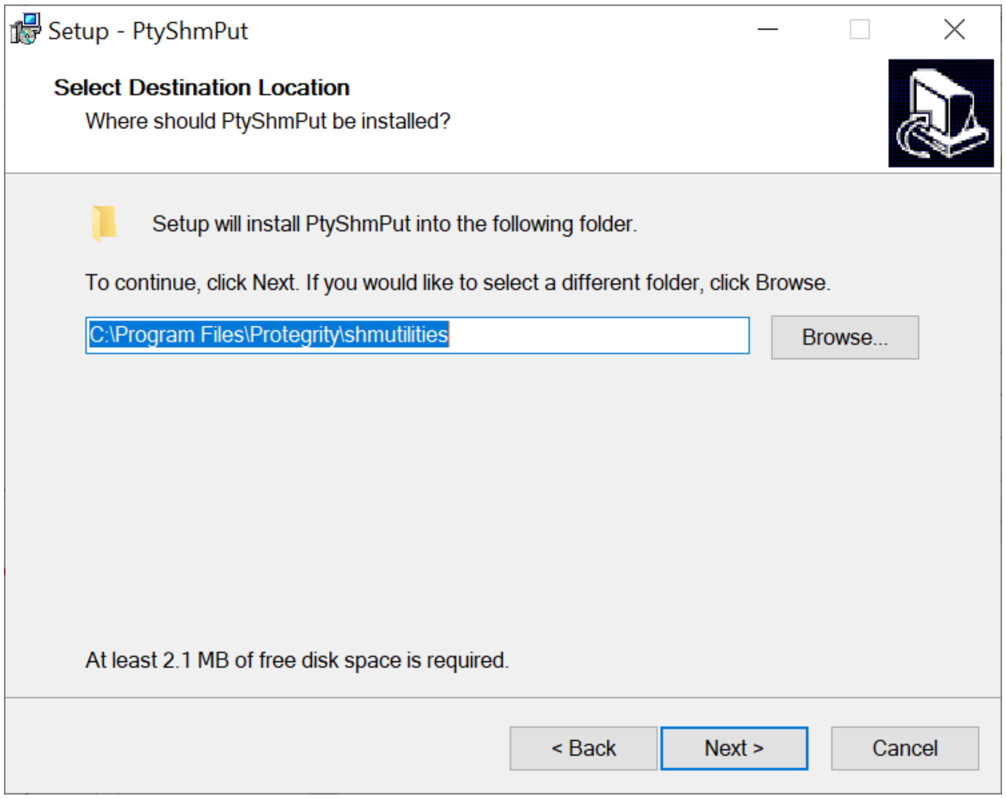
Welcome to the PtyShmPut Setup Wizard

This will install PtyShmPut on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

4. Click **Next**.
   The **Select Destination Location** screen appears.



Select Destination Location
Where should PtyShmPut be installed?

Setup will install PtyShmPut into the following folder.

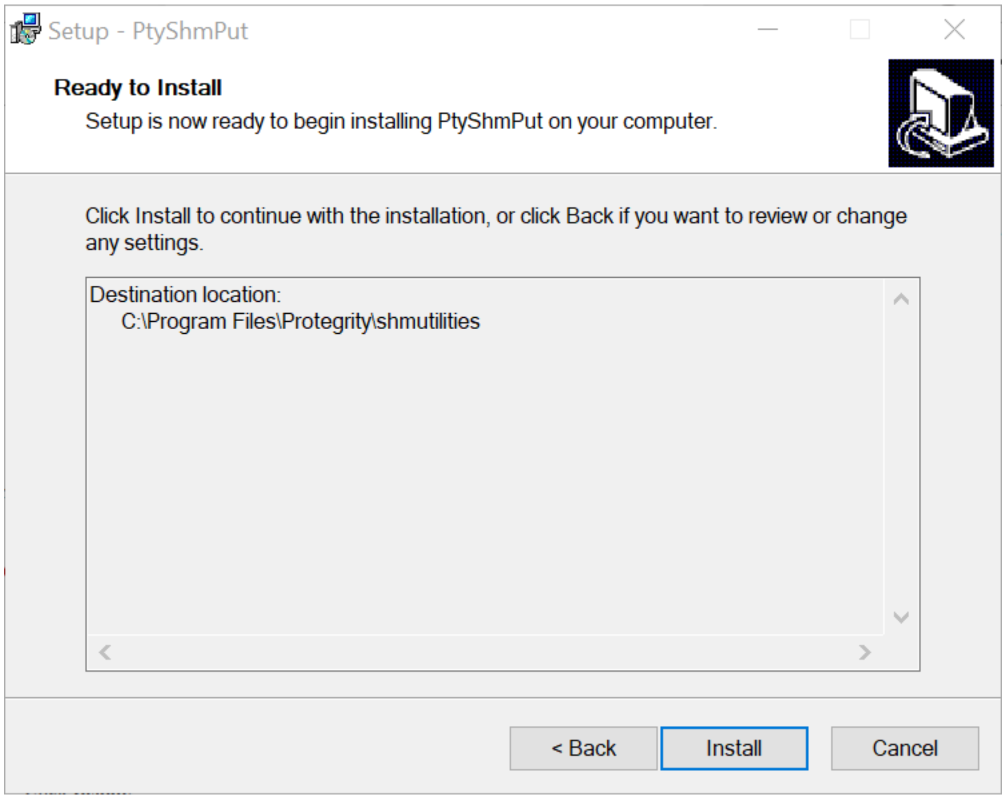
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\shmutilities

At least 2.1 MB of free disk space is required.

5. Select the location where the *PtyShmPut* utility needs to be installed.
6. Click **Next**.

The **Ready to Install** screen appears.



Ready to Install

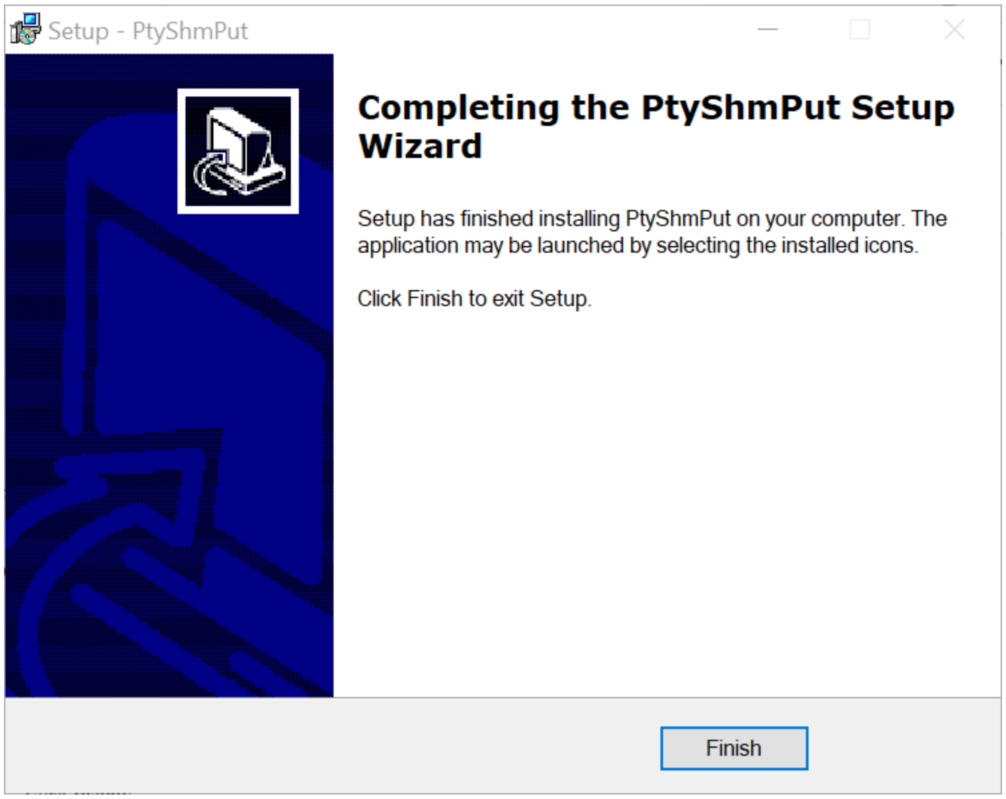
Setup is now ready to begin installing PtyShmPut on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\Protegrity\shmutilities

7. Click **Install**.

   The **Completing the PtyShmPut Setup Wizard** screen appears.



Completing the PtyShmPut Setup Wizard

Setup has finished installing PtyShmPut on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

8. From the **Completing the PtyShmPut Setup Wizard** screen, click **Finish** to complete the installation and exit.

The directories are created under the installation directory that was defined, and the installation files are installed in these directories.

9. Ensure that you have copied the encrypted policy package created in the Policy Package Creator Machine to the Policy Package Consuming Machine.
10. Navigate to the `C:\Program Files\Protegrity\shmutilities\scripts` directory on the Policy Package Consuming Machine.
11. Run the *imp_updater.py* script by using the following command.

    Note: Ensure that you run the command prompt as an administrator.

    ```
    python imp_updater.py <policy package name>
    ```

    A decrypted policy package is created.

    Note:
    If you are using same machine for installing *PtyShmPut* utility, then you need to uninstall the PEP server setup.

    Note:
    You can reconfigure the System Environment variable *PTY_SHM_POLICY_PATH* created for the policy path. However, you need to run the *imp_updater.py* utility again for the changes to take effect.

    Note:
    Ensure that you do not start the *PtyShmPut* service manually as it starts automatically during the execution of the*imp_updater.py* utility.

    If for any reason the *PtyShmPut* service is stuck at the **starting** state, then navigate to the **Task Manager** and click on the **End Task** option to end the *PtyShmPut* service manually.

The *PtyShmPut* utility is installed successfully.

3.2.8 Verifying the Policy Health on the Policy Package Consuming Machine

This section describes how to verify the policy health on a Windows platform.

To verify the policy health on the Policy Package Consuming Machine on the Windows platform:

1. On the policy consuming machine, navigate to the `C:\Program Files\Protegrity\shmutilities\bin` directory.
2. To verify whether the data protection policies are published successfully, run the following script.

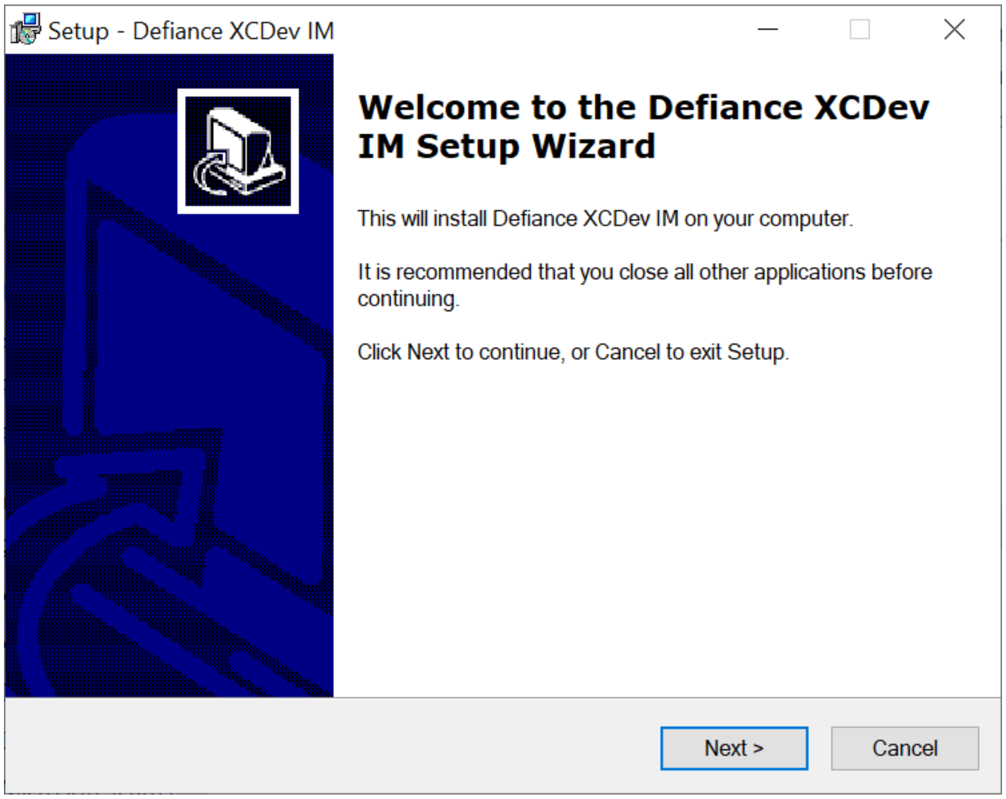
    ```
    PTYPolicyHealthCheck.bat
    ```

3.2.9 Installing Immutable Application Protector C on the Policy Consuming Machine

This section describes how to install the IAP C on the policy consuming machine on a Windows platform.

To install the IAP C on the Windows platform:

1. Run the following setup file to install the IAP C.
   XCDevIMSetup_Windows_<bit-version>_<version>.exe
   The **Welcome to the Defiance XCDev IM Setup Wizard** appears.


Setup - Defiance XCDev IM

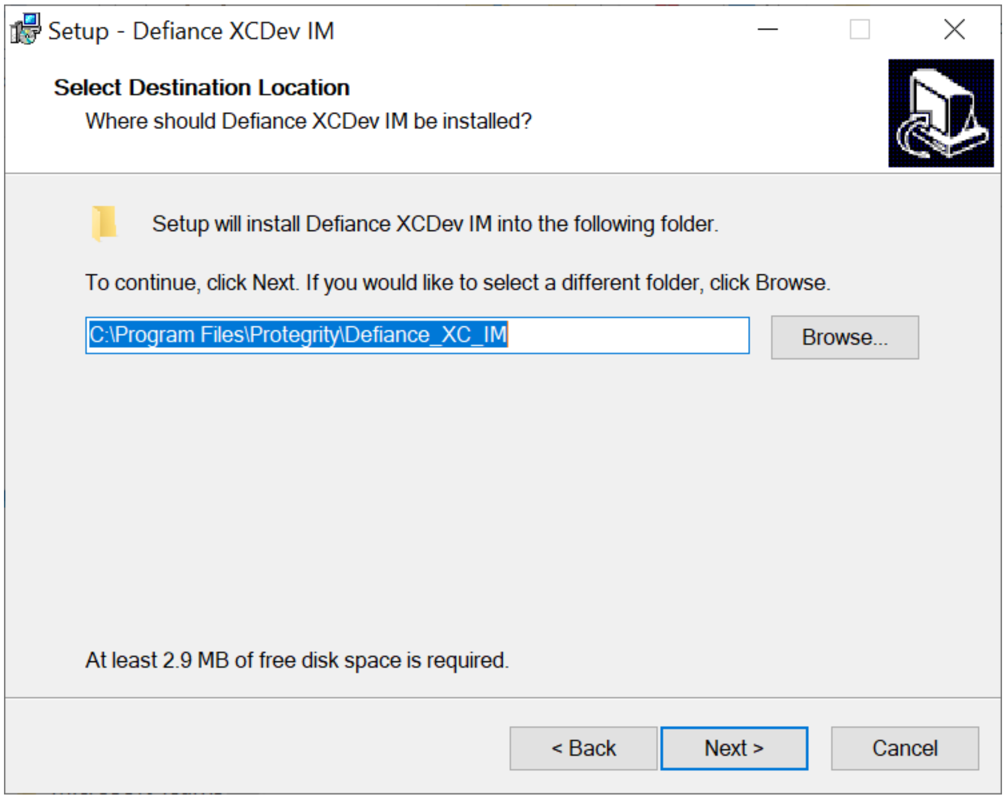
Welcome to the Defiance XCDev IM Setup Wizard

This will install Defiance XCDev IM on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

2. Click **Next**.
   The **Select Destination Location** screen appears.


Setup - Defiance XCDev IM

Select Destination Location

Where should Defiance XCDev IM be installed?

Setup will install Defiance XCDev IM into the following folder.

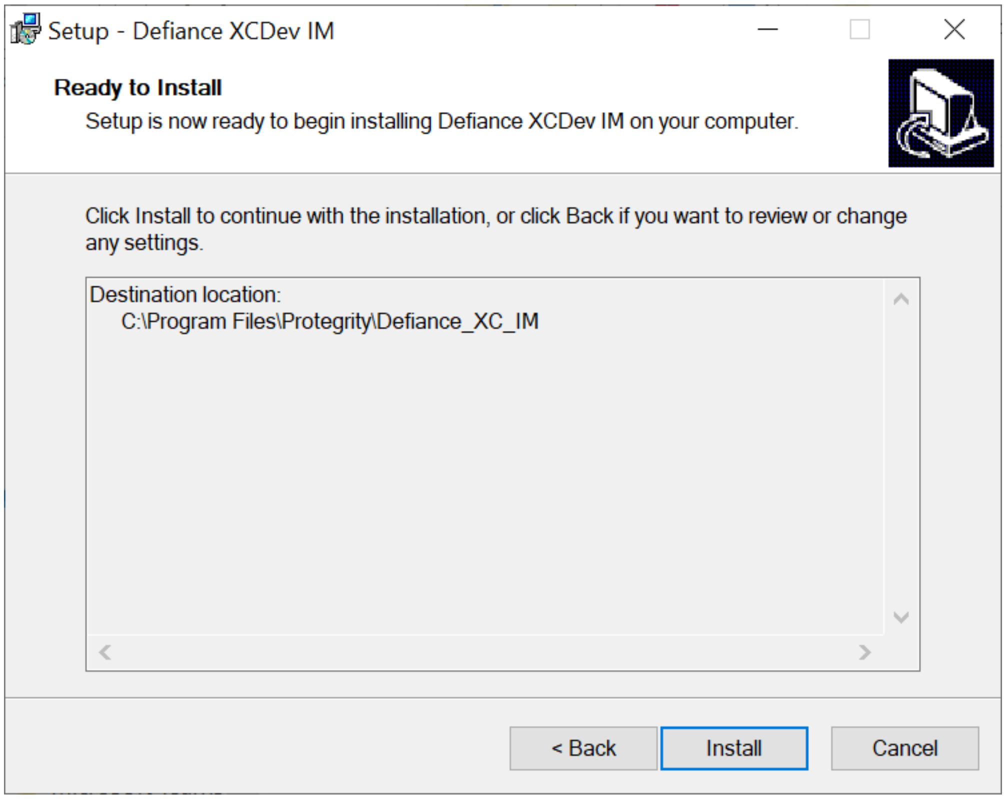
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\Defiance_XC_IM

At least 2.9 MB of free disk space is required.

3. Click **Next**.

   The **Ready to Install** screen appears.


Setup - Defiance XCDev IM

Ready to Install

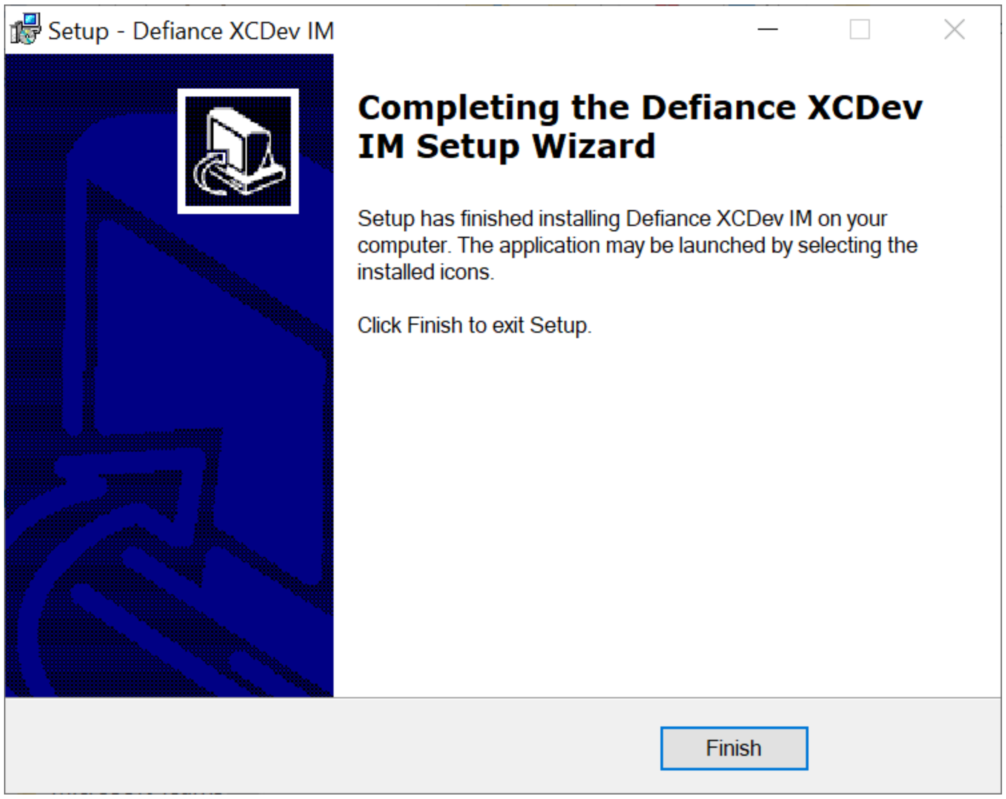
Setup is now ready to begin installing Defiance XCDev IM on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
    C:\Program Files\Protegrity\Defiance_XC_IM

4. Click **Install**.

   The **Completing the Defiance XCDev IM Setup Wizard** screen appears.



Completing the Defiance XCDev IM Setup Wizard

Setup has finished installing Defiance XCDev IM on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

5. From the **Completing the Defiance XCDev IM Setup Wizard** screen, click **Finish** to complete the installation and exit.

   The directories are created under the installation directory that was defined and the installation files are installed in these directories.

   Note:
   The *xcpep_imp.plm* library is installed in the `C:\Program Files\Protegrity\Defiance_XC_IM\bin` directory.

   The header files will be installed in the `C:\Program Files\Protegrity\Defiance_XC_IM\include` directory.

The IAP C is installed successfully.

3.2.10 Uninstalling the Log Forwarder from the Windows platform

This section describes how to uninstall the Log Forwarder from a Windows platform.

➤ To uninstall the Log Forwarder from the Windows platform:

1. Perform the following steps to stop the Log Forwarder.
   a. From the Windows Start Menu, search and select **Services**.
   b. Navigate to the **Logforwarder** directory.
   c. Right-click the **Logforwarder** service and click **Stop**.
2. Run the uninstall utility for Log Forwarder located in the `C:\Program Files\Protegrity\fluent-bit` directory.
3. Delete the `C:\Program Files\Protegrity\fluent-bit` directory.

The Log Forwarder is uninstalled.

3.2.11 Uninstalling the PEP Server from Windows

This section describes how to uninstall the PEP server from a Windows platform.

➤ To uninstall the PEP server from a Windows platform:

1. Perform the following steps to stop the PEP server.
   a. From the Windows Start Menu, search and select **Services**.
   b. Navigate to the **Protegrity PEP Server** directory.
   c. Right-click the **Protegrity PEP Server** service and click **Stop**.
2. Run the uninstall utility for the PEP server located in the *`C:\Program Files\Protegrity\defiance_dps`* directory.
3. Delete the *`C:\Program Files\Protegrity\defiance_dps`* directory.

The PEP server is uninstalled.

3.2.12 Uninstalling PtyShmCat and PtyShmPut Utilities on Windows

This section describes how to uninstall the PtyShmCat and PtyShmPut utilities from the Windows platform.

➤ To uninstall the PtyShmCat and PtyShmPut utilities from the Windows platform:

1. Navigate to the *`C:\Program Files\Protegrity\shmutilities`* directory.
2. Run the following file to uninstall the PtyShmCat and PtyShmPut utilities.
   ```
   unins000.exe
   ```
3. Delete the following directory.
   `C:\Program Files\Protegrity\shmutilities`

The PtyShmCat and PtyShmPut utilities are uninstalled.

3.2.13 Uninstalling Immutable Application Protector C from Windows

This section describes how to uninstall the IAP C from the Windows platform.

➤ To uninstall the IAP C from the Windows platform:

1. Navigate to the *`C:\Program Files\Protegrity\Defiance_XC_IM`* directory.
2. Run the following file to uninstall the IAP C.

`unins000.exe`

3. Delete the following directory.
   `C:\Program Files\Protegrity\Defiance_XC_IM`

The IAP C is uninstalled.

3.2.14 Running IAP C - Example

This section provides examples on how to use the IAP C protect, unprotect, and reprotect APIs. It is assumed that policies are made available to the Policy Consuming Machine in a decrypted form and the IAP C has been successfully installed.

For more information on running the IAP C sample application, refer to the section *Running IAP - Example*.

Chapter 4

Installing Immutable Application Protector (IAP) Java

The Immutable Application Protector Java provides APIs for performing security operations.

All standard protection techniques offered by Protegrity are applicable to the Immutable Application Protector Java.

4.1 Setting up Immutable Application Protector (IAP) Java on Linux

This section describes how to set up the IAP Java on a Linux or Unix platform.

4.1.1 Verifying the Prerequisites

This section describes the prerequisites for installing the IAP Java on a Linux or Unix platform.

Before you begin
Ensure that the following prerequisites are met before installing the IAP Java on a Linux or Unix platform:

- Glibc, version 2.4 or later, is installed.
- GCC, version 3.2.3 or later, is installed.
- The ESA is installed, configured, and running.
- The IP address or host name of the ESA is noted.
- Java, version 1.8 or later, is installed.

4.1.2 Downloading the Immutable Application Protector Java Installation Package

This section describes how to download the IAP Java on a Linux or Unix platform.

To download the IAP Java on the Linux or Unix platform:

1. Download the *IAP_Linux-ALL-<bit-version>_x86-64_JRE-1.8-<bit-version>_<version>.tgz* installation package to any location on the machine where you want to install the protector.
2. Create a directory on your machine where you want to install the protector.
3. Extract the IAP Java installation package using the following command.

   ```
   tar -xvf IAP_Linux-ALL-<bit-version>_x86-64_JRE-1.8-<bit-version>_<version>.tgz
   ```

   The following setup files are extracted:

- *PolicyUtilSetup_Linux_x64_<version>.sh*
- *APJavaIMSetup_Linux_x64_<bit_version>_<version>.sh*
- *manifest.json*

4.1.3 Installing the Immutable Application Protector Java on Policy Consuming Machine

This section describes how to install the IAP Java on a Linux or Unix platform.

To install the IAP Java on the Linux or Unix platform:

1. Run the following script to install the Immutable Application Protector Java.

   ```
   ./APJavaIMSetup_Linux_<bit-version>_<version>.sh
   ```

   The library directory path for the IAP Java is the `<installation_directory>/applicationprotector/java/lib` directory.
2. Check that the following libraries are installed in the IAP Java installation directory:
   - *ApplicationProtectorJava.jar*
   - *ApplicationProtectorJava.properties*
   - *jna-5.8.0.jar*
   - *jna-platform-5.8.0.jar*
   - *jpeplite_imp.plm*
3. Link your application to the *ApplicationProtectorJava.properties* and *.jar* files listed in *step 2*.

   The application should be linked with the required path variables available in the required OS.

4.1.4 Running the REST Client utility on the Policy Package Creator Machine

This section describes how to run the cURL utility on the Policy Package Machine.

To run the cURL utility:

1. Run the following command to know the PEP version supported by the ESA:

   `curl -k -v -u "<user>:<password>" -X GET https://<ESA_IP_ADDRESS>/pty/v1/imp/version`

   Sample output:

   ```
   {
     "version" : "<version>",
     "buildVersion" : "<build version>",
     "pepVersions" : [ "<pepVersions supported by the ESA>" ]
   }
   ```

   Note: Ensure that the password complexity meets OWASP recommendations. For more information about the password requirements, refer to section *Password Strength*.
2. Run the following command to fetch the policy dump from the ESA:

   ```
   curl -k -v --location 'https: //<ESA_IP>/pty/v1/imp/export' \
   -u "<USER>:<PASSWORD>" \
   --header 'Content-Type: application/json' \
   ```

```
--data '{
    "name": "<Request_Name>",
    "dataStoreName": "<DataStoreName>",
    "pepVersion": "<pepVersion>",
    "emptyString":"NULL",
    "caseSensitive":"YES",
    "output":"STDOUT",
    "kek": {
        "pkcs5": {
            "password": "<PASSWORD>",
            "salt": "<SALT>"
        }
    }
}' -o <Policy_Package_Name>.json
```

Note: Ensure that the output parameter is set to *STDOUT* to avoid the loss of logs.

Note: Ensure that the value passed for the *pepVersion* parameter in the command matches with the *pepVersion* parameter in the *manifest.json* file.

Note: The values passed for the parameters in the command must match the corresponding values in the ESA.

Note: It is recommended that the *<USER>* exporting the policy should have minimum privileges. The least privilege that is required is the *Export IMP* permissions on the ESA.

An encrypted policy package file is generated at the local path defined by the user.

3. Ensure that you have copied the encrypted policy package to the Policy Package Consuming Machine.

4.1.5 Running the PolicyUtil Binary on the Policy Consuming Machine

This section describes how to run the *import* utility on the Policy Package Consuming Machine.

To run the *import* utility:

1. Run the following command to install the PolicyUtil setup, which has the *impgen2* binary.

```
./PolicyUtilSetup_Linux_x64_<version>.sh
```

2. The following directory is created in the `/opt/protegrity` directory.

   `/opt/protegrity/imp-policy-util`

3. Navigate to the `/opt/protegrity/imp-policy-util/bin/impgen2` directory available on the Policy Package Consuming Machine.
4. Run the *impgen2* binary by using the following command.

```
./impgen2 import -passphrase <password> -salt <salt> -policy-file <file_path>/
<policy_name>.json -pepversion <pepversion>
```

Note: The IMP package PEP version must match the protector PEP version.

Note:

Passphrase and salt can either be given as clear texts or as environment variables.

> For example: *PTY_SALT* or *PTY_SALT_PATH*.
>
> If a file is used for storing passphrase and salt, then ensure that there is no new line in it. For more information regarding the binary, run `./impgen2 help`.

The *PolicyUtil* binary repopulates the shared memory on the Policy Package Consuming Machine.

5. Run the following command to view the populated shared memory on the Policy Package Consuming Machine.

```
ipcs -a
```

After the shared memory is populated, the protector can be used to perform the protect, unprotect, and reprotect operations.

4.1.6 Verifying the Policy Health on the Policy Package Consuming Machine

This section describes how to verify the policy health on the Policy Package Consuming Machine.

To verify the policy health:

1. Navigate to the `/opt/protegrity/imp-policy-util/bin/` directory.
2. Run the following command to verify the policy health.

```
./impgen2 health
```

The following output is expected.

```
health check status: healthy
```

4.1.7 Uninstalling PolicyUtil Binary from Policy Package Consuming Machine

This section describes how to uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine.

To uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine:

1. Navigate to the `/opt/protegrity/imp-policy-util` directory.
2. Remove the `/imp-policy-util` directory.

4.1.8 Uninstalling Immutable Application Protector Java from Linux or Unix

This section describes how to uninstall the IAP Java from a Linux or Unix platform.

To uninstall the IAP Java from a Linux or Unix platform:

1. Navigate to the `/opt/protegrity/applicationprotector/java` directory.
2. Delete the `/java` directory.

The IAP Java is uninstalled.

4.1.9 Running IAP Java - Example

This section provides examples on how to use the IAP Java protect, unprotect, and reprotect APIs. It is assumed that policies are made available to the Policy Package Consuming Machine in a decrypted form and the IAP Java has been successfully installed.

For more information on running the IAP Java sample application, refer to the section *Running IAP - Example*.

4.2 Setting up Immutable Application Protector Java on Windows

This section describes how to set up the IAP Java on a Windows platform.

4.2.1 Verifying the Prerequisites

Ensure that the following prerequisites are met:

- The Windows operating system, 64-bit is installed, configured, and running
- The ESA is installed, configured, and running
- The IP address or host name of the ESA is noted
- Java, version 1.8 or later, is installed
- Python, version 3.10.0 or later, is installed
- Install the cryptography package using the following command.

  ```
  pip install cryptography
  ```

- Install the password strength package using the following command.

  ```
  pip install password-strength
  ```

4.2.2 Downloading the Immutable Application Protector (IAP) Java Installation Package

This section describes how to download the IAP Java installation package on the Windows platform.

To download the IAP Java installation package on the Windows platform:

1. Create a directory on your machine where you want to install the protector.
2. Download the *IAP_WIN-ALL-64_x86-64_JRE-1.8-64_<version>.zip* installation package.
3. Extract the files from the *IAP_WIN-ALL-64_x86-64_JRE-1.8-64_<version>.zip* installation package.

   The following files are extracted:

   - *APJavaIMSetup_Windows_x64_<version>.exe*
   - *LogforwarderSetup_Windows_x64_<version>.exe*
   - *PepServerSetup_Windows_x64_<version>.exe*
   - *PtyShmCatSetup_Windows_x64_<version>.exe*
   - *PtyShmPutSetup_Windows_x64_<version>.exe*

4.2.3 Installing Log Forwarder on the Policy Package Consuming Machine

This section describes how to install the Log Forwarder on the Policy Package Consuming Machine on a Windows platform using the Windows Wizard and through the Silent mode of installation.

Note:
Installation of the Log Forwarder component is optional for the IAP on 64-bit Windows deployments.

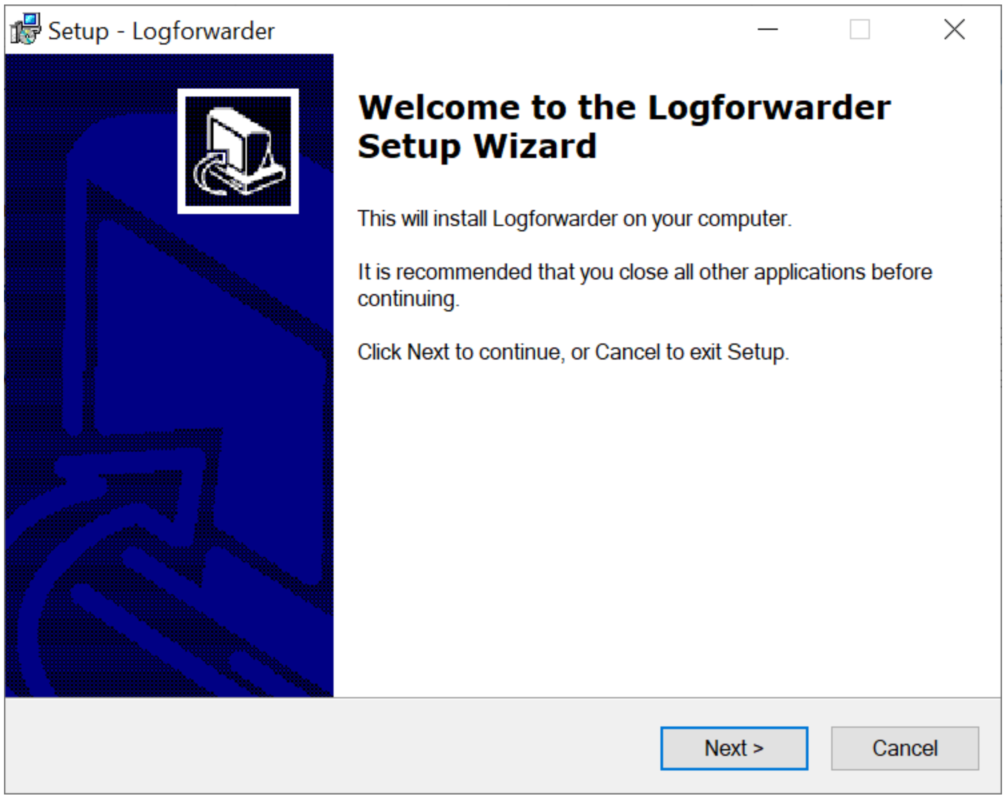
If you want to forward the logs to a Security Information and Event Management (SIEM), then install the Log Forwarder on the Policy Package Consuming Machine.

4.2.3.1 Using Windows Wizard

This section describes how to install the Log Forwarder on the Policy Package Consuming Machine on a Windows platform using the Windows Wizard.

To install the Log Forwarder on the Policy Package Consuming Machine using the Windows Wizard:

1. Run the *LogforwarderSetup_Windows_x64_<version>.exe* installer from the created directory.
   The **Welcome to the Log Forwarder Setup Wizard** screen appears.



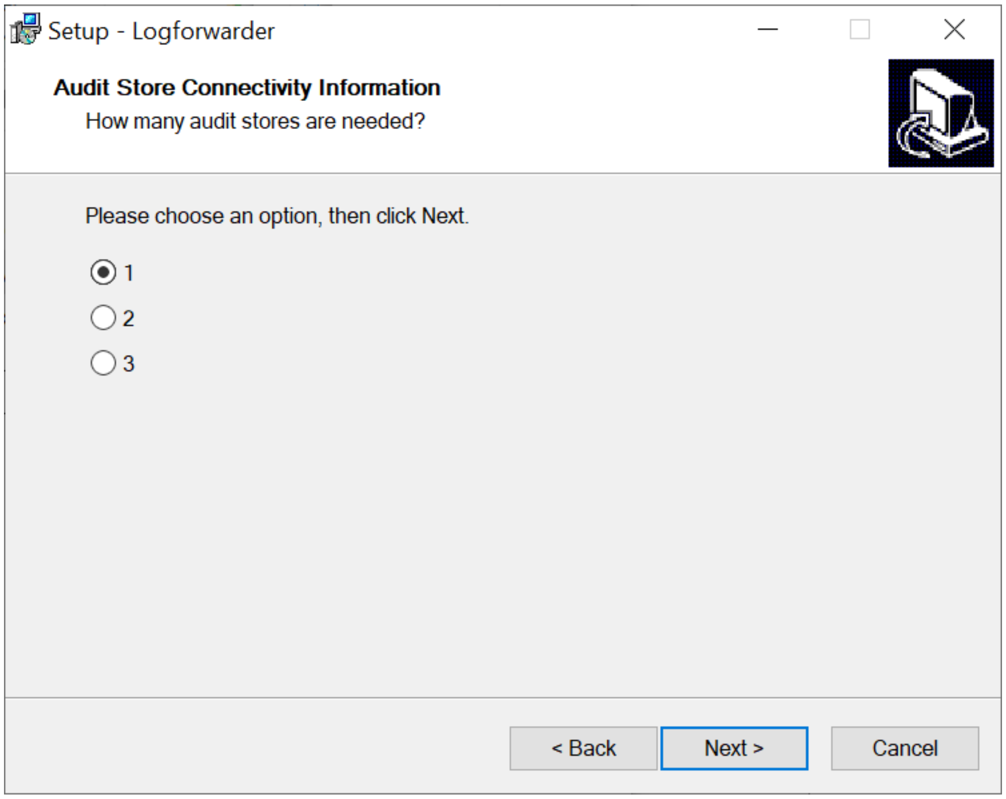
Welcome to the Logforwarder Setup Wizard

This will install Logforwarder on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

2. Click **Next**.
   The **Audit Store Connectivity Information** screen appears to select the number of audit stores that are needed.


Setup - Logforwarder

Audit Store Connectivity Information

How many audit stores are needed?

Please choose an option, then click Next.

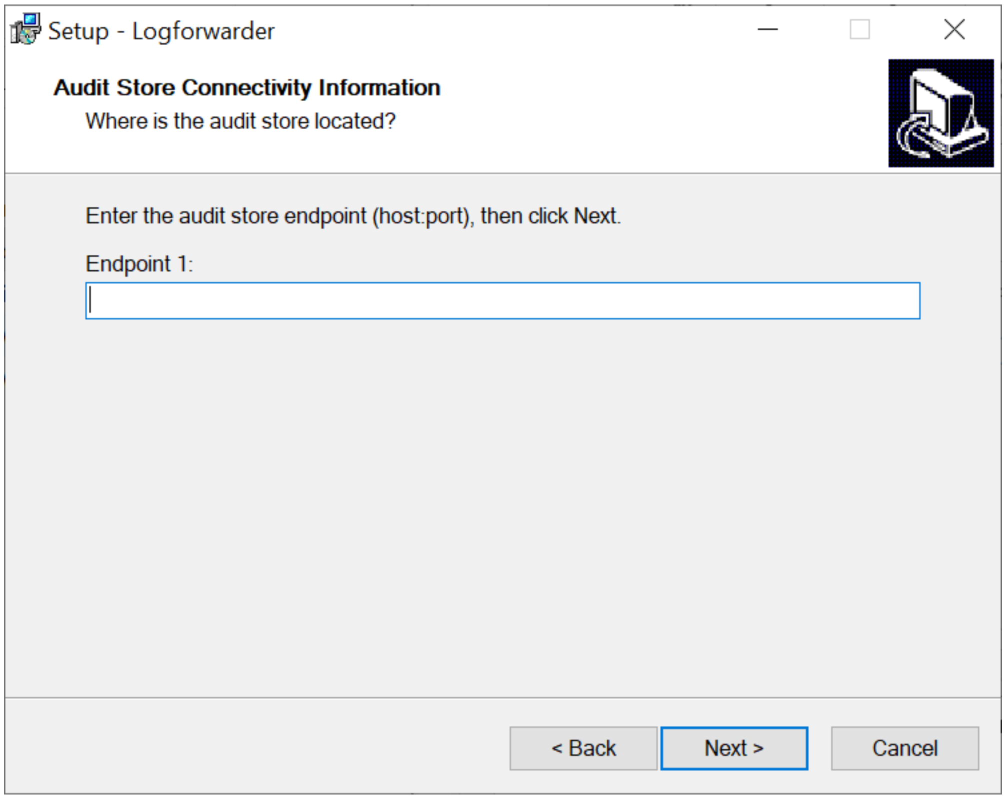
[✓ 1] 2 3

3. Select the number of Audit Stores as required.

   The maximum number of Audit Stores that can be configured is 3.

4. Click **Next**.

   The **Audit Connectivity Information** screen appears to enter the location of the audit store.


Setup - Logforwarder

Audit Store Connectivity Information

Where is the audit store located?

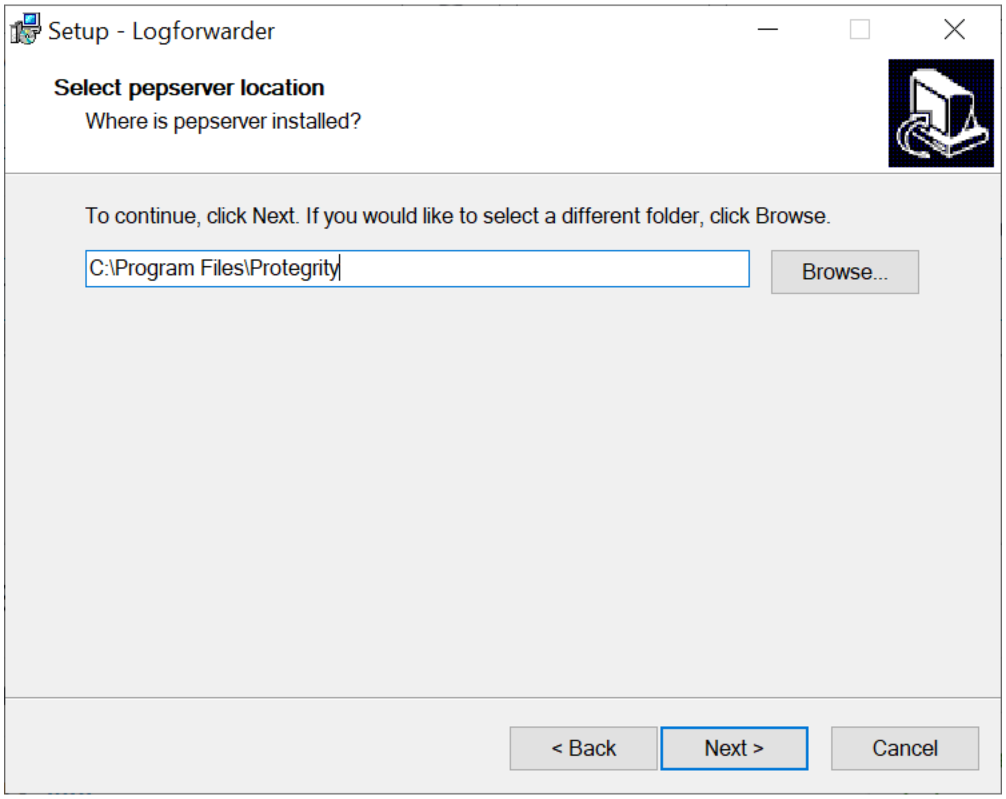
Enter the audit store endpoint (host:port), then click Next.

Endpoint 1:

5. Enter the Audit Store endpoint as *Hostname/IPAddress:port* in the **Endpoint 1** field.

   Note:
   The default port number for the Audit Store is *9200*.

6. Click **Next**.
   The **Select pepserver location** screen appears.



Setup - Logforwarder

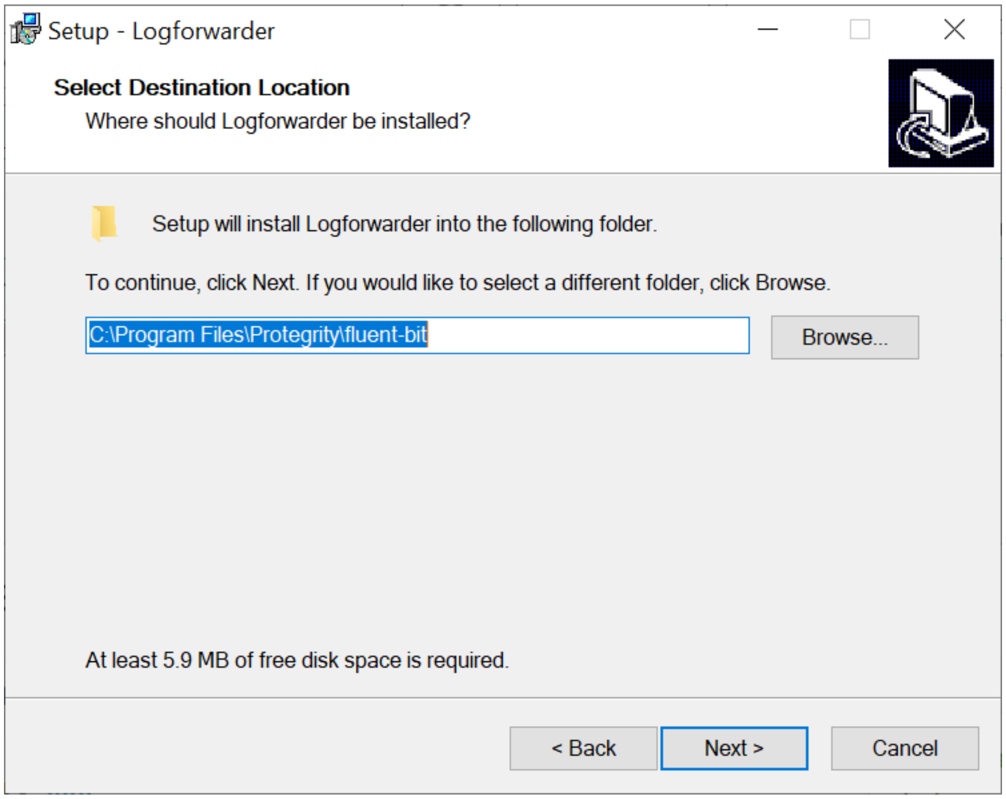
Select pepserver location

Where is pepserver installed?

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity

7. Click **Next**.
   The **Select Destination Location** screen appears.



Setup - Logforwarder

Select Destination Location

Where should Logforwarder be installed?

Setup will install Logforwarder into the following folder.

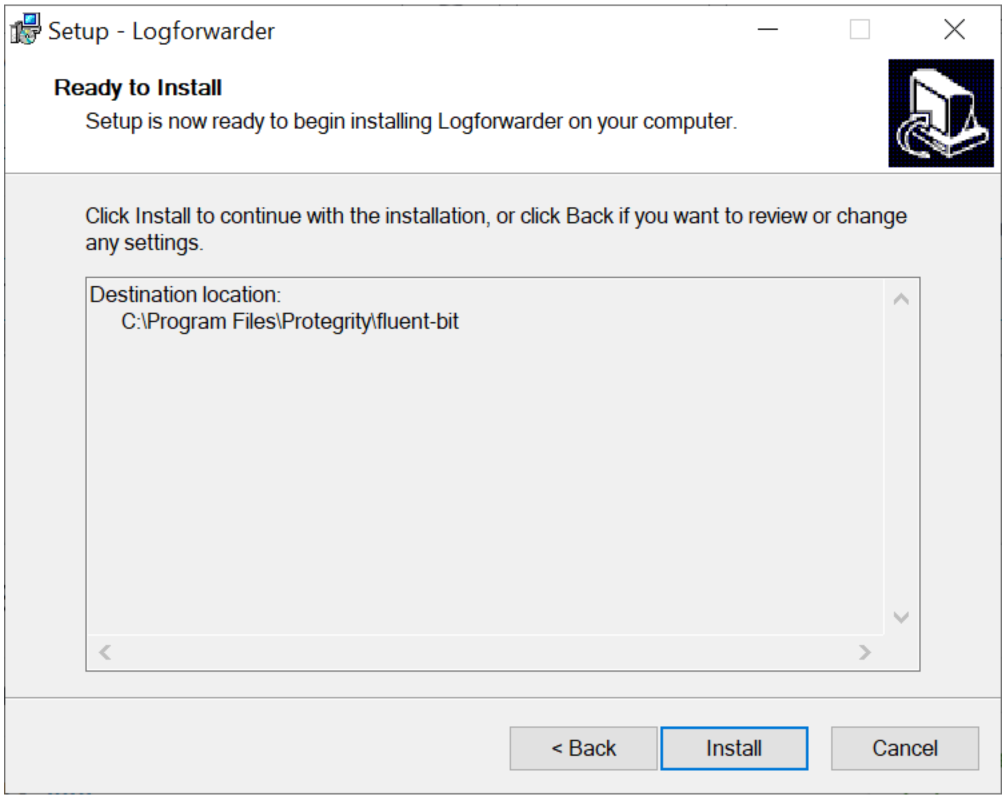
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\fluent-bit

At least 5.9 MB of free disk space is required.

8. Set the installation directory for the Log Forwarder to `C:\Program Files\Protegrity\fluent-bit`.
9. Click **Next**.

   The **Ready to Install** screen appears.



Setup - Logforwarder

Ready to Install

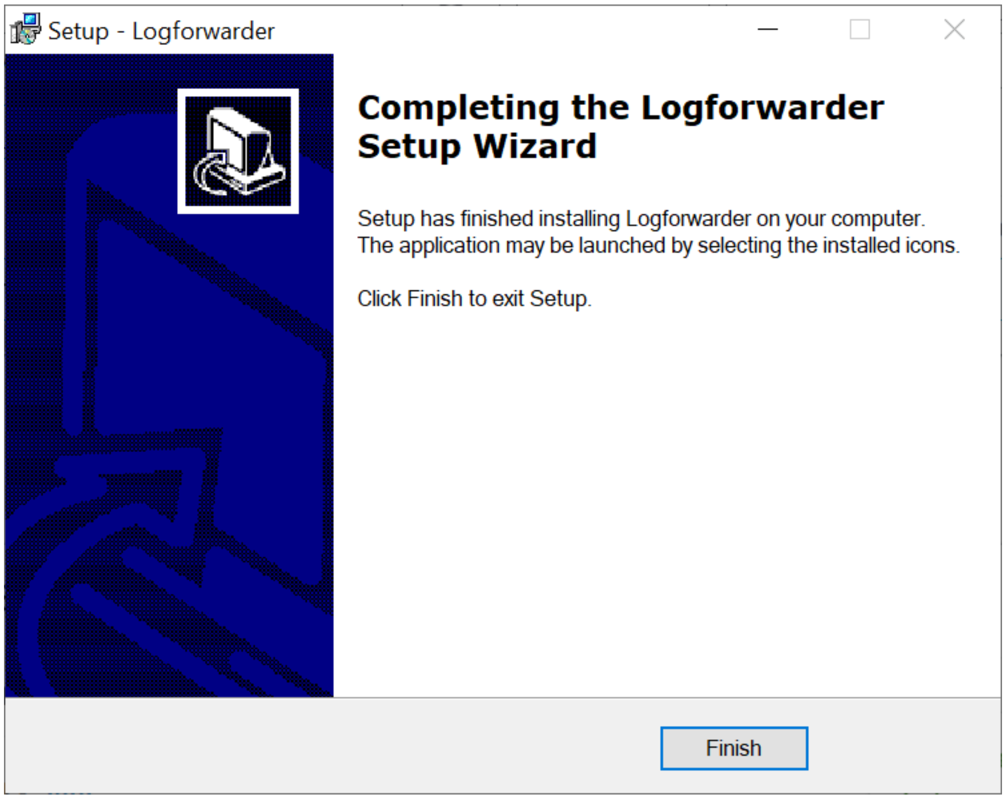
Setup is now ready to begin installing Logforwarder on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
    C:\Program Files\Protegrity\fluent-bit

10. Click **Install**.

    The **Completing the Logforwarder Setup Wizard** screen appears.


Setup - Logforwarder

Completing the Logforwarder Setup Wizard

Setup has finished installing Logforwarder on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

11. From the **Completing the Logforwarder Setup Wizard** screen, click **Finish** to complete the installation and exit.

    The directories are created under the installation directory that was defined, and the installation files are installed in these directories.

The Log Forwarder is installed successfully.

4.2.3.2 Using Silent Mode of Installation

This section describes how to install the Log Forwarder on the Policy Package Consuming Machine on a Windows platform through the Silent mode of installation.

You can also execute the Log Forwarder installer without any manual intervention, which is also known as the Silent mode of installation. The following parameters must be provided to execute the installer in the Silent mode.

Table 4-1: Parameter List for Silent Installation

Parameter	Description
-endpoint1, -endpoint2, -endpoint3	Audit Store IP address and the port number where the Log Forwarder listens for logs. **Note:** The default port number is *9200*. **Note:** The parameters *-endpoint2* and *-endpoint3* are optional.
-dir	Installation directory of the Log Forwarder, which is optional. If the installation directory is not specified, then the installation path is the default directory, which is the `C:\Program Files\Protegrity\fluent-bit` directory.

Parameter	Description
-pepdir	Installation directory of the PEP server, which is optional. If the installation directory is not specified, then the installation path is the default directory, which is the `C:\Program Files\Protegrity` directory.

At the command prompt, type the following command from the installer directory.

```
.\LogforwarderSetup_Windows_x64.exe -endpoint1 <ip address:port number> [-endpoint2 <ip
address:port number>]
[-endpoint3 <ip address and port number>]
```

If you want to install the Log Forwarder and the PEP server in a directory other than the default directory, then you can add the *-dir* parameter to the command to specify the Log Forwarder installation directory and the *-pepdir* parameter to the command to specify the PEP server installation directory.

The following snippet displays a sample command.

```
.\LogforwarderSetup_Windows_x64.exe -endpoint1 <ip address:port number> [-endpoint2 <ip
address:port number>]
[-endpoint3 <ip address and port number>] -dir <Log Forwarder installation directory> -pepdir
<PEP server installation directory>
```

4.2.4 Installing PEP server on the Policy Package Creator Machine

This section describes how to install the PEP server on the Policy Package Creator Machine on a Windows platform using the Windows Wizard and through the Silent mode of installation.

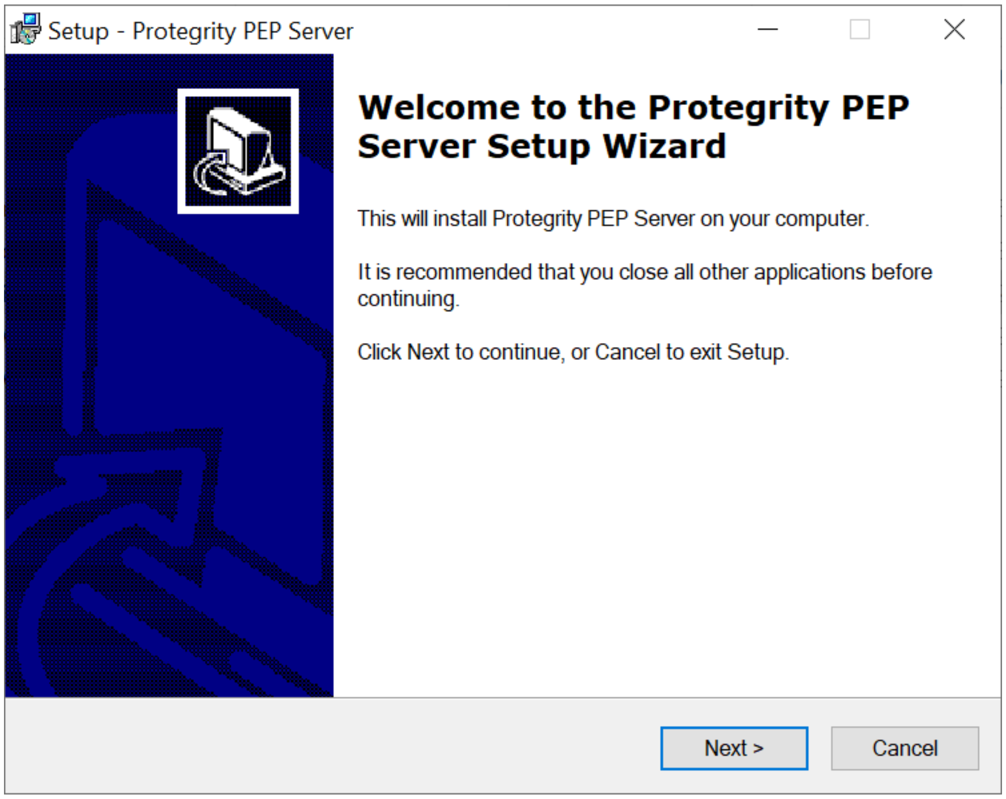
4.2.4.1 Using Windows Wizard

This section describes how to install the PEP server on the Policy Package Creator Machine on a Windows platform using the Windows Wizard.

To install the PEP server on the Policy Package Creator Machine using the Windows Wizard:

1. Run the *PepServerSetup_Windows_<bit-version>_<version>.exe* file from its installed directory.

   The **Welcome to the Protegrity PEP Server Setup Wizard** appears.



Welcome to the Protegrity PEP Server Setup Wizard

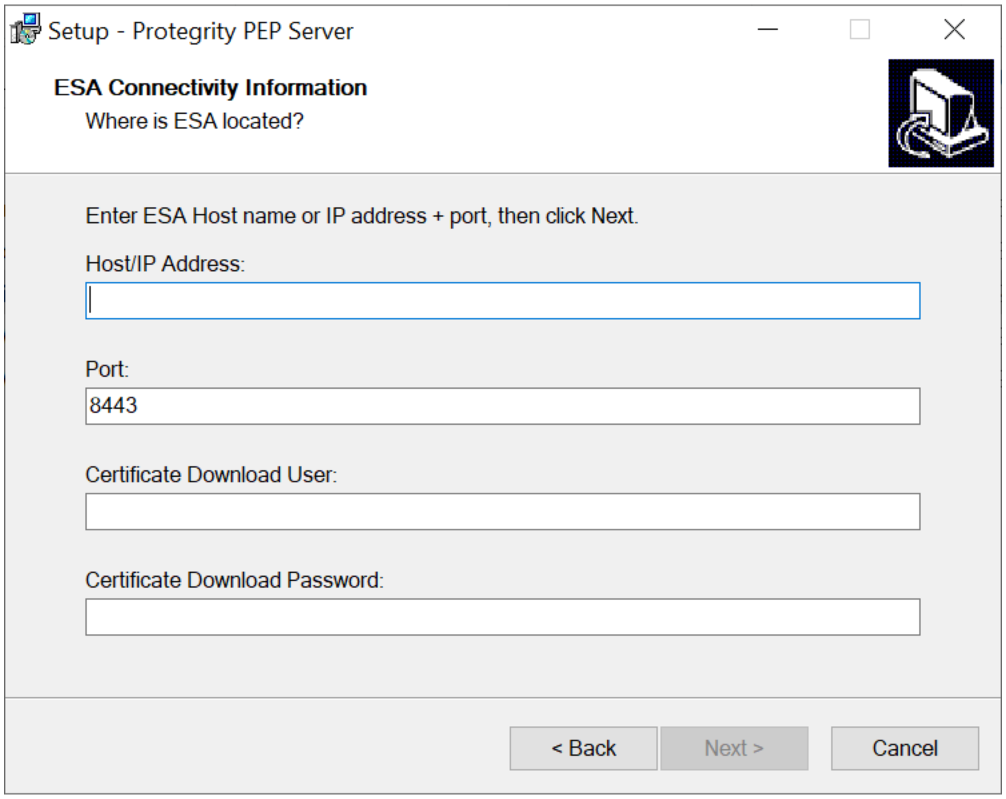
This will install Protegrity PEP Server on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

2. Click **Next**.

   The **ESA Connectivity Information** screen appears.



ESA Connectivity Information

Where is ESA located?

Enter ESA Host name or IP address + port, then click Next.

Host/IP Address:

Port: 8443

Certificate Download User:

Certificate Download Password:

3. Enter the ESA Host name or IP Address in the **Host/IP Address** field.

4. Enter the ESA host listening port details in the **Port** field.

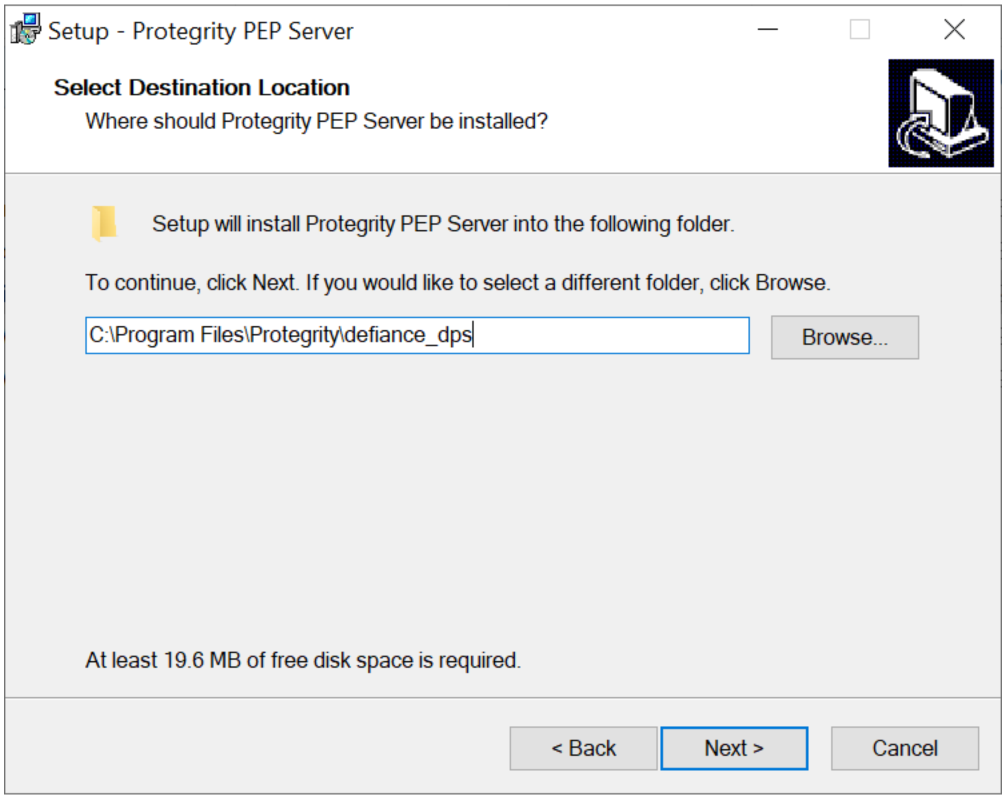
> **Note:** Ensure that the ESA is up and running with the *HubController* service in running status to enable the automatic downloading of certificates.

5. Enter the **Certificate Download User**.

> **Note:** It is recommended to use *admin* as the user.

6. Enter the **Certificate Download Password**.
7. Click **Next**.

   The **Select Destination Location** screen appears.



Setup - Protegrity PEP Server

Select Destination Location

Where should Protegrity PEP Server be installed?

Setup will install Protegrity PEP Server into the following folder.

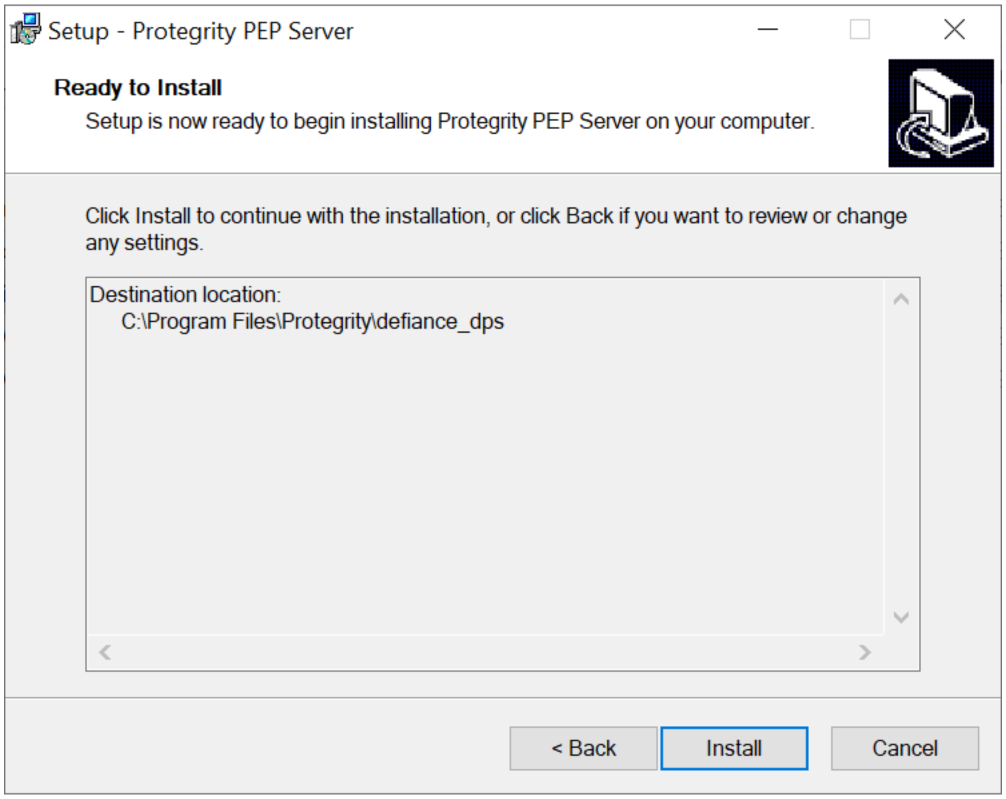
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\defiance_dps

At least 19.6 MB of free disk space is required.

8. Set the installation directory for the PEP server to `C:\Program Files\Protegrity\defiance_dps`.
9. Click **Next**.

   The **Ready to Install** screen appears.


Setup - Protegrity PEP Server

Ready to Install

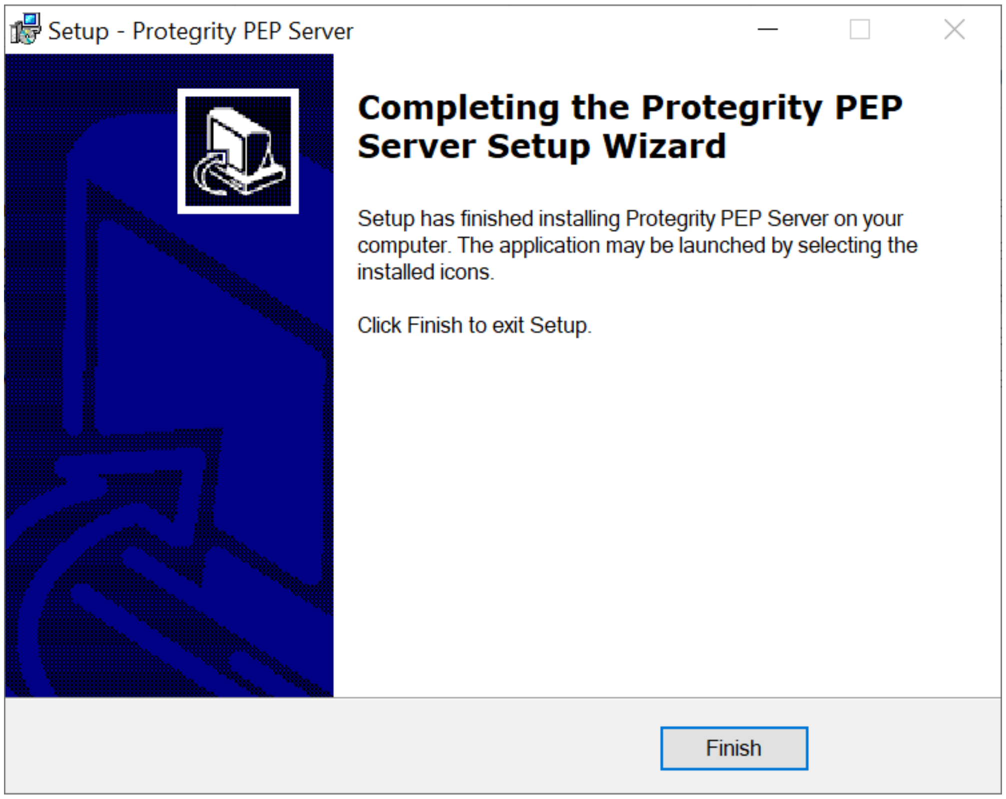
Setup is now ready to begin installing Protegrity PEP Server on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\Protegrity\defiance_dps

10. Click **Install**.

    The **Completing the Protegrity PEP Server Setup Wizard** screen appears.


Setup - Protegrity PEP Server

Completing the Protegrity PEP Server Setup Wizard

Setup has finished installing Protegrity PEP Server on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

11. From the **Completing the Protegrity PEP Server Setup Wizard** screen, click **Finish** to complete the installation and exit.

The directories are created under the installation directory that was defined and the installation files are installed in these directories.

> **Note:**
> If you want to manually install the certificates to the `C:\Program Files\Protegrity\defiance_dps\data` directory of the PEP server, then navigate to the `C:\Program Files\Protegrity\defiance_dps\bin` directory, and run the following command:
>
> ***`GetCertificates -u admin <Admin Username> [-h <ESA host name or IP address>] [-p portno] [-d directory]`***
>
> This initiates secure communication between the PEP server and the ESA.
>
> Enter the password for the user *admin*.

The PEP server is installed successfully.

4.2.4.2 Using Silent Mode of Installation

This section describes how to install the PEP server on the Policy Creator Machine on a Windows platform through the Silent mode of installation.

You can also execute the PEP server installer without any manual intervention, which is also known as the Silent mode of installation. The following parameters must be provided to execute the installer in Silent mode.

Table 4-2: Parameter List for Silent Installation

Parameter	**Description**
-esa	Specifies the ESA IP address.
-esaport	Specifies the ESA port, which is optional. The default value is *8443*.
-certuser	Specifies the ESA user to download certificates.
-certpw	Specifies the ESA user password to download certificates.
-dir	Specifies the installation directory, which is optional. If the installation directory is not specified, then the installation path is the default directory, which is the `C:\Program Files\Protegrity\defiance_dps` directory.

At the command prompt, type the following command from the installer directory.

```
PepServerSetup_Windows_<bit-version>.exe -certuser <username> -esa <esaIP> -certpw <password>
```

If you want to install the PEP server in a directory other than the default directory, then you can add the *-dir* parameter to the command to specify the directory. The following command displays a sample snippet.

```
PepServerSetup_Windows_<bit-version>.exe -certuser <username> -esa <esaIP> -certpw <password>
-dir <installation-directory-path>
```

4.2.5 Creating Passphrase and Salt on the Policy Package Creator Machine and Policy Package Consuming Machine

The passphrase and the salt are used to generate a key. The *PtyShmCat* utility uses this key to encrypt the policy. This is used to implement Passphrase Based Encryption (PBE). The salt is used as an initialization vector for generating the key.

To create the passphrase and the salt:

1. Create two text files, *passphrase.txt* and *salt.txt* to store the passphrase and salt respectively.

   Important: Ensure that you use the same passphrase and salt on the Policy Package Creator Machine and the Policy Package Consuming Machine.

   Caution: This is a known security issue that data in passphrase and salt files are in clear text. You must ensure that these files are not exposed to unauthorized users. If an unauthorised user gets access to the passphrase and salt files, then they can decrypt the policy package.

   Note: Ensure that the *passphrase.txt* and *salt.txt* files do not contain a new line in it.

2. Ensure that you have installed the cryptographic and password strength package.

   For more information about installing the cryptographic package and password strength package, refer to the section *Verifying the Prerequisites*.

3. Set the following System Environment variables using the following commands.

   a. ***`PTYSALT=<path of salt file>`***

      For example:

      ```
      set PTYSALT=C:\Program Files\Protegrity\shmutilities\salt.txt
      ```

   b. ***`PTYPASSPHRASE=<path of passphrase file>`***

      For example:

      ```
      set PTYPASSPHRASE=C:\Program Files\Protegrity\shmutilities\passphrase.txt
      ```

4. Ensure that the password set in the *passphrase.txt* and *salt.txt* files must consist of the following:
   - A minimum total length of 12 characters
   - A minimum of 1 upper case letter
   - A minimum of 2 special characters
   - A minimum of 2 non-letter characters such as a digital or a special character
   - A minimum of 30 entropy bits

     Note: Entropy bits define the randomness of the password and the strength defines the complexity of the password.

   - A password that has 0.5 strength

     For more information about the password strength, refer to section *Password Strength*.

4.2.6 Running the PtyShmCat Utility on the Policy Package Creator Machine

This section describes how to run the *PtyShmCat* utility on the Policy Package Creator Machine on a Windows platform.

Note:

If you do not install the Log Forwarder, then ensure that the log output parameter in the *pepserver.cfg* file, which is present at `C:\Program Files\Protegrity\defiance_dps\data`, is set to *stdout* before creating the policy package.

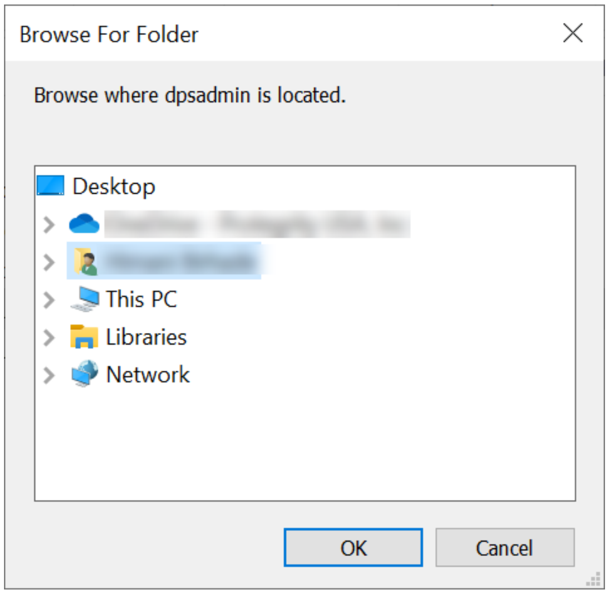
Note:

You must restart the PEP server from the **Services** after updating the configurations in the *pepserver.cfg* file for the changes to take effect.

➤ To run the *PtyShmCat* utility on the Windows platform:

1. Install the *PtyShmCat* utility by running the following setup file.
   `PtyShmCatSetup_Windows_<bit-version>_<version>.exe`

   The **Browse For Folder** screen appears.

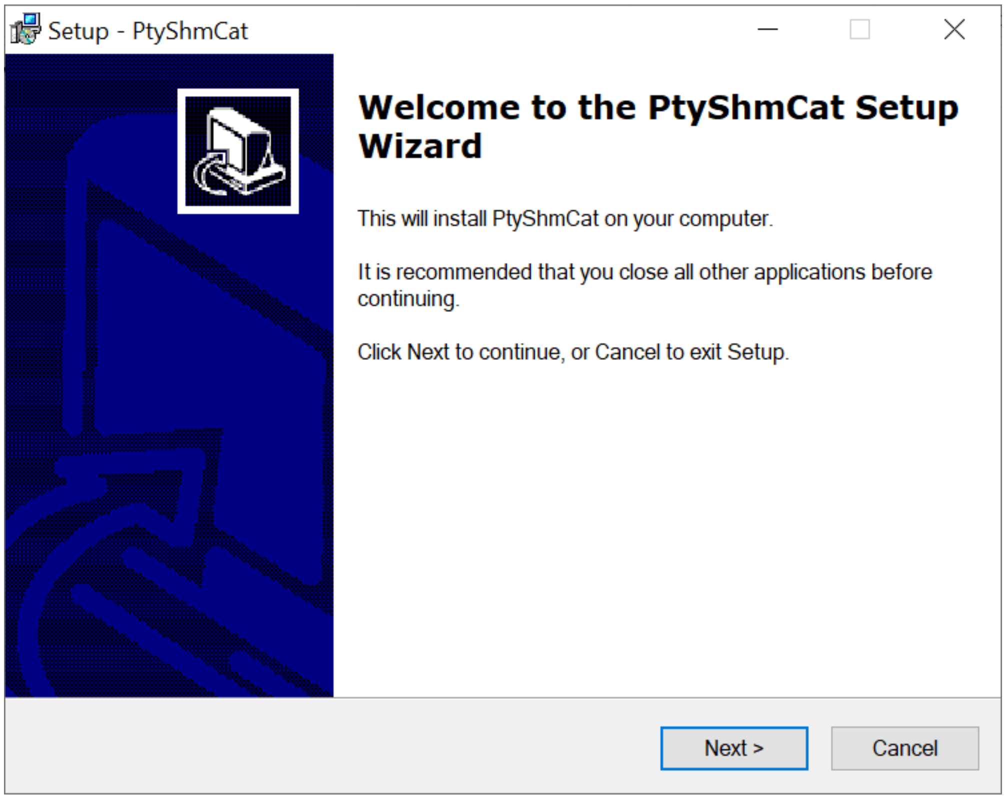
   

   Browse For Folder

   Browse where dpsadmin is located.

2. Select the path where the dpsadmin is located.
3. Click **OK**.

   The **Welcome to the PtyShmCat Setup Wizard** screen appears.


Setup - PtyShmCat

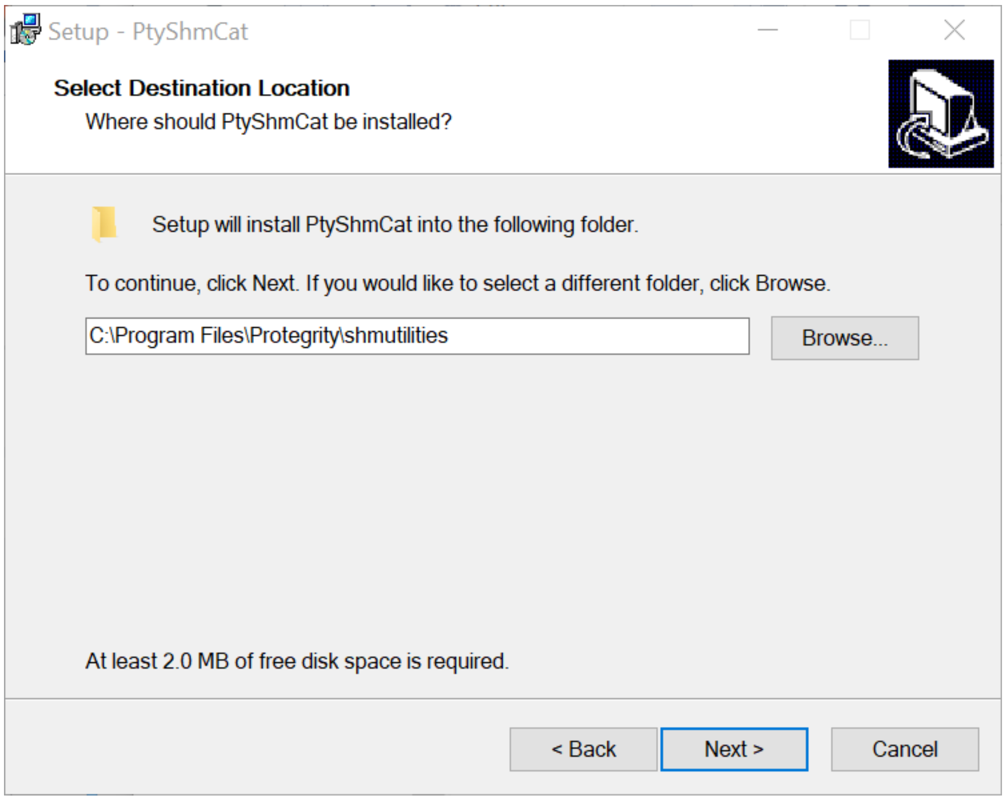
Welcome to the PtyShmCat Setup Wizard

This will install PtyShmCat on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

4. Click **Next**.
   The **Select Destination Location** screen appears.


Setup - PtyShmCat

Select Destination Location
Where should PtyShmCat be installed?

Setup will install PtyShmCat into the following folder.

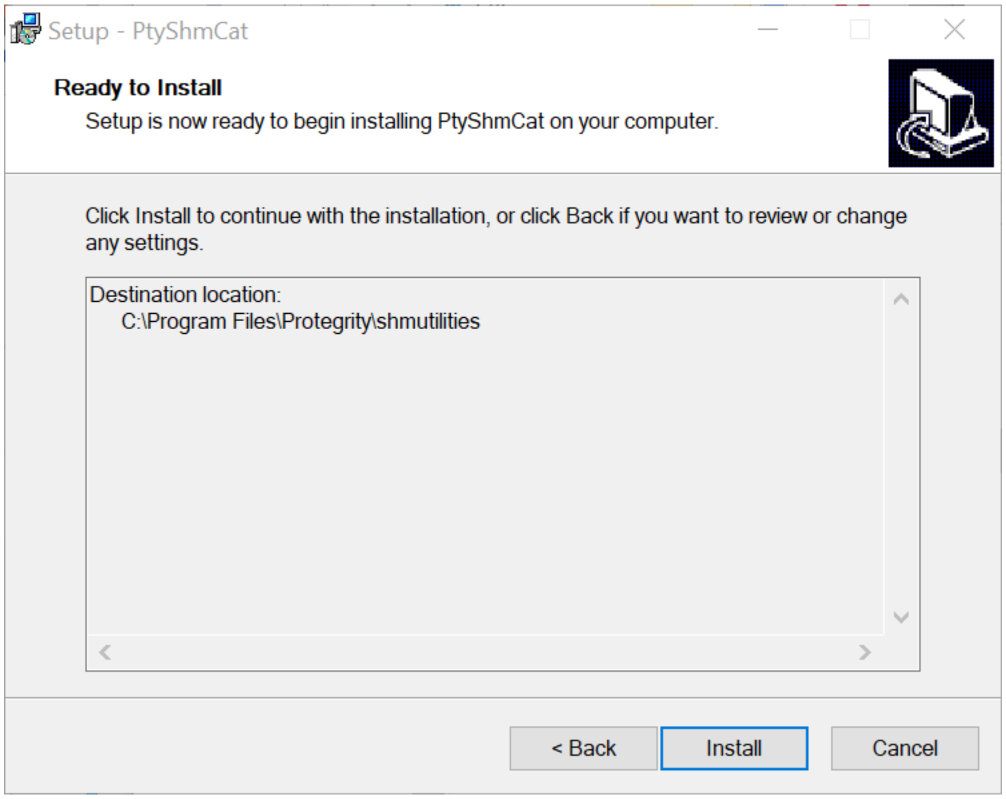
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\shmutilities

At least 2.0 MB of free disk space is required.

5. Select the installation directory for the *PtyShmCat* utility to `C:\Program Files\Protegrity\shmutilities`.
6. Click **Next**.

The **Ready to Install** screen appears.



Ready to Install

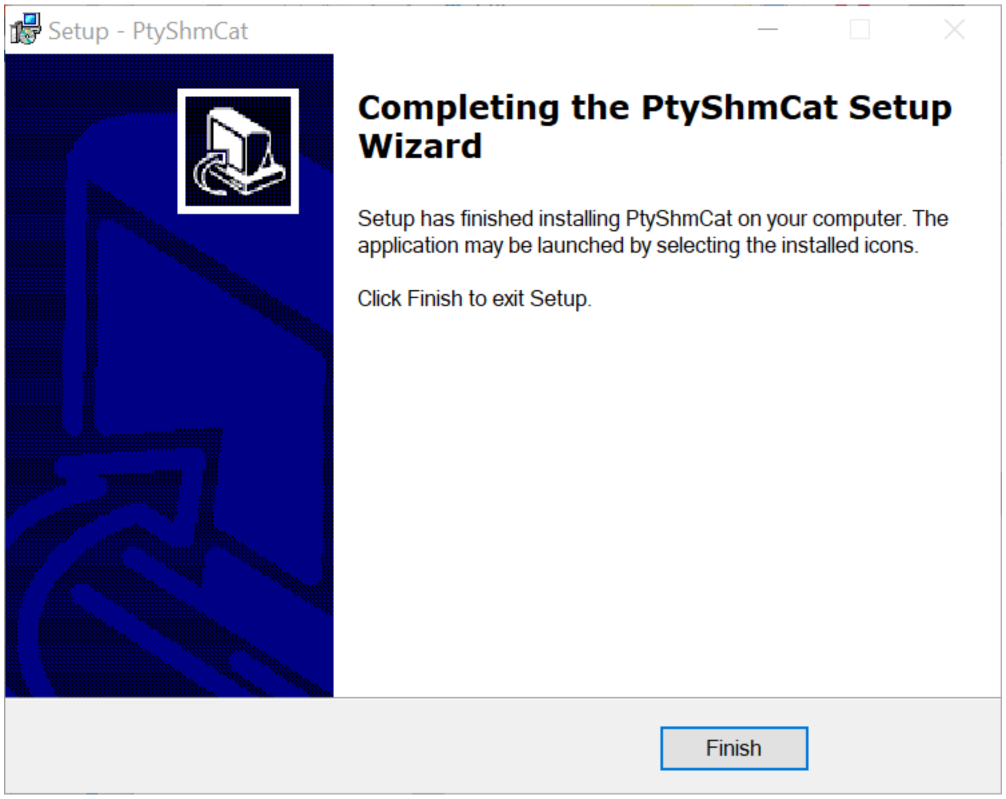
Setup is now ready to begin installing PtyShmCat on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\Protegrity\shmutilities

7. Click **Install**.

   The **Completing the PtyShmCat Setup Wizard** screen appears.



Completing the PtyShmCat Setup Wizard

Setup has finished installing PtyShmCat on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

8. From the **Completing the PtyShmCat Setup Wizard** screen, click **Finish** to complete the installation and exit.

The directories are created under the installation directory that was defined, and the installation files are installed in these directories.

9. Run the *imp_creator.py* script by using the following command.
   ```
   python imp_creator.py <policy package name> --local_path="<path to store policy package>"
   ```
   An encrypted policy package is generated at the local path defined by the user.

   > **Note:**
   > Ensure that you have copied the encrypted policy package to the Policy Package Consuming Machine.

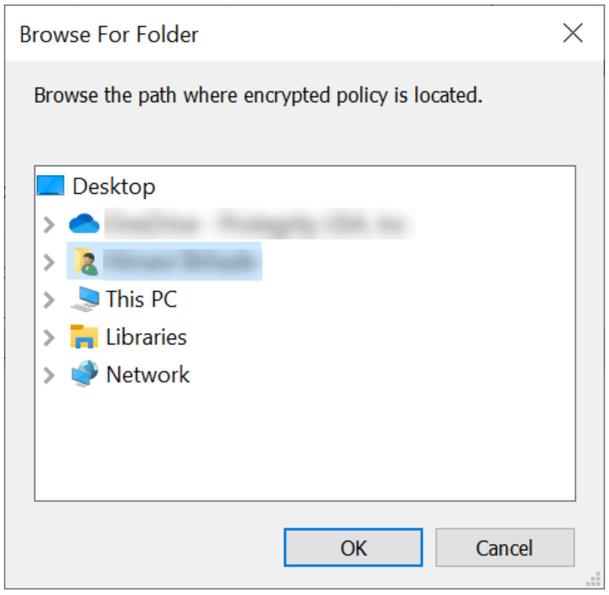
The *PtyShmCat* utility is installed successfully.

4.2.7 Running the PtyShmPut Utility on the Policy Package Consuming Machine

This section describes how to run the *PtyShmPut* utility on the Policy Package Consuming Machine on a Windows platform.

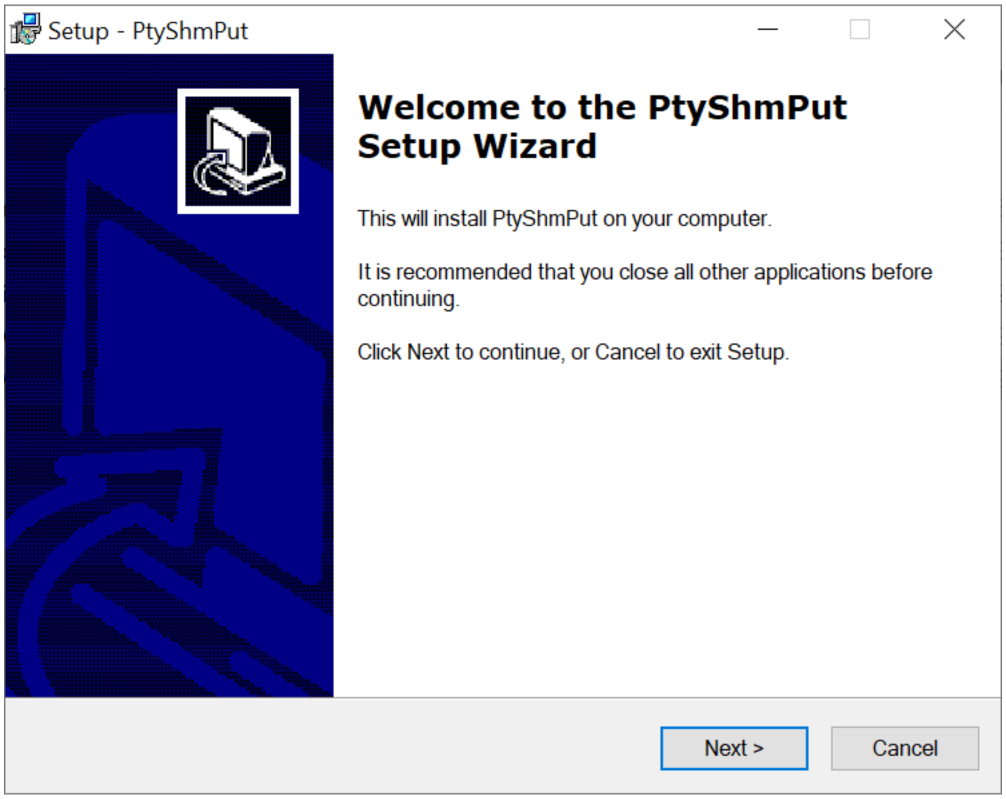
To run the *PtyShmPut* utility on the Windows platform:

1. Install the *PtyShmPut* utility by running the following setup file.
   ```
   PtyShmPutSetup_Windows_<bit-version>_<version>.exe
   ```
   The **Browse For Folder** screen appears.


Browse For Folder

Browse the path where encrypted policy is located.

2. Select the path where the encrypted policy path is located.
3. Click **OK**.
   The **Welcome to the PtyShmPut Setup Wizard** screen appears.



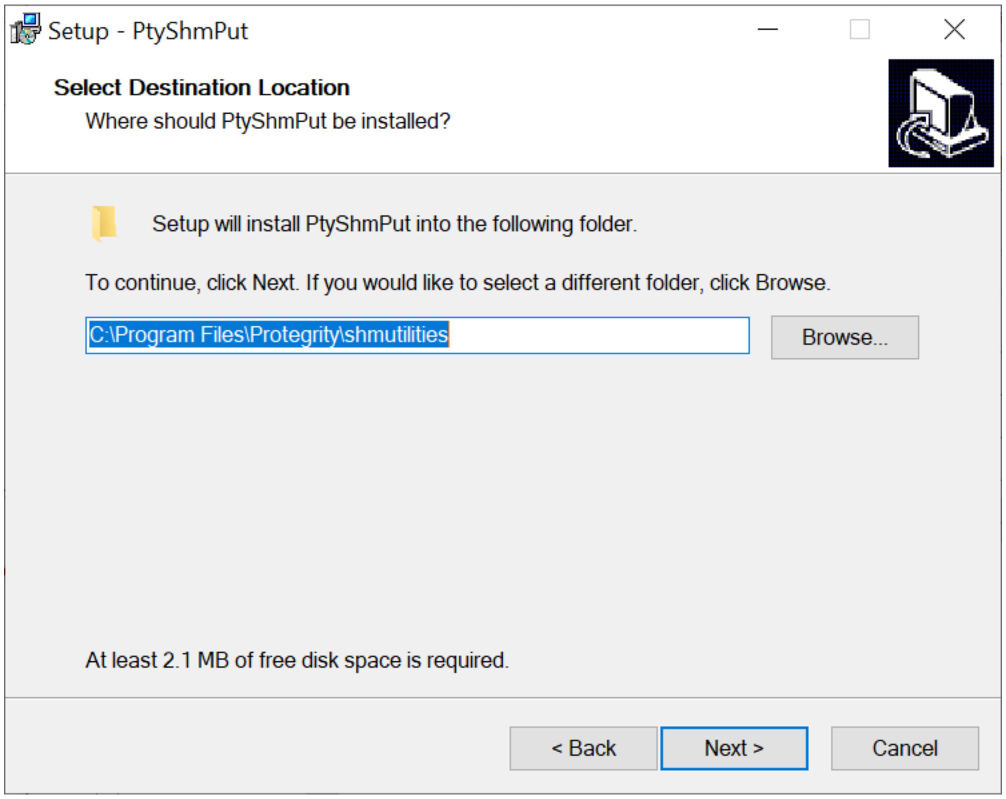
Welcome to the PtyShmPut Setup Wizard

This will install PtyShmPut on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

4. Click **Next**.
   The **Select Destination Location** screen appears.



Select Destination Location
Where should PtyShmPut be installed?

Setup will install PtyShmPut into the following folder.

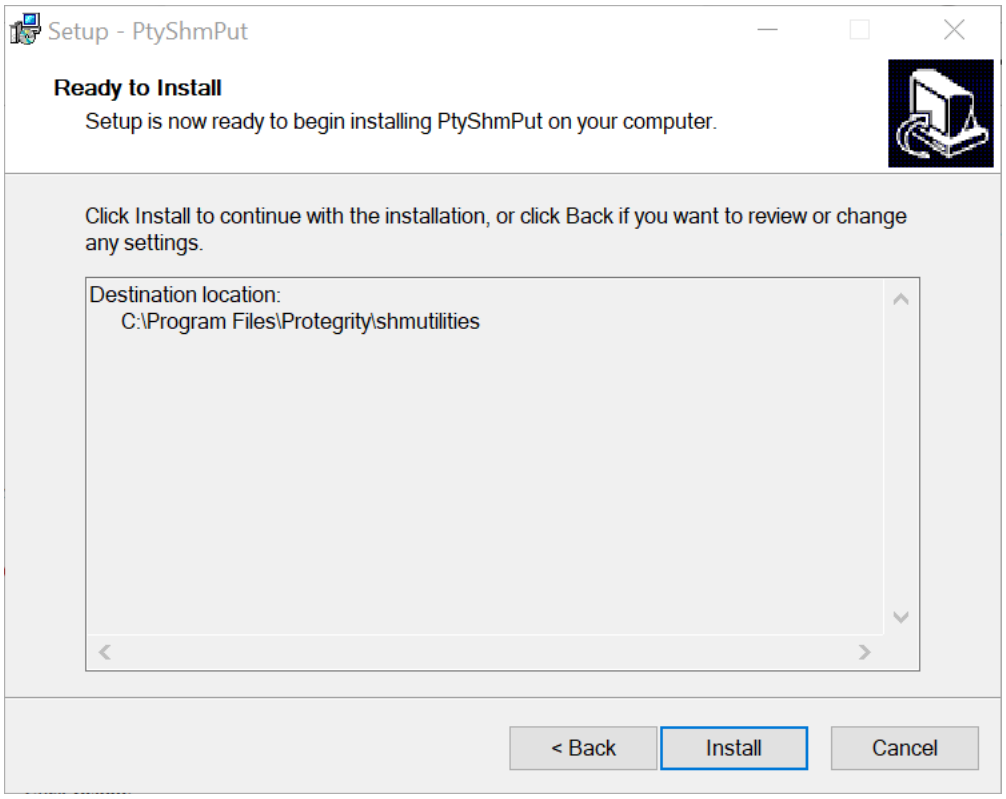
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\shmutilities

At least 2.1 MB of free disk space is required.

5. Select the location where the *PtyShmPut* utility needs to be installed.
6. Click **Next**.

The **Ready to Install** screen appears.



Ready to Install

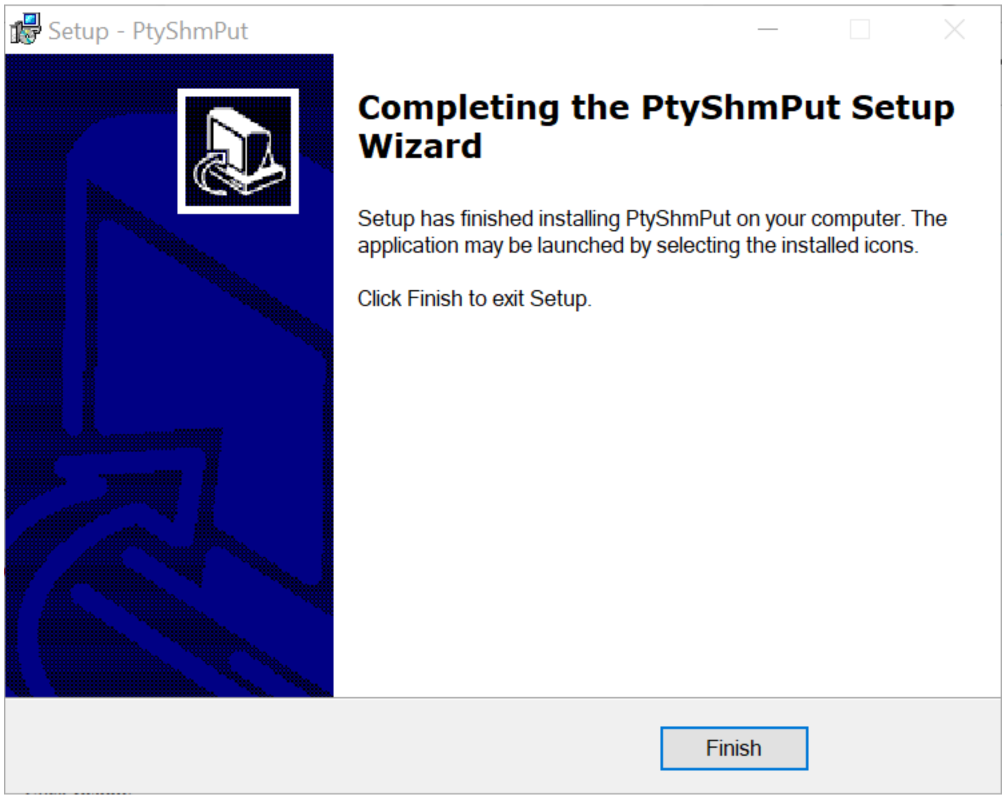
Setup is now ready to begin installing PtyShmPut on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Program Files\Protegrity\shmutilities

7. Click **Install**.

   The **Completing the PtyShmPut Setup Wizard** screen appears.



Completing the PtyShmPut Setup Wizard

Setup has finished installing PtyShmPut on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

8. From the **Completing the PtyShmPut Setup Wizard** screen, click **Finish** to complete the installation and exit.

The directories are created under the installation directory that was defined, and the installation files are installed in these directories.

9. Ensure that you have copied the encrypted policy package created in the Policy Package Creator Machine to the Policy Package Consuming Machine.
10. Navigate to the `C:\Program Files\Protegrity\shmutilities\scripts` directory on the Policy Package Consuming Machine.
11. Run the *imp_updater.py* script by using the following command.

    Note: Ensure that you run the command prompt as an administrator.

    ```
    python imp_updater.py <policy package name>
    ```

    A decrypted policy package is created.

    Note:
    If you are using same machine for installing *PtyShmPut* utility, then you need to uninstall the PEP server setup.

    Note:
    You can reconfigure the System Environment variable *PTY_SHM_POLICY_PATH* created for the policy path. However, you need to run the *imp_updater.py* utility again for the changes to take effect.

    Note:
    Ensure that you do not start the *PtyShmPut* service manually as it starts automatically during the execution of the*imp_updater.py* utility.

    If for any reason the *PtyShmPut* service is stuck at the **starting** state, then navigate to the **Task Manager** and click on the **End Task** option to end the *PtyShmPut* service manually.

The *PtyShmPut* utility is installed successfully.

4.2.8 Verifying the Policy Health on the Policy Package Consuming Machine

This section describes how to verify the policy health on a Windows platform.

To verify the policy health on the Policy Package Consuming Machine on the Windows platform:

1. On the policy consuming machine, navigate to the `C:\Program Files\Protegrity\shmutilities\bin` directory.
2. To verify whether the data protection policies are published successfully, run the following script.

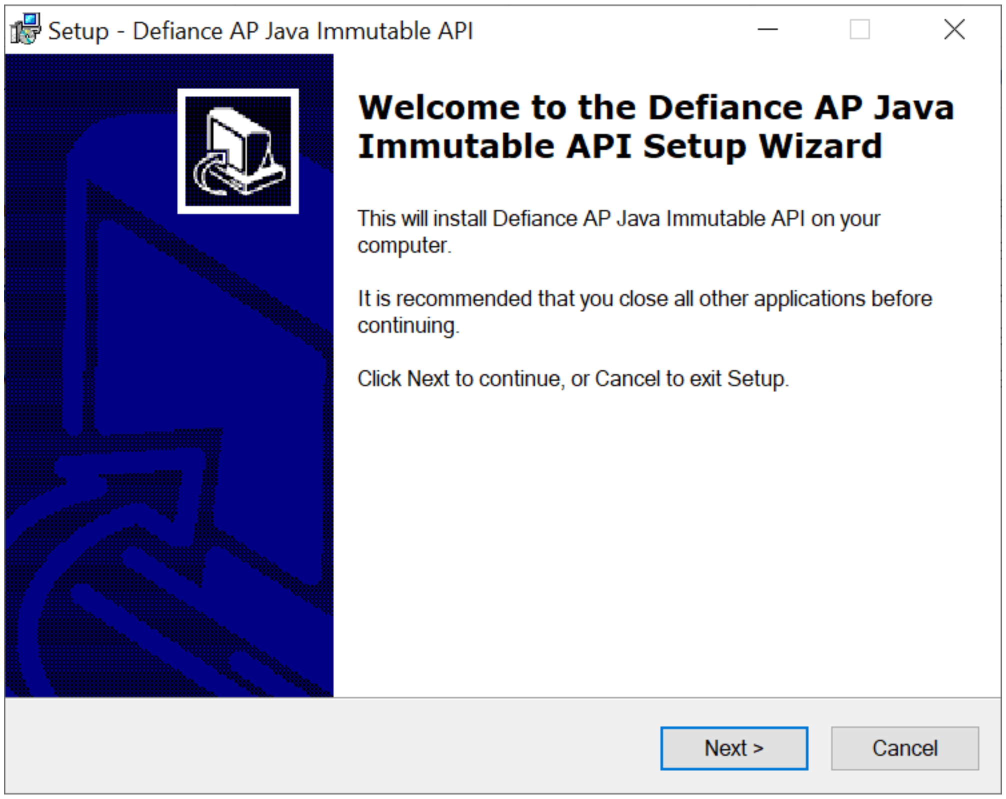
    ```
    PTYPolicyHealthCheck.bat
    ```

4.2.9 Installing Immutable Application Protector Java on the Policy Package Consuming Machine

This section describes how to install the IAP Java on the Policy Package Consuming Machine on a Windows platform.

To install the IAP Java on the Windows platform:

1. Run the following setup file to install the IAP Java.
   `APJavaIMSetup_Windows_x64_<version>.exe`
   The **Welcome to the Defiance AP Java Immutable API Setup Wizard** appears.



Setup - Defiance AP Java Immutable API

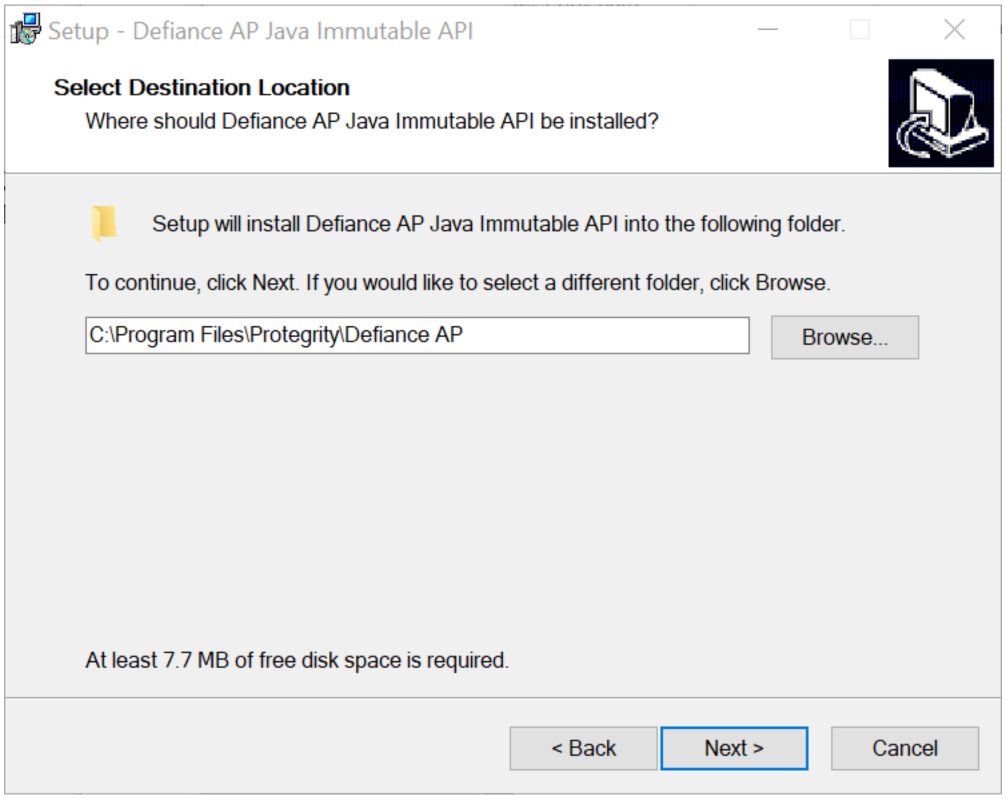
Welcome to the Defiance AP Java Immutable API Setup Wizard

This will install Defiance AP Java Immutable API on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

2. Click **Next**.
   The **Select Destination Location** screen appears.



Setup - Defiance AP Java Immutable API

Select Destination Location

Where should Defiance AP Java Immutable API be installed?

Setup will install Defiance AP Java Immutable API into the following folder.

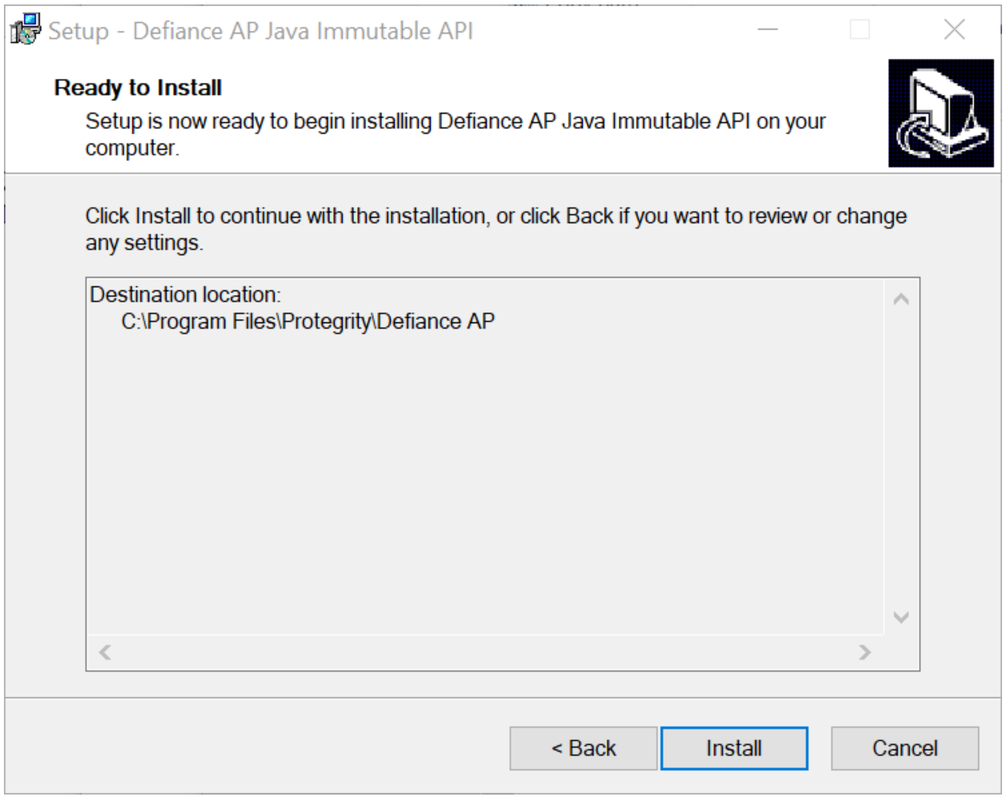
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\Protegrity\Defiance AP

At least 7.7 MB of free disk space is required.

3. Click **Next**.

   The **Ready to Install** screen appears.



Setup - Defiance AP Java Immutable API

Ready to Install

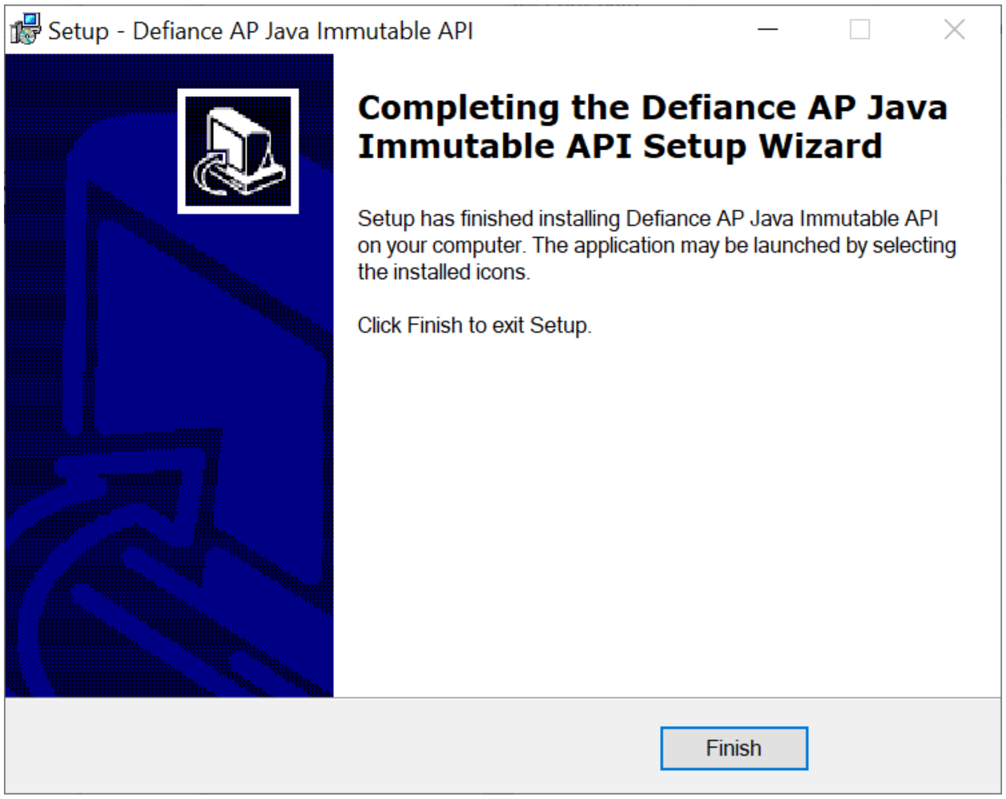
Setup is now ready to begin installing Defiance AP Java Immutable API on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
    C:\Program Files\Protegrity\Defiance AP

4. Click **Install**.

   The **Completing the Defiance AP Java Immutable API Setup Wizard** screen appears.


Setup - Defiance AP Java Immutable API

Completing the Defiance AP Java Immutable API Setup Wizard

Setup has finished installing Defiance AP Java Immutable API on your computer. The application may be launched by selecting the installed icons.

Click Finish to exit Setup.

5. From the **Completing the Defiance AP Java Immutable API Setup Wizard** screen, click **Finish** to complete the installation and exit.

   The directories are created under the installation directory that was defined and the installation files are installed in these directories.

6. Verify that the following libraries are installed in the IAP Java installation directory:
   - *ApplicationProtectorJava.jar*
   - *ApplicationProtectorJava.properties*
   - *jna-5.8.0.jar*
   - *jna-platform-5.8.0.jar*
   - *jpeplite_imp.plm*
7. Link your application to the *ApplicationProtectorJava.properties* and *.jar* files.

The IAP Java is installed successfully.

4.2.10 Uninstalling the Log Forwarder from the Windows platform

This section describes how to uninstall the Log Forwarder from a Windows platform.

➤ To uninstall the Log Forwarder from the Windows platform:

1. Perform the following steps to stop the Log Forwarder.
   a. From the Windows Start Menu, search and select **Services**.
   b. Navigate to the **Logforwarder** directory.
   c. Right-click the **Logforwarder** service and click **Stop**.

2. Run the uninstall utility for Log Forwarder located in the `C:\Program Files\Protegrity\fluent-bit` directory.
3. Delete the `C:\Program Files\Protegrity\fluent-bit` directory.

The Log Forwarder is uninstalled.

4.2.11 Uninstalling the PEP Server from Windows

This section describes how to uninstall the PEP server from a Windows platform.

To uninstall the PEP server from a Windows platform:

1. Perform the following steps to stop the PEP server.
   a. From the Windows Start Menu, search and select **Services**.
   b. Navigate to the **Protegrity PEP Server** directory.
   c. Right-click the **Protegrity PEP Server** service and click **Stop**.
2. Run the uninstall utility for the PEP server located in the `C:\Program Files\Protegrity\defiance_dps` directory.
3. Delete the `C:\Program Files\Protegrity\defiance_dps` directory.

The PEP server is uninstalled.

4.2.12 Uninstalling PtyShmCat and PtyShmPut Utilities on Windows

This section describes how to uninstall the PtyShmCat and PtyShmPut utilities from the Windows platform.

To uninstall the PtyShmCat and PtyShmPut utilities from the Windows platform:

1. Navigate to the `C:\Program Files\Protegrity\shmutilities` directory.
2. Run the following file to uninstall the PtyShmCat and PtyShmPut utilities.
   ```
   unins000.exe
   ```
3. Delete the following directory.
   ```
   C:\Program Files\Protegrity\shmutilities
   ```

The PtyShmCat and PtyShmPut utilities are uninstalled.

4.2.13 Uninstalling Immutable Application Protector Java on Windows

This section describes how to uninstall the IAP Java from a Windows latform.

To uninstall the IAP Java from the Windows platform:

1. Navigate to the `C:\Program Files\Protegrity\Defiance AP` directory.
2. Run the following file to uninstall the IAP Java.

   ```
   unins000.exe
   ```

3. Delete the following directory.

   ```
   C:\Program Files\Protegrity\Defiance AP
   ```

The IAP Java is uninstalled.

4.2.14 Running IAP Java - Example

This section provides examples on how to use the IAP Java protect, unprotect, and reprotect APIs. It is assumed that policies are made available to the Policy Package Consuming Machine in a decrypted form and the IAP Java has been successfully installed.

For more information on running the IAP Java sample application, refer to the section *Running IAP - Example*.

Chapter 5

Installing Immutable Application Protector (IAP) Python

The Immutable Application Protector Python provides APIs for performing security operations.

All standard protection techniques offered by Protegrity are applicable to the Immutable Application Protector Python.

5.1 Setting up Immutable Application Protector Python on Linux or Unix

This section describes how to set up the IAP Python on a Linux or Unix platform.

5.1.1 Verifying the Prerequisites

This section describes the prerequisites for setting up the IAP Python on a Linux or Unix platform.

Before you begin
Ensure that the following prerequisites are met:

- Glibc, version 2.12 or higher, is installed.
- GCC, version 3.2.3 or higher, is installed.
- The ESA is installed, configured, and running.
- The IP address or host name of the ESA is noted.
- Python, version 3.7, 3.8, 3.9, 3.10, and 3.11, for the Linux operating system.

5.1.2 Downloading the Immutable Application Protector Python Installation Package

This section describes the steps to download the IAP Python installation package on a Linux or Unix platform.

To setup the IAP Python installation package on the Linux or Unix platform:

1. Download the *IAP_Linux-ALL-64_x86-64_PY-3.11_<version>.tgz* installation package to any location on the machine where you want to install the protector.
2. Create a directory on your policy package creator machine where you want to install the PEP server.
3. Extract the IAP Python installation package using the following command.

   ```
   tar -xvf IAP_Linux-ALL-64_x86-64_PY-3.11_<version>.tgz
   ```

   The following setup files are extracted:

   - *APPythonIMSetup_Linux_x64_9.2.0.0.2.tar*

- *PolicyUtilSetup_Linux_x64_<version>.sh*
- *manifest.json*

5.1.3 Installing the Immutable Application Protector Python on Policy Consuming Machine

This section describes how to install the IAP Python on a Linux or Unix platform.

To install the IAP Python on the Linux or Unix platform:

1. Run the following command to install the IAP Python setup file.

```
pip install APPythonIMSetup_Linux_x64_<version>.tgz
```

2. Run the following command to verify that the installation is done correctly.

```
pip list
```

5.1.4 Running the REST Client utility on the Policy Package Creator Machine

This section describes how to run the cURL utility on the Policy Package Machine.

To run the cURL utility:

1. Run the following command to know the PEP version supported by the ESA:

curl -k -v -u "<user>:<password>" -X GET https://<ESA_IP_ADDRESS>/pty/v1/imp/version

Sample output:

```
{
  "version" : "<version>",
  "buildVersion" : "<build version>",
  "pepVersions" : [ "<pepVersions supported by the ESA>" ]
}
```

Note: Ensure that the password complexity meets OWASP recommendations. For more information about the password requirements, refer to section *Password Strength*.

2. Run the following command to fetch the policy dump from the ESA:

```
curl -k -v --location 'https: //<ESA_IP>/pty/v1/imp/export' \
-u "<USER>:<PASSWORD>" \
--header 'Content-Type: application/json' \
--data '{
    "name": "<Request_Name>",
    "dataStoreName": "<DataStoreName>",
    "pepVersion": "<pepVersion>",
    "emptyString":"NULL",
    "caseSensitive":"YES",
    "output":"STDOUT",
    "kek": {
        "pkcs5": {
            "password": "<PASSWORD>",
            "salt": "<SALT>"
        }
    }
}' -o <Policy_Package_Name>.json
```

> **Note:** Ensure that the output parameter is set to *STDOUT* to avoid the loss of logs.

> **Note:** Ensure that the value passed for the *pepVersion* parameter in the command matches with the *pepVersion* parameter in the *manifest.json* file.

> **Note:** The values passed for the parameters in the command must match the corresponding values in the ESA.

> **Note:** It is recommended that the *<USER>* exporting the policy should have minimum privileges. The least privilege that is required is the *Export IMP* permissions on the ESA.

An encrypted policy package file is generated at the local path defined by the user.

3. Ensure that you have copied the encrypted policy package to the Policy Package Consuming Machine.

5.1.5 Running the PolicyUtil Binary on the Policy Consuming Machine

This section describes how to run the *import* utility on the Policy Package Consuming Machine.

➤ To run the *import* utility:

1. Run the following command to install the PolicyUtil setup, which has the *impgen2* binary.

   ```
   ./PolicyUtilSetup_Linux_x64_<version>.sh
   ```

2. The following directory is created in the `/opt/protegrity` directory.
   `/opt/protegrity/imp-policy-util`
3. Navigate to the `/opt/protegrity/imp-policy-util/bin/impgen2` directory available on the Policy Package Consuming Machine.
4. Run the *impgen2* binary by using the following command.

   ```
   ./impgen2 import -passphrase <password> -salt <salt> -policy-file <file_path>/
   <policy_name>.json -pepversion <pepversion>
   ```

> **Note:** The IMP package PEP version must match the protector PEP version.

> **Note:**
> Passphrase and salt can either be given as clear texts or as environment variables.
>
> For example: *PTY_SALT* or *PTY_SALT_PATH*.
>
> If a file is used for storing passphrase and salt, then ensure that there is no new line in it. For more information regarding the binary, run `./impgen2 help`.

The *PolicyUtil* binary repopulates the shared memory on the Policy Package Consuming Machine.

5. Run the following command to view the populated shared memory on the Policy Package Consuming Machine.

   ```
   ipcs -a
   ```

   After the shared memory is populated, the protector can be used to perform the protect, unprotect, and reprotect operations.

5.1.6 Verifying the Policy Health on the Policy Package Consuming Machine

This section describes how to verify the policy health on the Policy Package Consuming Machine.

➤ To verify the policy health:

1. Navigate to the `/opt/protegrity/imp-policy-util/bin/` directory.
2. Run the following command to verify the policy health.

   ```
   ./impgen2 health
   ```

   The following output is expected.

   ```
   health check status: healthy
   ```

5.1.7 Uninstalling PolicyUtil Binary from Policy Package Consuming Machine

This section describes how to uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine.

➤ To uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine:

1. Navigate to the `/opt/protegrity/imp-policy-util` directory.
2. Remove the `/imp-policy-util` directory.

5.1.8 Uninstalling Immutable Application Protector Python from Linux

This section describes how to uninstall IAP Python from a Linux platform.

➤ To uninstall the IAP Python from the Linux platform:

1. Login to the machine from where you want to uninstall the IAP Python.
2. Uninstall the IAP Python by running the following command.

   ```
   pip uninstall appython
   ```

5.1.9 Running IAP Python - Example

This section provides examples on how to use the IAP Python protect, unprotect, and reprotect APIs. It is assumed that policies are made available to the policy consuming machine in a decrypted form and the IAP Python has been successfully installed.

For more information on running the IAP Python sample application, refer to the section *Running IAP - Example*.

Chapter 6

Installing Immutable Application Protector (IAP) Go

The Immutable Application Protector Go provides APIs for performing security operations.

All standard protection techniques offered by Protegrity are applicable to the Immutable Application Protector Go.

6.1 Setting up Immutable Application Protector Go on Linux or Unix

This section describes how to set up the IAP Go on a Linux or Unix platform.

6.1.1 Verifying the Prerequisites

This section describes the prerequisites foe setting up the IAP Go on a Linux or Unix platform.

Before you begin
Ensure that the following prerequisites are met:

- Glibc, version 2.12 or higher, is installed.
- GCC, version 3.2.3 or higher, is installed.
- The ESA is installed, configured, and running.
- The IP address or host name of the ESA is noted.
- Go 1.x is installed.

6.1.2 Downloading the Immutable Application Protector Go Installation Package

This section describes how to download the IAP Go on a Linux or Unix platform.

To download the IAP Go on the Linux or Unix platform:

1. Download the *IAP_Linux-ALL-64_x86-64_GO-1_<version>.tgz* installation package to any location on the machine where you want to install the protector.
2. Create a directory on your machine where you want to install the protector.
3. Extract the IAP Java installation package using the following command.

   ```
   tar -xvf IAP_Linux-ALL-64_x86-64_GO-1_<version>.tgz
   ```

   The following setup files are extracted:

   - *PolicyUtilSetup_Linux_x64_<version>.sh*

- *APGoIMSetup_Linux_x64_<version>.sh*
- *manifest.json*

6.1.3 Installing Immutable Application Protector Go on Linux or Unix

This section describes how to install the IAP Go on a Linux or Unix platform.

➤ To install the IAP Go on the Linux or Unix platform:

You can install the IAP Go using either the GOPATH approach or the Go Module approach.

Using the Go Module approach:

a. Run the IAP Go installer using the following command.

```
./APGoIMSetup_Linux_x64_<version>.sh
```

The prompt to continue the installation appears.

```
*****************************************************
 Welcome to the Defiance DPS Immutable AP Go API Setup Wizard
*****************************************************

This will install Immutable AP Go API on your computer.
Do you want to continue? [yes or no]
```

b. If you want to continue with the installation of the IAP Go, then type *yes* else type *no*.

If you type *yes*, then the prompt to enter the installation directory appears.

```
Please enter installation directory
[/opt/protegrity]:
```

If you type *no*, then the installation of the IAP Go aborts.

c. Press **ENTER** after you type the installation directory. The following prompt appears.

```
Would you like to install AP Go as a Go Module? [yes or no]
```

d. Type *yes* to install the IAP Go using the Go Module approach.

Note: Ensure that you have installed the Go binaries on your local machine to install the AP Go using the Go Module approach.

e. Provide the path to the *private* Version Control System repository (VCS). For example, `privaterepo.example.com/app` where *app* is the name of project. The private VCS path will be used by the installer as the module path for the **`go mod init`** command.

The code path to be published in the *private* repository is pointed out by the IAP Go installer after successful installation. For example, you can publish the code under `/opt/protegrity/applicationprotector/go/src/privaterepo.example.com/app` directory.

Caution: For the IAP Go module to work accurately, the Protegrity Native Library (*gopepprovider.plm*) is included in the `/opt/protegrity/applicationprotector/go/src/privaterepo.example.com/app` directory. This library will also be a part of the published code.

f. To use the IAP Go module in your application, you need to add the import path (the repository path where the IAP Go module is published) as per your repository path. For example,

```
import "privaterepo.example.com/app/apgo"
```

For more information about how to configure Go to use the private repository module, refer to the section *Appendix B: Using the Go Module with a Private GitLab Repository* in the *Protegrity Application Protector Guide 9.2.0.0*.

Using the GOPATH approach:

a. Run the IAP Go installer using the following command.

```
./APGoIMSetup_Linux_x64_<version>.sh
```

The prompt to continue the installation appears.

```
*****************************************************
 Welcome to the Defiance DPS Immutable AP Go API Setup Wizard
*****************************************************

This will install Immutable AP Go API on your computer.
Do you want to continue? [yes or no]
```

b. If you want to continue with the installation of the IAP Go, then type *yes* else type *no*.

If you type *yes*, then the prompt to enter the installation directory appears.

```
Please enter installation directory
[/opt/protegrity]:
```

If you type *no*, then the installation of the IAP Go aborts.

c. After you enter the installation directory, the following prompt appears.

```
Would you like to install AP Go as a Go Module? [yes or no]
```

d. Type *no* to install the IAP Go using the GOPATH approach.

e. Set the following environment variables to run the IAP Go application:

```
export GOPATH=$GOPATH:/opt/protegrity/applicationprotector/go/
export CGO_CFLAGS="-I/opt/protegrity/applicationprotector/go/include"
export CGO_LDFLAGS="-L/opt/protegrity/applicationprotector/go/lib"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/protegrity/applicationprotector/go/lib
```

The IAP Go is installed in the */opt/protegrity* directory using the GOPATH approach.

f. To use the IAP Go package in your application, you need to add the import path. For example,

```
import "protegrity.com/apgo"
```

The default installation directory for the IAP Go on the Linux or Unix platform is listed in the following table.

Table 6-1: IAP Go Installation Directory

Platform	Directory
Linux/Unix	*/opt/protegrity/applicationprotector/go*

6.1.4 Running the REST Client utility on the Policy Package Creator Machine

This section describes how to run the cURL utility on the Policy Package Machine.

To run the cURL utility:

1. Run the following command to know the PEP version supported by the ESA:
   curl -k -v -u "<user>:<password>" -X GET https://<ESA_IP_ADDRESS>/pty/v1/imp/version

   Sample output:

   ```
   {
     "version" : "<version>",
     "buildVersion" : "<build version>",
     "pepVersions" : [ "<pepVersions supported by the ESA>" ]
   }
   ```

   Note: Ensure that the password complexity meets OWASP recommendations. For more information about the password requirements, refer to section *Password Strength*.

2. Run the following command to fetch the policy dump from the ESA:

   ```
   curl -k -v --location 'https: //<ESA_IP>/pty/v1/imp/export' \
   -u "<USER>:<PASSWORD>" \
   --header 'Content-Type: application/json' \
   --data '{
       "name": "<Request_Name>",
       "dataStoreName": "<DataStoreName>",
       "pepVersion": "<pepVersion>",
       "emptyString":"NULL",
       "caseSensitive":"YES",
       "output":"STDOUT",
       "kek": {
           "pkcs5": {
               "password": "<PASSWORD>",
               "salt": "<SALT>"
           }
       }
   }' -o <Policy_Package_Name>.json
   ```

   Note: Ensure that the output parameter is set to *STDOUT* to avoid the loss of logs.

   Note: Ensure that the value passed for the *pepVersion* parameter in the command matches with the *pepVersion* parameter in the *manifest.json* file.

   Note: The values passed for the parameters in the command must match the corresponding values in the ESA.

   Note: It is recommended that the *<USER>* exporting the policy should have minimum privileges. The least privilege that is required is the *Export IMP* permissions on the ESA.

   An encrypted policy package file is generated at the local path defined by the user.

3. Ensure that you have copied the encrypted policy package to the Policy Package Consuming Machine.

6.1.5 Running the PolicyUtil Binary on the Policy Consuming Machine

This section describes how to run the *import* utility on the Policy Package Consuming Machine.

To run the *import* utility:

1. Run the following command to install the PolicyUtil setup, which has the *impgen2* binary.

   ```
   ./PolicyUtilSetup_Linux_x64_<version>.sh
   ```

2. The following directory is created in the `/opt/protegrity` directory.

   `/opt/protegrity/imp-policy-util`

3. Navigate to the `/opt/protegrity/imp-policy-util/bin/impgen2` directory available on the Policy Package Consuming Machine.
4. Run the *impgen2* binary by using the following command.

   ```
   ./impgen2 import -passphrase <password> -salt <salt> -policy-file <file_path>/
   <policy_name>.json -pepversion <pepversion>
   ```

   Note: The IMP package PEP version must match the protector PEP version.

   Note:
   Passphrase and salt can either be given as clear texts or as environment variables.

   For example: *PTY_SALT* or *PTY_SALT_PATH*.

   If a file is used for storing passphrase and salt, then ensure that there is no new line in it. For more information regarding the binary, run `./impgen2 help`.

   The *PolicyUtil* binary repopulates the shared memory on the Policy Package Consuming Machine.

5. Run the following command to view the populated shared memory on the Policy Package Consuming Machine.

   ```
   ipcs -a
   ```

   After the shared memory is populated, the protector can be used to perform the protect, unprotect, and reprotect operations.

6.1.6 Verifying the Policy Health on the Policy Package Consuming Machine

This section describes how to verify the policy health on the Policy Package Consuming Machine.

To verify the policy health:

1. Navigate to the `/opt/protegrity/imp-policy-util/bin/` directory.
2. Run the following command to verify the policy health.

   ```
   ./impgen2 health
   ```

   The following output is expected.

   ```
   health check status: healthy
   ```

6.1.7 Uninstalling PolicyUtil Binary from Policy Package Consuming Machine

This section describes how to uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine.

➤ To uninstall the *PolicyUtil* binary from the Policy Package Consuming Machine:

1. Navigate to the `/opt/protegrity/imp-policy-util` directory.
2. Remove the `/imp-policy-util` directory.

6.1.8 Uninstalling the Immutable Application Protector Go from Linux

This section describes how to uninstall the IAP Go from the Linux or Unix platform.

1. Navigate to the `/opt/protegrity/applicationprotector/go` directory.
2. Delete the `/go` directory.

The IAP Go in uninstalled.

6.1.9 Running IAP Go - Example

This section provides examples on how to use the IAP Go protect, unprotect, and reprotect APIs. It is assumed that policies are made available to the policy consuming machine in a decrypted form and the IAP Go has been successfully installed.

For more information on running the IAP Go sample application, refer to the section *Running IAP - Example*.

Chapter 7

Running IAP - Example

This section provides examples on how to use the Immutable Application Protector protect, unprotect, and reprotect APIs. It is assumed that policies are made available to the Policy Package Consuming Machine in a decrypted form and the IAP has been successfully installed.

The tasks can be divided in the following order.

1. Create the data elements and data store in the Policy Management on the ESA Web UI.
2. Create the member sources and roles and configure the policy.
3. Configure a policy.
4. Configure the trusted application.
5. Add a trusted application to a data store.
6. Populate the policy from the ESA to the policy consumption machine.
7. Install the Immutable Application Protector.
8. Use the protection and unprotection APIs by first protecting the string array and then unprotecting the protected value.

7.1 Creating Data Elements and Data Stores in Policy Management

This section describes how to create a data element and a data store on the ESA.

Before you begin
Before you run the application, decide on how you would like to protect the data – using encryption or tokenization. Protection and unprotection methods are available for both encryption and tokenization.

➤ To create the data elements and the data stores:

1. To create a data element, navigate to the ESA Web UI, then to **Policy Management** > **Data Elements & Masks** > **Data Elements**.

   For more details about creating data elements, refer to the section *Working With Data Elements* in the *Policy Management Guide 9.1.0.4*.

2. To create a data store, navigate to the ESA Web UI, then to **Policy Management** > **Data Stores**.

   For more details about creating data stores, refer to the section *Working with Data Stores* in the *Policy Management Guide 9.1.0.4*.

7.2 Creating Member Sources and Roles

This section describes how to create a member source and a role on the ESA.

➤ To create the member sources and roles:

1. To create a member source, navigate to the ESA Web UI, then to **Policy Management** > **Roles & Member Sources** > **Member Sources**.

   For more information about creating the member sources, refer to the section *Working with Member Sources* in the *Policy Management Guide 9.1.0.4*.

2. To create a role, navigate to the ESA Web UI, then to **Policy Management** > **Roles & Member Sources** > **Roles**.

   For more information about creating the roles, refer to the section *Working with Roles* in the *Policy Management Guide 9.1.0.4*.

7.3 Configuring a Policy

This section describes how to configure a policy on the ESA.

➤ To configure a new policy:

1. On the ESA Web UI, navigate to **Policy Management** > **Policies & Trusted Applications** > **Policies**.
2. Click **Add New Policy**.

   The **New Policy** screen appears.

3. After the policy is configured for the application user, add the permissions, data elements, roles, and data stores to the policy and then save it.
4. Deploy the policy using the Policy Management Web UI.

   For more information about creating a data security policy, refer to the section *Creating and Deploying Policies* in the *Policy Management Guide 9.1.0.4*.

7.4 Configuring a Trusted Application

This section describes how to configure a trusted application on the ESA.

Before you begin

To ensure that only the applications and users configured in the ESA are able to access the IAP APIs, these have to be configured as trusted applications under the ESA security policy. If a policy is deployed but the application or the user is not trusted, then while performing any protect or unprotect operations, an error message - `API consumer is not part of the trusted applications, please contact the Security Officer` - appears and the IAP aborts.

> **Note:** The IAP C does not support trusted application.

To configure a new trusted application:

1. On the ESA Web UI, navigate to **Policy Management** > **Policies & Trusted Applications** > **Trusted Application**.
2. Create a trusted application.

   > **Note:**
   > Ensure that the application name of the Trusted Application matches with the public class name to successfully use the APIs.
   >
   > The application username must be configured as the user invoking the sample IAP application.

3. Deploy the trusted application using the Policy Management Web UI.

   For more information about the trusted applications, refer to the section *Working with Trusted Applications* in the *Policy Management Guide 9.1.0.4*.

7.5 Adding a Trusted Application to a Data Store

This section describes how to add a trusted application to a data store on the ESA.

To add a trusted application to a data store:

1. On the ESA Web UI, navigate to **Policy Management** > **Data Stores**.

   The list of all the data stores appear.
2. Select the required data store.

   The screen to edit the data store appears.
3. Under the **Trusted Applications** tab, click **Add**.

   The screen to add the trusted application appears.
4. Select the required trusted application and click **Add**.
5. Select the required policy and deploy it using the Policy Management Web UI.

   For more information about adding a trusted application to a data store, refer to the section *Adding Trusted Applications to the Data Store* in the *Policy Management Guide 9.1.0.4*.

7.6 Creating and Using the Policy Package on the Policy Consuming Machine

This section describes how to use the policy package on the policy consuming machine. This step is applicable only for the Linux IAP deployment where the *IMPS Export* API is used to fetch the policy package instead of the *PtyShmCat* utility.

➤ To use the policy package on the policy consuming machine:

1. Run the REST Client command on the policy package creator machine.

   For more information about running REST Client command, refer to the section *Running the REST Client utility on the Policy Package Creator Machine*.
2. Ensure that you have copied the encrypted policy package created in the policy package creator machine to the policy consuming machine.
3. Run the PolicyUtil binary on the policy consuming machine.

   For more information about running the PolicyUtil binary, refer to the section *Running the PolicyUtil Binary on the Policy Consuming Machine*.

7.7 Installing Immutable Application Protector

This section describes how to install the Immutable Application Protector.

7.7.1 Installing Immutable Application Protector C

This section describes how to install the IAP C.

➤ To install the IAP C:

1. The IAP C must be successfully installed to protect or unprotect sensitive information.

   For more information about installing the IAP C, refer to the section *Installing Immutable Application Protector (IAP) C*.
2. To verify that the software is installed correctly, run the **GetVersion** method to check the version of the installed IAP C.

   The following code snippet can be used to check the version of the IAP C.

```
/* Illustrates how to call getVersion() api to know the version of Immutable Application
protector
*/

#include <stdio.h>
#include <string.h>
#include "xcapi.h"

if( XC_SUCCESS == XCGetVersion( szVersion, sizeof( szVersion ) ) )
  {
    printf( "%s\n", szVersion );
  }

  rReturnCode = XCInitLib( &phXCHandle, "" );
  /* Initialzie the API */
  if( XC_SUCCESS ==  rReturnCode )
```

7.7.2 Installing Immutable Application Protector Java

This section describes how to install the IAP Java.

➤ To install the IAP Java:

1. The IAP Java must be successfully installed to protect or unprotect sensitive information.

   For more information about installing the IAP Java, refer to the section *Installing Immutable Application Protector (IAP) Java*.
2. To verify that the software is installed correctly, run the **`GetVersion`** method to check the version of the installed IAP Java.

   public java.lang.String getVersion()

   The following code snippet can be used to check the version of the IAP Java.

```
/* Illustrates how to call getVersion() api to know the version of Immutable Application
protector
* Executing this for the first time creates a forensic entry that should be
* added to the authorized app
*
* Compiled as  : javac -cp ApplicationProtectorJava.jar AP_Java_getVersion
* Run as       : java -cp  ApplicationProtectorJava.jar AP_Java_getVersion
*
*/

import com.protegrity.ap.java.*;

public class AP_Java_getVersion {

    public static void main(String[] args) throws ProtectorException {

        Protector protector=null;
        try {
            protector=Protector.getProtector();
            System.out.println("Product version : "+protector.getVersion());
        } catch (ProtectorException e) {
            e.printStackTrace();
            throw e;
        }

   }
}
```

7.7.3 Installing Immutable Application Protector Python

This section describes how to install the IAP Python.

To install the IAP Python:

1. The IAP Python must be successfully installed to protect or unprotect sensitive information.

   For more information about installing the IAP Python, refer to the section *Installing Immutable Application Protector (IAP) Python*.
2. To verify that the software is installed correctly, run the **`GetVersion`** method to check the version of the installed the IAP Python.

   The following code snippet can be used to check the version of the IAP Python.

```
/from appython import Protector
protector = Protector()
session = protector.create_session("PolicyUser")
print(protector.get_version())
```

7.7.4 Installing Immutable Application Protector Go

This section describes how to install the IAP Go.

To install the IAP Go:

1. The IAP Go must be successfully installed to protect or unprotect sensitive information.
   For more information about installing IAP Go, refer to the section *Installing Immutable Application Protector (IAP) Go*.
2. To verify that the software is installed correctly, run the **`GetVersion`** method to check the version of the installed IAP Go.
   The following code snippet can be used to check the version of the IAP Go.

```
package main
import (
"fmt"
apgo "protegrity.com/apgo"
)
func main() {
fmt.Printf("AP Go Version: %s\n\n", apgo.GetVersion())
}
```

7.8 Using the IAP APIs

This section describes how to use the IAP using a sample application.

7.8.1 Using the IAP C APIs

This section describes how to use IAP C APIs using a sample application of the Linux and Windows platforms.

7.8.1.1 Using the IAP C APIs on Linux

This section describes how to use the IAP C APIs on a Linux platform using a sample application.

After you have set up the the policy, you can begin testing the IAP C APIs for protection, unprotection, and reprotection.

To run the sample code:

1. Copy and rename *xcpep_imp.plm* to *libxcpep.so* in the same directory as *<installation_directory>/defiance_xc/bin*.
2. Run the following commands:

```
gcc -g -o sample -DUNIX -I /opt/protegrity/defiance_xc/include -L/opt/protegrity/
defiance_xc/bin -lxcpep_imp -lpthread sample.c

export LD_LIBRARY_PATH=/opt/protegrity/defiance_xc/bin/

./sample 0 0 0 USER1 AES128
```

For more information about the errors you might encounter while using the IAP C on the Linux platform, refer to the section *Immutable Application Protector (IAP) On-Premise Errors* in the *Protegrity Troubleshooting Guide 9.2.0.0*.

The following is a sample application for the IAP C.

```
/*
* Sample program to protect sensitive data
* Read xcapi.h for more instructions
*/
```

```
#include <stdio.h>
#include <string.h>
#include "xcapi.h"
int main( int argc, char* argv[] )
{
    /* Handle to the XC library */
    XC_HANDLE phXCHandle = XC_NULL;
    /* Handle to the XC session */
    XC_SESSION phSession = XC_NULL;
    /* The parameter can have different values, read xcapi.h */
    XC_CHAR szParameter[512] = {0};
    /* User to decrypt keyexport.dat file */
    XC_CHAR szUser[512] = {0};
    /* Password to decrypt keyexport.dat file */
    XC_CHAR szPassword[512] = {0};
    /* User with appropriate privileges in the DPS policy */
    XC_CHAR szPolicyUser[512] = {0};
    /* Data element from the DPS policy */
    XC_CHAR szDataElement[512] = {0};
    /* Character of data read from the terminal */
    char szBuf[512] = {0};
    /* Buffer that will hold the cipher text of the character */
    /* data read from the terminal */
    XC_BYTE bOutputData[512 + XC_BYTES_OVERHEAD] = {0};
    /* Resulting cipher text length */
    XC_UINT4 ui4OutPutDataLength = sizeof( bOutputData );
    /* Buffer that will hold the clear text of the character */
    /* data decrypted from the encrypted buffer */
    XC_BYTE bOutputData2[512 + XC_BYTES_OVERHEAD] = {0};
    /* Resulting clear text length */
    XC_UINT4 ui4OutPutData2Length = sizeof( bOutputData );
    /* Input data length */
    XC_UINT4 ui4InPutDataLength = 0;
    /* Loop counter */
    XC_UINT4 ui4Count = 0;
    /* Version info */
    XC_CHAR szVersion[512] = {0};
    /* Return Code */
    XC_RETURNCODE rReturnCode = XC_FAILED;
    XC_PARAM_EX stXCParamEx = {0};
    XC_PARAM_EX stXCParamExDec = {0};
    stXC_ACTION_RESULT stActionResult;
    stXC_ACTION_RESULT stActionResultDec;
    XC_BYTE bOutNullInd = XC_FALSE;
    if( 6 != argc )
    {
        printf( "Usage: xcencxclite keyexportfile user password policyUser DataElement\n" );
        printf( "Example: xcencxclite ./keyexport.dat eipuser eippw defaultuser1 CCN\n\n" );
        printf( "Usage: xcencxcpep communicationId user password policyUser DataElement\n" );
        printf( "Example: xcencxcpep 0 notused notused defaultuser1 CCN\n" );
        printf( "Note: Supply user and password even though not used.\n\n" );
        printf( "Usage: xcencxcclient host:port:protocol user password policyUser
DataElement\n" );
        printf( "Example: xcencxcclient 127.0.0.1:15910:tcp notused notused defaultuser1
CCN\n" );
        printf( "Example: xcencxcclient 127.0.0.1:15920:ssl certuser certpw defaultuser1
CCN\n" );
        printf( "Note: Supply user and password even though not used when protocol is
tcp.\n\n" );
        return( 0 );
    }
    if( sizeof( szParameter ) > strlen( argv[1] ) )
    {
        memcpy( szParameter, argv[1], strlen( argv[1] ) );
    }
    if( sizeof( szUser ) > strlen( argv[2] ) )
    {
        memcpy( szUser, argv[2], strlen( argv[2] ) );
    }
    if( sizeof( szPassword ) > strlen( argv[3] ) )
    {
        memcpy( szPassword, argv[3], strlen( argv[3] ) );
    }
    if( sizeof( szPolicyUser ) > strlen( argv[4] ) )
    {
        memcpy( szPolicyUser, argv[4], strlen( argv[4] ) );
    }
    if( sizeof( szDataElement ) > strlen( argv[5] ) )
    {
        memcpy( szDataElement, argv[5], strlen( argv[5] ) );
    }
     /* Get version info */
    if( XC_SUCCESS == XCGetVersion( szVersion, sizeof( szVersion ) ) )
    {
        printf( "%s\n", szVersion );
    }
    rReturnCode = XCInitLib( &phXCHandle, "" );
    /* Initialze the API */
    if( XC_SUCCESS == rReturnCode )
    {
        /* Read character string from the terminal */
        printf( "Enter a string to encrypt and press enter: " );
        fgets( szBuf, 50, stdin );
        printf( "\n" );
        ui4InPutDataLength = strlen( szBuf );
        /* Remove the last input '\n' */
        if( strlen( szBuf ) < 49 )
        {
            ui4InPutDataLength -= 1;
            szBuf[ui4InPutDataLength] = '\0';
        }
        printf( "Open session to XC..." );
        rReturnCode = XCOpenSession( phXCHandle,
                                     szUser,
                                     szPassword,
                                     szParameter,
                                     &phSession );
        if( XC_SUCCESS == rReturnCode )
        {
            printf( "OK\n" );
            printf( "Encrypting..." );

            stXCParamEx.ui4DataType = XC_DATATYPE_BYTE;
            stXCParamEx.ui4Operation = XC_ANY_FUNCTION;

            rReturnCode = XCProtect( phXCHandle,
                                         phSession,
                                         XC_TRUE,
                                         szPolicyUser,
                                         szDataElement,
                                         XC_NULL,
                                         0,
                                         ( XC_BYTE* )szBuf,
                                         ui4InPutDataLength,
                                         XC_FALSE,
                                         bOutputData,
                                         &ui4OutPutDataLength,
                                         &bOutNullInd,
                                         &stXCParamEx,
                                         sizeof( stXCParamEx ),
                                         &stActionResult );
            /* Protect the characters read from the terminal */
            if( XC_SUCCESS == rReturnCode && XC_LOGRETURNSUCCESS ==
stActionResult.ui4LogSeverity )
            {
                printf( "OK\n" );
                printf( "Cipher text: " );
                for( ui4Count = 0; ui4Count < ui4OutPutDataLength; ui4Count++ )
                {
                    printf( "%02x", bOutputData[ui4Count] );
                }
                printf( "\n" );
                printf( "Decrypting..." );

                stXCParamExDec.ui4DataType = XC_DATATYPE_BYTE;
                stXCParamExDec.ui4Operation = XC_ANY_FUNCTION;
                rReturnCode = XCUnprotect( phXCHandle,
                                            phSession,
                                            XC_TRUE,
                                            szPolicyUser,
                                            szDataElement,
                                            XC_NULL,
                                            0,
                                            ( XC_BYTE* )bOutputData,
                                            ui4OutPutDataLength,
                                            XC_FALSE,
                                            bOutputData2,
                                            &ui4OutPutData2Length,
                                            &bOutNullInd,
                                            &stXCParamExDec,
                                            sizeof( stXCParamExDec ),
                                            &stActionResultDec );
                /* Protect the characters read from the terminal */
                if( XC_SUCCESS == rReturnCode && XC_LOGRETURNSUCCESS ==
stActionResult.ui4LogSeverity )
                {
                    printf( "OK\n" );
                    printf( "Clear text: " );
                    /* Null terminate */
                    bOutputData2[ui4OutPutData2Length] = '\0';
                    printf( "%s\n", bOutputData2 );
                }
                else
                {
                    printf( "\nFailed to encrypt:error code:%d\n", rReturnCode );
                }
            }
            else
            {
                printf( "\nFailed to encrypt:error code:%d\n", rReturnCode );
            }
        }
        else
        {
            printf( "\nFailed to open session:error code:%d\n", rReturnCode );
        }
        XCCloseSession( phXCHandle, &phSession );
    }
    else
    {
        printf( "Failed to initialize XC API, error code:%d\n", rReturnCode );
    }
    XCTerminateLib( &phXCHandle );
    return 0;
}
```

7.8.1.2 Using the IAP C APIs on Windows

This section describes how to use the IAP C APIs on a Windows platform using a sample application.

After you have set up the the policy, you can begin testing the IAP C APIs to perform protect, unprotect, and reprotect operations.

To run the sample code:

1. Open the x64 Native Tools Command Prompt provided by the Visual Studio application.
2. Set the path of the *xcpep_imp.plm* file to the PATH system variable.

   For example:

   ```
   set PATH=%PATH%;C:\Program Files\Protegrity\Defiance_XC_IM\bin
   ```

3. Run the following command to compile the sample code.

```
cl /I "C:\Program Files\Protegrity\Defiance_XC_IM\include" /D "_WIN64" /MT
"C:\Program Files\Protegrity\Defiance_XC_IM\lib\xcpep_imp.lib" sample.c
```

A *sample.exe* executable file is created.

4. Run the following sample command to perform the protect, unprotect, and reprotect operations.

```
sample.exe 0 0 0 USER1 AES128
```

Note:
If you want to see a list of available parameters, then run the following command.

```
sample.exe -h
```

For more information about the errors you might encounter while using the IAP C on the Windows platform, refer to the section *Immutable Application Protector (IAP) On-Premise Errors* in the *Protegrity Troubleshooting Guide 9.2.0.0*.

The following is a sample application for the IAP C.

```
/*
* Sample program to protect sensitive data
* Read xcapi.h for more instructions
*/

#include <stdio.h>
#include <string.h>
#include "xcapi.h"
int main( int argc, char* argv[] )
{
    /* Handle to the XC library */
    XC_HANDLE phXCHandle = XC_NULL;
    /* Handle to the XC session */
    XC_SESSION phSession = XC_NULL;
    /* The parameter can have different values, read xcapi.h */
    XC_CHAR szParameter[512] = {0};
    /* User to decrypt keyexport.dat file */
    XC_CHAR szUser[512] = {0};
    /* Password to decrypt keyexport.dat file */
    XC_CHAR szPassword[512] = {0};
    /* User with appropriate privileges in the DPS policy */
    XC_CHAR szPolicyUser[512] = {0};
    /* Data element from the DPS policy */
    XC_CHAR szDataElement[512] = {0};
    /* Character of data read from the terminal */
    char szBuf[512] = {0};
    /* Buffer that will hold the cipher text of the character */
    /* data read from the terminal */
    XC_BYTE bOutputData[512 + XC_BYTES_OVERHEAD] = {0};
    /* Resulting cipher text length */
    XC_UINT4 ui4OutPutDataLength = sizeof( bOutputData );
    /* Buffer that will hold the clear text of the character */
    /* data decrypted from the encrypted buffer */
    XC_BYTE bOutputData2[512 + XC_BYTES_OVERHEAD] = {0};
    /* Resulting clear text length */
    XC_UINT4 ui4OutPutData2Length = sizeof( bOutputData );
    /* Input data length */
    XC_UINT4 ui4InPutDataLength = 0;
    /* Loop counter */
    XC_UINT4 ui4Count = 0;
    /* Version info */
    XC_CHAR szVersion[512] = {0};
    /* Return Code */
    XC_RETURNCODE rReturnCode = XC_FAILED;
    XC_PARAM_EX stXCParamEx = {0};
    XC_PARAM_EX stXCParamExDec = {0};
    stXC_ACTION_RESULT stActionResult;
    stXC_ACTION_RESULT stActionResultDec;
    XC_BYTE bOutNullInd = XC_FALSE;
    if( 6 != argc )
    {
        printf( "Usage: xcencxclite keyexportfile user password policyUser DataElement\n" );
        printf( "Example: xcencxclite ./keyexport.dat eipuser eippw defaultuser1 CCN\n\n" );
        printf( "Usage: xcencxcpep communicationId user password policyUser DataElement\n" );
        printf( "Example: xcencxcpep 0 notused notused defaultuser1 CCN\n" );
        printf( "Note: Supply user and password even though not used.\n\n" );
        printf( "Usage: xcencxcclient host:port:protocol user password policyUser
DataElement\n" );
        printf( "Example: xcencxcclient 127.0.0.1:15910:tcp notused notused defaultuser1
CCN\n" );
        printf( "Example: xcencxcclient 127.0.0.1:15920:ssl certuser certpw defaultuser1
CCN\n" );
        printf( "Note: Supply user and password even though not used when protocol is
tcp.\n\n" );
        return( 0 );
    }
    if( sizeof( szParameter ) > strlen( argv[1] ) )
    {
        memcpy( szParameter, argv[1], strlen( argv[1] ) );
    }
    if( sizeof( szUser ) > strlen( argv[2] ) )
    {
        memcpy( szUser, argv[2], strlen( argv[2] ) );
    }
    if( sizeof( szPassword ) > strlen( argv[3] ) )
    {
        memcpy( szPassword, argv[3], strlen( argv[3] ) );
    }
    if( sizeof( szPolicyUser ) > strlen( argv[4] ) )
    {
        memcpy( szPolicyUser, argv[4], strlen( argv[4] ) );
    }
    if( sizeof( szDataElement ) > strlen( argv[5] ) )
    {
        memcpy( szDataElement, argv[5], strlen( argv[5] ) );
    }
     /* Get version info */
    if( XC_SUCCESS == XCGetVersion( szVersion, sizeof( szVersion ) ) )
    {
        printf( "%s\n", szVersion );
    }
    rReturnCode = XCInitLib( &phXCHandle, "" );
    /* Initialze the API */
    if( XC_SUCCESS == rReturnCode )
    {
        /* Read character string from the terminal */
        printf( "Enter a string to encrypt and press enter: " );
        fgets( szBuf, 50, stdin );
        printf( "\n" );
        ui4InPutDataLength = strlen( szBuf );
        /* Remove the last input '\n' */
        if( strlen( szBuf ) < 49 )
        {
            ui4InPutDataLength -= 1;
            szBuf[ui4InPutDataLength] = '\0';
        }
        printf( "Open session to XC..." );
        rReturnCode = XCOpenSession( phXCHandle,
                                     szUser,
                                     szPassword,
                                     szParameter,
                                     &phSession );
        if( XC_SUCCESS == rReturnCode )
        {
            printf( "OK\n" );
            printf( "Encrypting..." );
            stXCParamEx.ui4DataType = XC_DATATYPE_BYTE;
            stXCParamEx.ui4Operation = XC_ANY_FUNCTION;

            rReturnCode = XCProtect( phXCHandle,
                                        phSession,
                                        XC_TRUE,
                                        szPolicyUser,
                                        szDataElement,
                                        XC_NULL,
                                        0,
                                        ( XC_BYTE* )szBuf,
                                        ui4InPutDataLength,
                                        XC_FALSE,
                                        bOutputData,
                                        &ui4OutPutDataLength,
                                        &bOutNullInd,
                                        &stXCParamEx,
                                        sizeof( stXCParamEx ),
                                        &stActionResult );
            /* Protect the characters read from the terminal */
            if( XC_SUCCESS == rReturnCode && XC_LOGRETURNSUCCESS ==
stActionResult.ui4LogSeverity )
            {
                printf( "OK\n" );
                printf( "Cipher text: " );
                for( ui4Count = 0; ui4Count < ui4OutPutDataLength; ui4Count++ )
                {
                    printf( "%02x", bOutputData[ui4Count] );
                }
                printf( "\n" );
                printf( "Decrypting..." );

                stXCParamExDec.ui4DataType = XC_DATATYPE_BYTE;
                stXCParamExDec.ui4Operation = XC_ANY_FUNCTION;

                rReturnCode = XCUnprotect( phXCHandle,
                                            phSession,
                                            XC_TRUE,
                                            szPolicyUser,
                                            szDataElement,
                                            XC_NULL,
                                            0,
                                            ( XC_BYTE* )bOutputData,
                                            ui4OutPutDataLength,
                                            XC_FALSE,
                                            bOutputData2,
                                            &ui4OutPutData2Length,
                                            &bOutNullInd,
                                            &stXCParamExDec,
                                            sizeof( stXCParamExDec ),
                                            &stActionResultDec );
                /* Protect the characters read from the terminal */
                if( XC_SUCCESS == rReturnCode && XC_LOGRETURNSUCCESS ==
stActionResult.ui4LogSeverity )
                {
                    printf( "OK\n" );
                    printf( "Clear text: " );
                    /* Null terminate */
                    bOutputData2[ui4OutPutData2Length] = '\0';
                    printf( "%s\n", bOutputData2 );
                }
                else
                {
                    printf( "\nFailed to encrypt:error code:%d\n", rReturnCode );
                }
            }
            else
            {
                printf( "\nFailed to encrypt:error code:%d\n", rReturnCode );
            }
        }
        else
        {
            printf( "\nFailed to open session:error code:%d\n", rReturnCode );
        }
        XCCloseSession( phXCHandle, &phSession );
    }
    else
    {
        printf( "Failed to initialize XC API, error code:%d\n", rReturnCode );
    }
    XCTerminateLib( &phXCHandle );
    return 0;
}
```

7.8.2 Using the IAP Java APIs

This section describes how to use the IAP Java APIs using a sample application.

Once the policy is available on the consuming machine, you can begin testing the IAP Java APIs for protection, unprotection, and reprotection.

Note:

When a short running application has completed its execution (in less than a second), the audit logs will not be seen. In such cases, the *protector.flushAudits()* API needs to be invoked.

For more information on the flushAudits API, refer to the section *flushAudits* of the *Protegrity APIs, UDFs, and Commands Reference Guide 9.2.0.0*.

The following is a sample application for the IAP Java.

For more information about the errors you might encounter while using the IAP Java, refer to the section *Immutable Application Protector (IAP) On-Premise Errors* in the *Protegrity Troubleshooting Guide 9.2.0.0*.

```
/* HelloWorld.java
*
* Illustrates how to call the new java api
* Executing this for the first time will create a forensic entry that will
* have to be added to the authorized app
* Compiled as  : javac -cp ApplicationProtectorJava.jar HelloWorld.java
* Run as       :
* java -cp  ApplicationProtectorJava.jar HelloWorld AuthPolicyUser DataElement Data
* java version : 1.8.0_45
* Package Name: IAP_Linux-ALL-64_x86-64_JRE-1.8_9.2.0.0.4.tgz
*
* Use either token elements or DTP2 elements or NoEncryption as DataElement
* while running this code.
*/

import com.protegrity.ap.java.*;

public class HelloWorld {
    public static void main(String[] args) throws ProtectorException {
        if (args.length == 3) {
            System.out.println(" AuthUser : "+args[0]);
            System.out.println(" DataElement : "+args[1]);
            System.out.println(" Clear Text Data : "+args[2]);
            stringTest(args[0],args[1],args[2]);
        } else {
            System.out.println(" Usage : java -cp ApplicationProtectorJava.jar HelloWorld
AuthUser DataElement Data");
            System.out.println(" Example : java -cp ApplicationProtectorJava.jar HelloWorld
USER TKCCN 4111111111111111");
            System.exit(0);
        }
    }
    static public void stringTest(String USR, String DE, String Data) throws
ProtectorException {
        boolean result = false;
        String[] inputStringArray = new String[1];
        String[] ProtectStringArray = new String[1];
        String[] ReProtectStringArray = new String[1];
        String[] UnProtectStringArray = new String[1];
        SessionObject session = null;
        Protector protector = null;
        inputStringArray[0] = Data;

        try {
            /** Instantiate the protector
             * This should be invoked only once in the lifetime of each application that uses
the protector.
             * Protection operations can be performed only on successful initialization of
the protector.
            **/
            protector = Protector.getProtector();
            System.out.println("1.) Protector Instantiated!! ");

            // Set input parameters and create session
            session = protector.createSession(USR);
            System.out.println("2.) Session created ");

            // test protect/reprotect/unprotect methods
            if (!protector.protect(session, DE, inputStringArray, ProtectStringArray))
                System.out.println("protect api failed !!! \nError : " +
protector.getLastError(session));
            else
                System.out.println("3.) Protect Output   : " + ProtectStringArray[0]);

            /*
             * if(!protector.reprotect(session,DE,DE,ProtectStringArray,ReProtectStringArray
             * ))
             * System.out.println("reprotect api failed !!! \nError : "+protector.
             * getLastError(session));
             * else
             * System.out.println( "4.) ReProtect Output: " + ReProtectStringArray[0] );
             * if(!protector.unprotect(session,DE,ReProtectStringArray,UnProtectStringArray)
             * )
             * System.out.println("unprotect api failed !!! \nError : "+protector.
             * getLastError(session));
             * else
             * System.out.println( "4.) UnProtect Output: " + UnProtectStringArray[0] );
             */
            if (!protector.unprotect(session, DE, ProtectStringArray, UnProtectStringArray))
                System.out.println("unprotect api failed !!! \nError : " +
protector.getLastError(session));
            else
                System.out.println("4.) UnProtect Output : " + UnProtectStringArray[0]);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // FlushAudits should be invoked only once at the end of the application
lifecycle.
            protector.flushAudits();
        }
    }
}
```

7.8.3 Using the IAP Python APIs

This section describes how to use the IAP Python APIs using a sample application.

Once the policy is available on the consuming machine, you can begin testing the IAP Python APIs for protection, unprotection, and reprotection.

Note:

When a short running application has completed its execution (in less than a second), the audit logs will not be seen. In such cases, the *flush_audits()* API needs to be invoked.

For more information on the flushAudits API, refer to the section *flush_audits()* of the *Protegrity APIs, UDFs, and Commands Reference Guide 9.2.0.0*.

The following is a sample application for the IAP Python.

For more information about the errors you might encounter while using the IAP Python, refer to the section *Immutable Application Protector (IAP) On-Premise Errors* in the *Protegrity Troubleshooting Guide 9.2.0.0*.

```
# -*- coding: utf-8 -*-
from appython import Protector
if __name__ == "__main__":
# Initialize the protector
protector = Protector()
# Create session with policy user
session = protector.create_session("PolicyUser")
# Protect operation
p_out = session.protect("sampledata123", "AlphaNumeric")
print("Protected Data: %s" %p_out)
# Reprotect operation
r_out = session.reprotect(p_out, "AlphaNumeric", "AlphaNumeric")
print("Reprotected Data: %s" %r_out)
# Unprotect operation
org = session.unprotect(r_out, "AlphaNumeric")
print("Unprotected Data: %s" %org)
```

7.8.4 Using the IAP Go APIs

This section describes how to use the IAP Go APIs using a sample application.

Once the policy is available on the consuming machine, you can begin testing the IAP Go APIs for protection, unprotection, and reprotection.

Note:

When a short running application has completed its execution (in less than a second), the audit logs will not be seen. In such cases, the *FlushAudits()* API needs to be invoked.

For more information on the flushAudits API, refer to the section *FlushAudits API* of the *Protegrity APIs, UDFs, and Commands Reference Guide 9.2.0.0*.

The following is a sample application for the IAP Go.

For more information about the errors you might encounter while using the IAP Go, refer to the section *Immutable Application Protector (IAP) On-Premise Errors* in the *Protegrity Troubleshooting Guide 9.2.0.0*.

```
package main

import (
        "fmt"
        "os"

        apgo "protegrity.com/apgo"
)

func main() {

        dataElement := os.Getenv("TEST_DATA_ELEMENT_NAME")
        policyUsr := os.Getenv("TEST_POLICY_USER")
        if dataElement == "" || policyUsr == "" {
               if dataElement == "" {
                       panic(fmt.Sprint("Env variable TEST_DATA_ELEMENT_NAME is empty. Cannot 
proceed with the tests."))
               }
               if policyUsr == "" {
                       panic(fmt.Sprint("Env variable TEST_POLICY_USER is empty. Cannot 
proceed with the tests."))
               }
        }
        fmt.Printf("Testing with data element - '%v' and policy user - '%v'\n", dataElement, 
policyUsr)

        fmt.Printf("AP Go Version: %s\n\n", apgo.GetVersion())

        // initilize the AP Go Package
        terminate, err := apgo.Init()
        if err != nil {
               panic(fmt.Sprintf("apgo.InitLib() failed. Error: %v", err))
        }
        defer terminate()

        input := "sampledata123"
        fmt.Println("Input Data: ", input)
        // create a new Protection Operation session for policyUsr
        session, err := apgo.NewSession(policyUsr)
        if err != nil {
               panic(fmt.Sprintf("apgo.NewSession() failed. Error: %v", err))
        }

        output, rc, err := session.ProtectStr(input, dataElement)
        if err != nil {
               panic(fmt.Sprintf("apgo.ProtectStr() failed. Return Code: %v, Error: %v", rc, 
err))
        }
        fmt.Printf("Protected Data: %v\n", output)

        org, rc, err := session.UnprotectStr(output, dataElement)
        if err != nil {
               panic(fmt.Sprintf("apgo.UnprotectStr() failed. Return Code: %v, Error: %v", 
rc, err))
        }
        fmt.Printf("Unprotected Data: %v\n\n", org)

        blkInput := []string{"hello", "world", "Protegrity"}
        fmt.Printf("Original Bulk Data: %v, len: %v\n", blkInput, len(blkInput))
        blkOut, retCodes, err := session.ProtectStrBulk(blkInput, dataElement)
        if err != nil {
               panic(fmt.Sprintf("apgo.ProtectStrBulk() failed. Return Codes: %v, Error: %v", 
retCodes, err))
        }
        fmt.Printf("Protected Bulk Data: %v, len: %v\n", blkOut, len(blkOut))

        blkOrg, retCodes, err := session.UnprotectStrBulk(blkOut, dataElement)
        if err != nil {
                panic(fmt.Sprintf("apgo.UnprotectStrBulk() failed. Return Codes: %v, Error: 
%v", retCodes, err))
        }
        fmt.Printf("Original Bulk Data: %v, len: %v\n\n", blkOrg, len(blkOrg))

}
```