Protegrity Database Protector Guide 9.1.0.0

Created on: Aug 8, 2024

Copyright

Table of Contents

Chapter 1

Introduction to this Guide

This Protegrity Database Protector Guide discusses about the Database Protector and its uses. The Guide also provides detailed information and examples of libraries and deployment architectures for all flavors of the Protegrity Database Protector.

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 Database Protector Installation Overview* introduces you to the concepts of the Database Protector.
- *Section 3 MS SQL Server Database Protector* provides detailed information about installing and configuring the MS SQL Database Protector.
- *Section 4 Netezza Database Protector* provides detailed information about installing and configuring the Netezza Database Protector.
- *Section 5 Oracle Database Protector* provides detailed information about installing and configuring the Oracle Database Protector.
- *Section 6 Greenplum Database Protector* provides detailed information about installing and configuring the Greenplum Database Protector.
- *Section 7 Teradata Database Protector* provides detailed information about installing and configuring the Teradata Database Protector.
- *Section 8 DB2 Open Systems Database Protector* provides detailed information about installing and configuring the DB2 Open Systems Database Protector.
- *Section 9 Trino Protector* provides detailed information about installing and configuring the Trino Protector.
- *Section 10 Supported Database Protectors Matrix* provides information about the supported database version and platforms for the database protectors.

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.

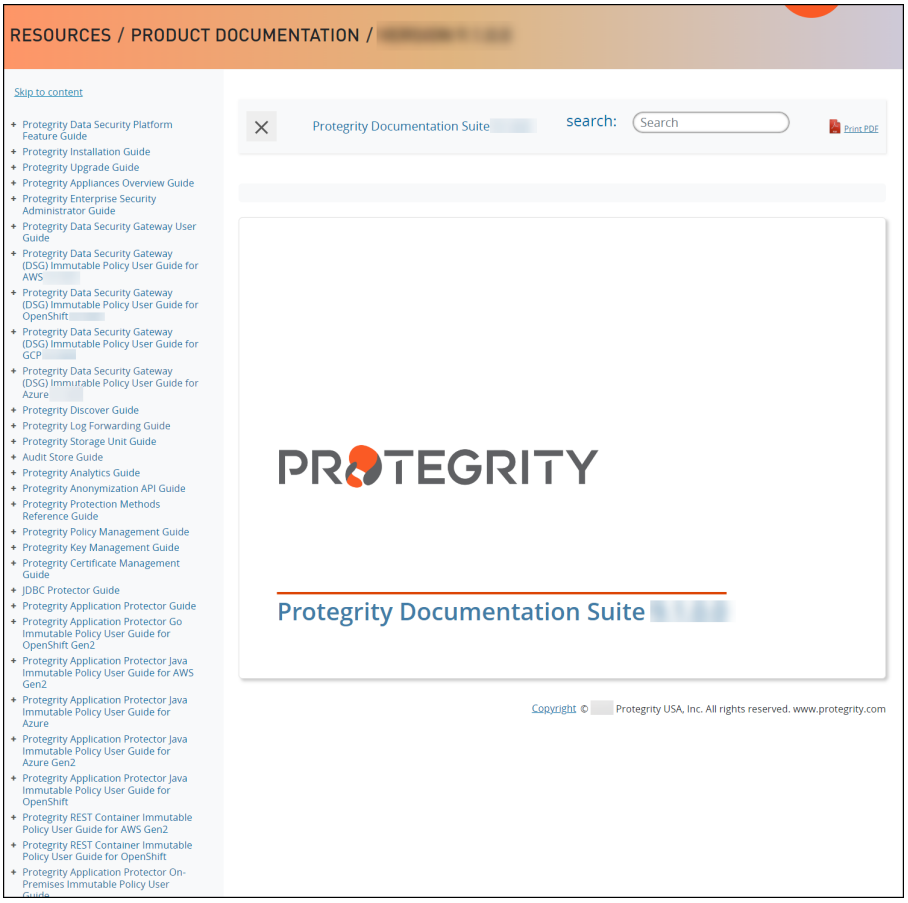

Figure 1-1: Documentation

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

Database Protector Installation Overview

The Database Protector provides protection for sensitive data stored in database tables. This column-level protection is tightly integrated into the database technology, which gives it the performance required to minimize any overhead imposed by data protection.

Protegrity provides Database Protectors for these platforms:

- Oracle
- MS SQL Server
- Teradata
- Greenplum
- IBM Netezza
- DB2 Open Systems
- Trino Protector

Note: "Open Systems" term refers to Windows, AIX and Linux.

Note: The Database Protector supports user names that are up to 255 characters in length. If the user name lengths supported by the various platforms are lower than 255 characters due to platform limitations, then the platform supported user name lengths are considered.

2.1 Setup Overview for Windows and UNIX

Before setting up the Database Protector, ensure that your configuration meets the minimum requirements. The following table describes the environment requirements.

Table 2-1 Environment Requirements

Configuration	Free Disk Space on Windows	Free Disk Space on UNIX
PEP server	50 MB per node	50 MB per node
User Defined Functions (UDFs) and Procedures	10 MB	10 MB
RAM	4 GB	4 GB

Note: For systems under heavy load with auditing turned on, the space required for the PEP Server (20 MB per node) can be higher.

These are the additional requirements:

- You need Administrator rights in the DBMS.
- Make sure that the Database Server is up and running.

The following figure shows the task flow for setting up the Database Protector on Windows and UNIX.

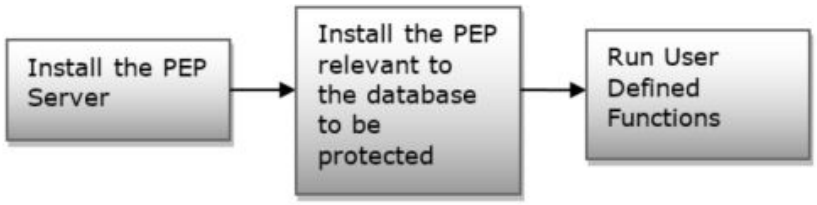

Figure 2-1: Database Protector Setup Task Flow for Windows and UNIX

> **Note:** CLR-based Microsoft SQL Server protector requires additional configuration. For more information on different database platforms, refer to the respective sections in this document.

2.1.1 Database Protector on Windows

When you install database protector, the system automatically sets up a directory structure with the appropriate files. The default location is C:\Program Files\Protegrity\Defiance DPS\pep directory, but you can specify an alternative directory path during installation. The following table describes the default directory structure.

Directory	Description
`\pep`	This directory contains the `*.plm` and `*.lib` files for all installed PEPs. `*.lib` files are provided only on Windows installations. `*.plm` files are Protegrity Loadable Modules, which are identical to `.dll` files in Windows and `.so` files in UNIX.
`\sqlscripts`	This directory contains the subdirectories relevant to each database, which include the required scripts to perform changes to tables and views in the database.
`sqlscripts\db2`	For DB2: this directory contains the following scripts: `createobjects.sql` and `dropobjects.sql`.
`sqlscripts\oracle`	For Oracle: this directory contains the following scripts:`createobjects.sql`, `createobjects_compat.sql`, and `dropobjects.sql`. `*compat.sql` script contains the old UDF definitions that are compatible with releases 6.0.3 and earlier.
`sqlscripts\sqlserver`	For MS SQL Server: this directory contains the following scripts: `dropobjects.sql`,`createassembly`, and `createfunctions`.
`pep\registry`	For MS SQL Server: this directory is created by the SQL Server and later PEP installer containing a `registry.reg` file which can be used to configure the PEP behavior in regards with MRU settings, and appending (yes or no) the domain name to the username.

The scripts folders are created based upon the type of database platform installed.

2.1.1.1 Installing Database Protector on Windows

To install the Database Protector on Windows:

Before you begin

> **Note:**
> Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.
>
> For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Policy Management Guide 9.1.0.0*.

1. Double-click or run the ***PEP<database>Setup_<OS>_<Version>.exe*** file for the specific database (the names and screens displayed for this step are for an Oracle database).
2. From the Setup Wizard, click **Next** to begin installation.
3. From the **Select Destination Location** screen, browse for the directory to which you want to install the Database Protector, or leave the default location (recommended).
4. Specify Database (this step is applicable to Teradata).
5. Click **Next** to continue.
6. From the **Ready to Install** screen, click **Install** to continue.
7. From the **Completing the Protegrity DPS PEP Oracle Setup Wizard** screen, click **Finish** to complete the installation and exit the wizard.
   Directories are created under the installation directory you specified, and the required files are installed to these directories.

2.1.2 Database Protector on UNIX

The following table describes the folders and files that are installed.

Directory	Description
\pep	This directory includes ***.plm*** modules for all installed PEPs. ***.PLM*** files are Protegrity Loadable Modules, which are identical to ***.dll*** files in Windows and ***.so*** files in UNIX.
pep/sqlscripts	This directory contains the subdirectories relevant to each database, which include the required scripts to perform changes to tables and views in the database.
pep/sqlscripts/db2	This directory contains the following scripts: ***createobjects.sql***, ***dropobjects.sql***, ***sample_enc.sql***, and ***sample_tok.sql***.
pep/sqlscripts/oracle	This directory contains the following scripts: ***createobjects.sql***, ***createobjects_compat.sql***, ***dropobjects.sql***, ***sample_enc.sql***, and ***sample_tok.sql***. *compat.sql script is the new UDF signature that is compatible with the UDF signature in Releases 6.0.3 and earlier.

Directory	Description
pep/sqlscripts/teradata	This directory contains the following scripts: • *createdecimalobjects.sql* • *createdecimalobjects_a.sql* • *createobjects.sql* • *createobjects_a.sql* • *createvarcharunicode.sql* • *createvarcharunicode_a.sql* • *dropobjects.sql* • *dropobjects_a.sql* • *dropvarcharunicode.sql* • *dropvarcharunicode_a.sql* • *sample_enc.sql* • *sample_tok.sql* • *testscript.sql* *_a.sql scripts are intended for new UDF signatures to be compatible with the UDF signatues in 4.6 (and earlier releases).
pep/sqlscripts/postgres	This directory contains the following scripts: • *createobjects.sql* • *dropobjects.sql* • *sample_enc.sql* • *sample_tok.sql* • *testscript.sql*
pep/sqlscripts/netezza	This directory contains the following scripts: • *createobjects.sql* • *dropobjects.sql* • *sample_enc.sql* • *sample_tok.sql* • *testscript.sql*

> **Note:** The scripts folders are created based upon the type of database platform installed.

2.1.2.1 Installing Database Protector on UNIX

➤ To install the Database Protector on UNIX:

Before you begin

> **Note:**

> Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.
>
> For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Policy Management Guide 9.1.0.0*.

1. Run the ***PEP<database>_setup_<os>_<version>.sh*** file for the specific database from the terminal or double-click the file and select **Run in Terminal**.
2. Select **YES** to begin installation.
3. Press **ENTER** to install into the destination directory.
4. Specify Database (this step is applicable to Teradata).
   Directories are created under /opt/Protegrity/defiance_dps, and the required files are installed to these directories.

2.1.3 Upgrading the Database Protector

You must uninstall the previous version of the Database Protector before upgrading.

Before you start the upgrade, ensure that the cached audit log files have been moved to the ESA and copy the PEP server configuration file to a temporary directory.

2.1.3.1 Preparing System for Upgrade

Before you begin upgrading the database protector, you must take a back up of the PEP Server.

➤ To prepare system for Upgrade:

1. Stop all database activity, such as protection or unprotection of data, for the Database Protector.
2. Stop the PEP Server from the CLI using this command:

   ```
   [/opt/protegrity/defiance_dps/bin]# .pepsrvctrl stop all
   ```

3. Take a backup of the PEP Server configuration file in a temporary directory. The configuration file is located in *../defiance_dps/data/pepserver.cfg* directory.

2.1.3.2 Dropping all UDF Functions

The UDF Functions for each of the database must be dropped before upgrading to the latest database protector.

➤ To drop all UDF Functions:

1. Login to the database as the *superuser*.
2. Execute the dropobject script which is located in *../defiance_dps/pep/sqlscripts/<Database Name>* directory.
   The script to drop the standard UDF functions is *dropobjects.sql*.

```
labs10td1410-1:/opt/protegrity/defiance_dps/pep/sqlscripts/teradata # ls -l drop*
-rw-r--r-- 1 root root 2081 Oct 21 11:32 dropobjects.sql
-rw-r--r-- 1 root root 1875 Oct 21 11:32 dropobjects_a.sql
-rw-r--r-- 1 root root  781 Oct 21 11:32 dropvarcharunicode.sql
-rw-r--r-- 1 root root  793 Oct 21 11:32 dropvarcharunicode_a.sql
```

Figure 2-2: Drop UDF Functions

> **Note:** Some database platforms include specific UDFs for decimal, Unicode and other data types. If these were also installed, then run the corresponding *dropobjects* script.

3. Backup and rename the `../defiance_dps` directory. Remove the `../defiance_dps` directory only if the upgrade is successful.

2.1.3.3 Upgrading the Database Protector

Upgrading the Database Protector involves installing the latest version of the protector and restarting the PEP server.

To upgrade the Database Protector:

Before you begin

> **Note:**
> Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.
>
> For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Policy Management Guide 9.1.0.0*.

1. Install the new version of the Database Protector.
   For more information about installing a new version of the Database Protector, refer to the *Installation Guide 9.1.0.0*.
2. If the `pepserver.cfg` file has been customized, then add the same information in the new configuration file.

2.1.3.4 Recreating the UDF Functions

After upgrading the database protector, you can recreate the UDF Functions that were dropped.

To re-create the UDF Functions:

1. Login to the database as the *superuser*.
2. Execute the *createobject* script located in `../defiance_dps/pep/sqlscripts/<Database Name>` directory.
   The script to create the standard UDF functions is `createobjects.sql`.

> **Note:** Some database platforms include specific UDFs for decimal, Unicode and other data types. If these were also installed, then run the corresponding *createobjects* script.

2.1.3.5 Starting the PEP Server

The PEP server must be started to ensure that the configurations are applied to the Database Protector.

To start the PEP server:

1. Run the following command to start the PEP server:
   ```
   /opt/protegrity/defiance_dps/bin]# ./pepsrvctrl start
   ```
2. Deploy the policies from the ESA to the Database Protector.

The upgraded Database Protector is now ready to protect, unprotect, or reprotect data.

Chapter 3

MS SQL Server Database Protector

The MS SQL Server Database Protector (MS SQL DB Protector) secures access to sensitive data on MS SQL Server databases. This Protector supports 2008, 2012, 2014, 2016, and 2019 versions of MS SQL Server in both 32-bit and 64-bit configurations.

This section describes the following tasks that you need to perform to setup and use the MS SQL Database Protector:

- MS SQL DB Protector installation
- MS SQL DB Protector configuration
- MS SQL DB Protector UDFs installation
- Functions and extended stored procedures

3.1 Installing and Uninstalling the MS SQL Database Protector

This section describes the procedure to install and uninstall the MS SQL Database Protector.

3.1.1 Verifying the Prerequisites

This section describes the prerequisites including the hardware, software, and network requirements for installing the MS SQL database protector.

The following are the prerequisites for installing the MS SQL Database Protector:

- The ESA 9.2.0.0 appliance is installed, configured, and running.
- The IP address or host name of the ESA is noted.
- The administrator rights for the operating system are granted.
- The DBA rights to the MS SQL database are granted.
- Before installing the protector, ensure that the Policy Information Management (PIM) is initialized, if you are installing the ESA for the first time. This prerequisite holds true for versions 7.2.0 and later releases.

> **Note:** For more information about initializing the PIM, refer to the section *Initializing the Policy Management* in the *Protegrity Policy Management Guide 9.2.0.0*.

3.1.2 Installing the Log Forwarder

The Log Forwarder sends audit logs of protect, unprotect operations to the ESA. This section describes the steps to install the Log Forwarder.

To Install the Log Forwarder:

1. Download the *DatabaseProtector_WIN-ALL-64_x86-64_MSSQL-ALL-64_9.2.0.0.x.zip* installation package made available by Protegrity.
2. To install the MS SQL protector, create a directory on the machine where you want to install the protector.

   > **Note:** If you do not create a directory to install the MS SQL protector, then it is installed in the default `C:\Program Files\Protegrity` directory.

3. To extract the files from the installation package to the created directory, right-click the file.
   A context menu appears.
4. From the context menu, select **Extract All**.
   The following files are extracted from the *DatabaseProtector_WIN-ALL-64_x86-64_MSSQL-ALL-64_9.2.0.0.x.zip* file:
   - *LogforwarderSetup_Windows_x64_9.2.0.0.x.exe*
   - *PepServerSetup_Windows_x64_9.2.0.0.x.exe*
   - *PepSQLServerSetup_Windows_x64_9.2.0.0.x.exe*
   - *U.S.Patent.No.6,321,201.Legend.txt*
5. To install the Log Forwarder, run the *LogforwarderSetup_Windows_x64_9.2.0.0.x.exe* file.
   The **Welcome to the Logforwarder Setup Wizard** appears.

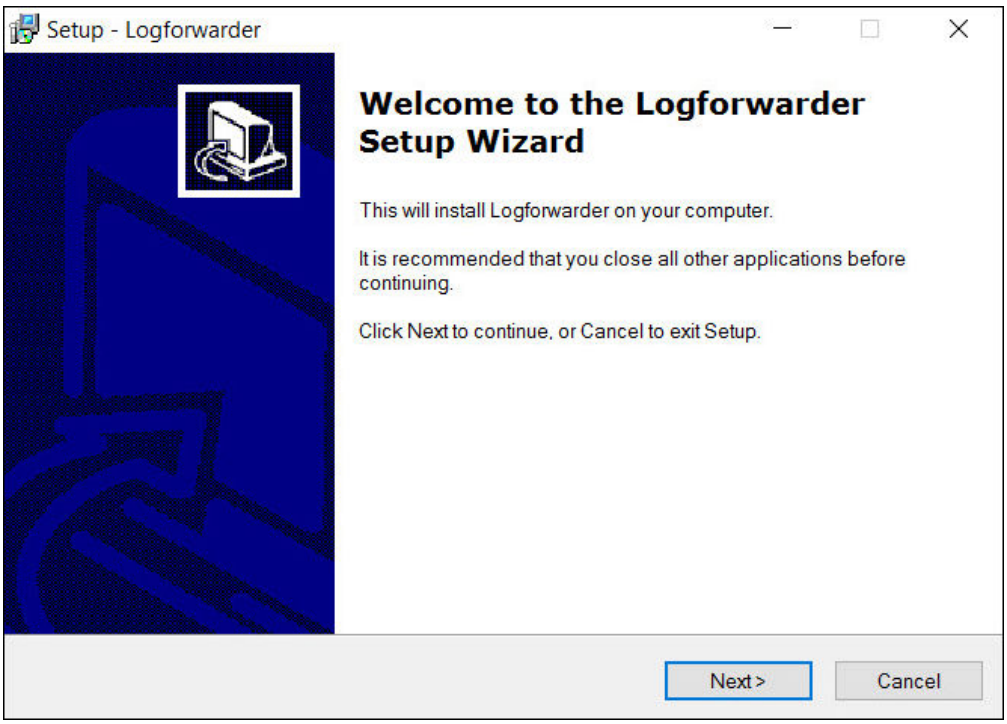

Figure 3-1: Welcome to the Logforwarder Sertup Wizard

Caution: It is mandatory to install the Log Forwarder before installing the PEP server to ensure that the MS SQL Database Protector is configured correctly.

6. Click **Next**.
   The **Audit Store Connectivity Information** screen appears.

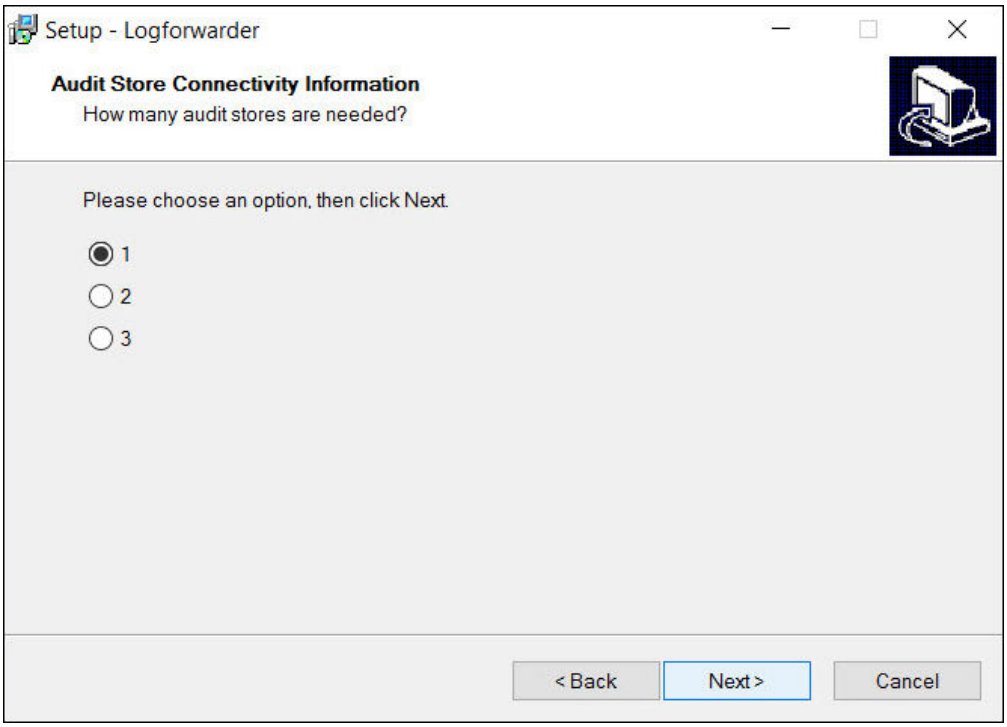

Figure 3-2: Number of Audit Stores

7. On the **Audit Store Connectivity Information** screen, select the number of Audit Stores as required.

> **Note:** You can enter a minimum of 1 to a maximum 3 Audit Store endpoints.

8. Click **Next**.
The **Audit Store Connectivity Information** screen appears.

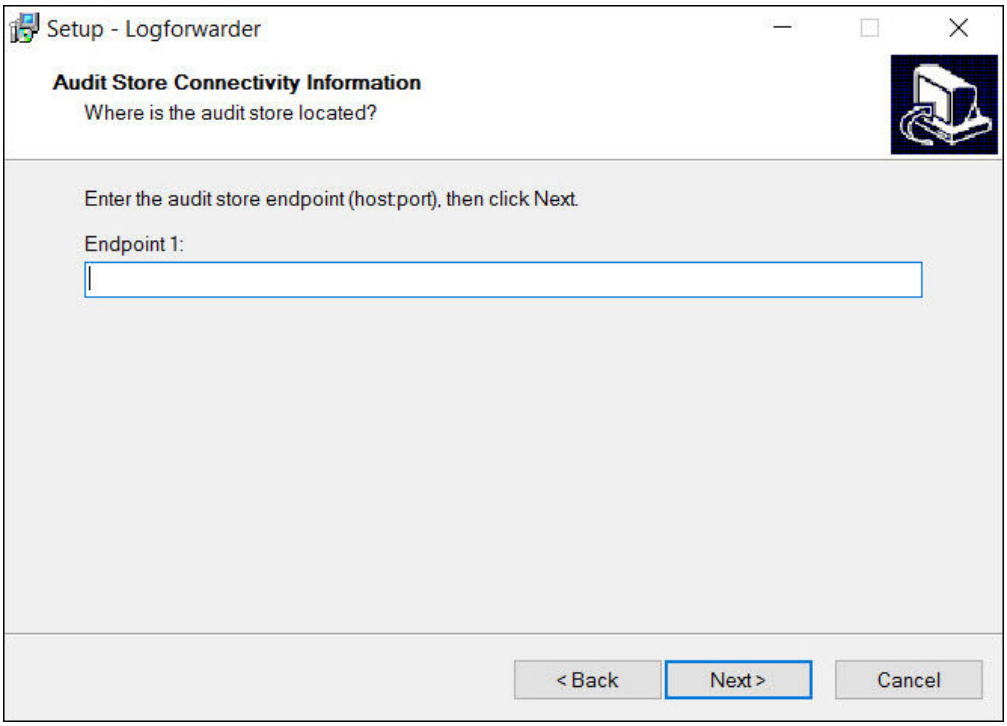

Figure 3-3: Location of Audit Store

9. On the **Audit Store Connectivity Information** screen, in the **Endpoint 1** box, enter the host and port of the ESA for the required Audit Store endpoint.
10. Click **Next**.
The **Select pepserver location** screen appears.

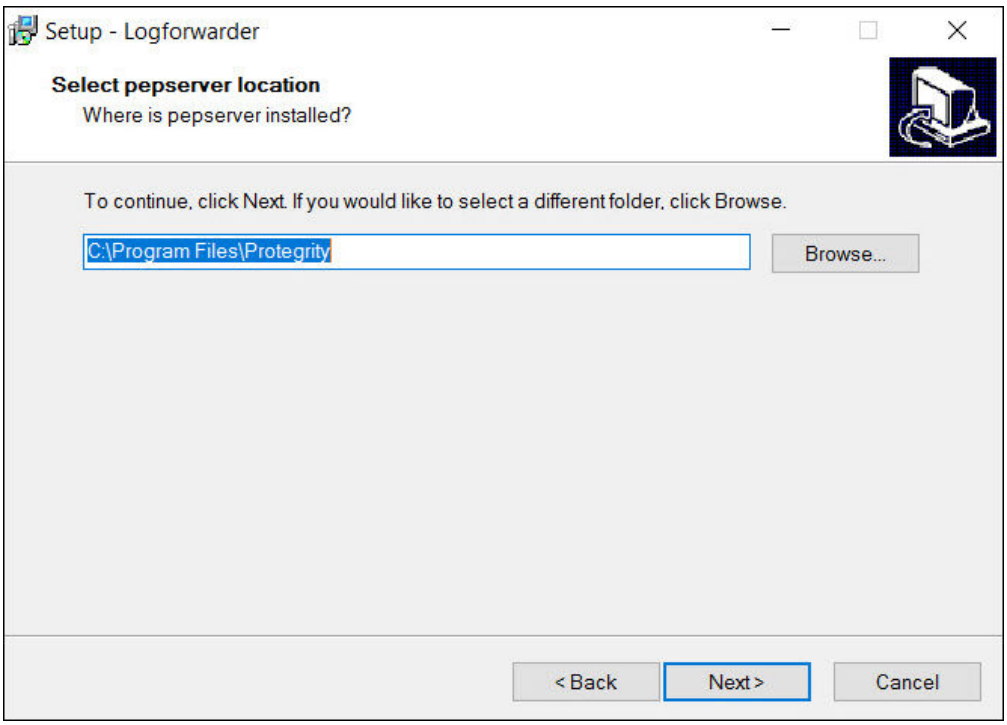

Figure 3-4: Select PEP server location

11. To change the default installation directory for the PEP server, perform the following steps.
    a. Click **Browse**.
    b. Select the required installation directory where you want to install the PEP server.
    c. Click **OK**.

> **Note:** The default installation directory is `C:\Program Files\Protegrity` where the PEP server is installed.

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

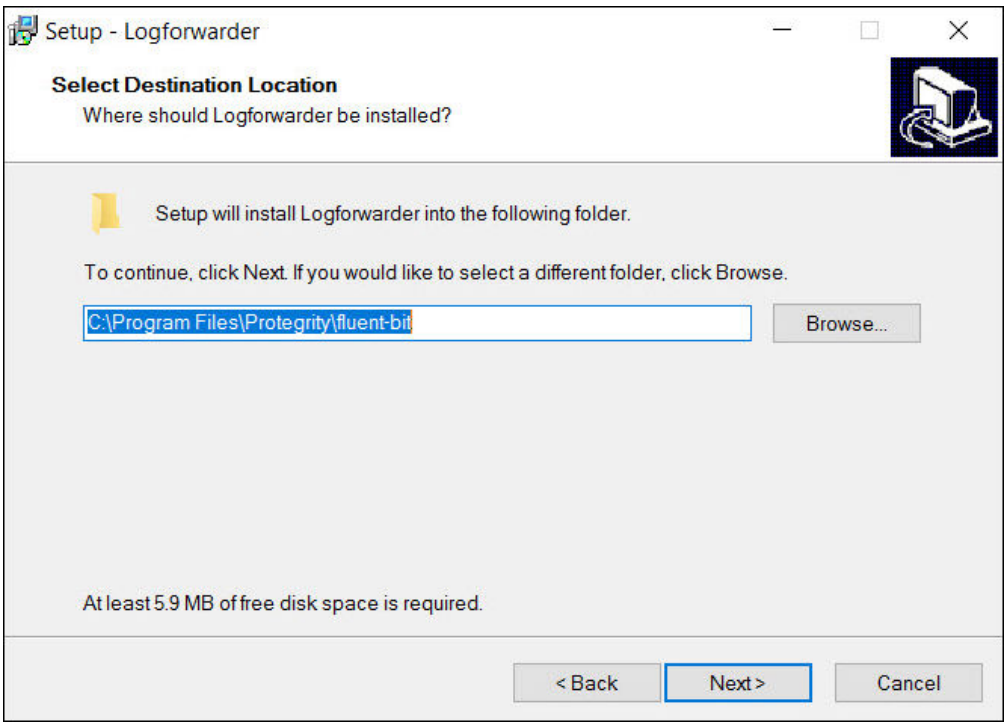

Figure 3-5: Select Destination Location for Logforwarder Setup

13. To change the default installation directory for Log Forwarder, perform the following steps.
    a. Click **Browse**.
    b. Select the required installation directory where you want to install the Log Forwarder.
    c. Click **OK**.

> **Note:**
> - The default installation directory is `C:\Program Files\Protegrity\fluent-bit` where the Log Forwarder is installed.
> - The Log Forwarder component is herein referred to as Fluent Bit.

14. Click **Next**.
    The **Ready to Install** screen appears.

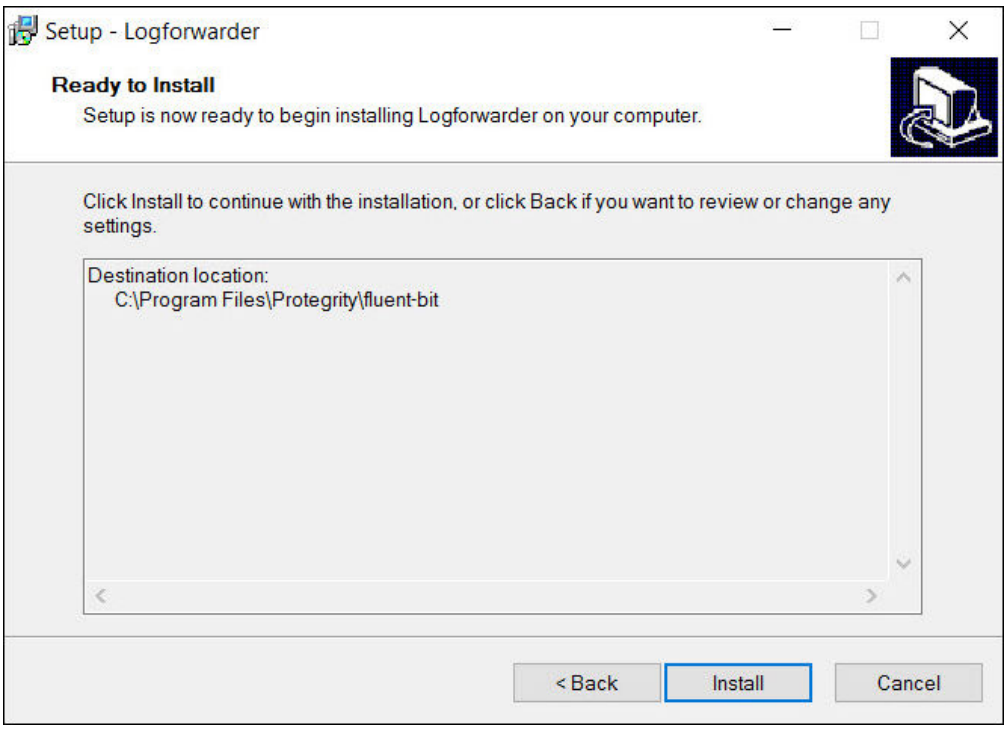

Figure 3-6: Ready to Install screen

15. Click **Install**.
    The installer completes the installation of the Log Forwarder in the specified location and the **Completing the Logforwarder Setup Wizard** screen appears.

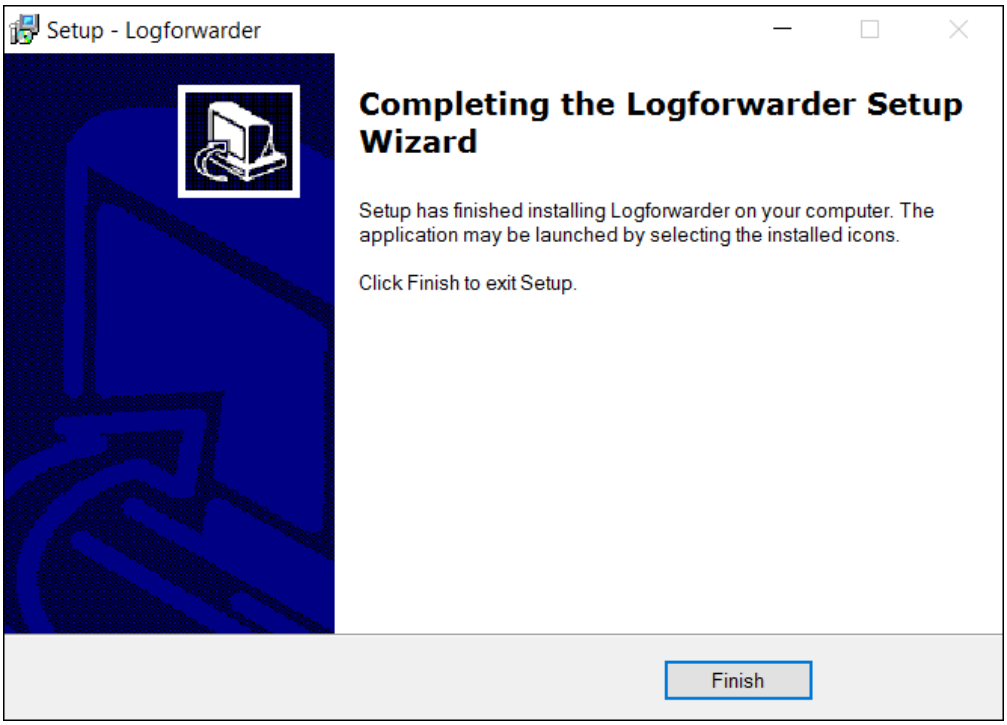

Figure 3-7: Completing the Logforwarder Setup Wizard

16. To exit the installation wizard, click **Finish**.
    The **Completing the Logforwarder Setup Wizard** exits.

> **Note:** After the Log Forwarder is installed successfully, to ensure that the Logforwarder sends logs to the ESA, verify that the Log Forwarder service is in *Running* state.

17. To verify the Logforwarder service state, perform the following step.
    a. Navigate to **Start** > **Control Panel** > **System and Security** > **Administrative Tools** > **Services**.

> **Note:**
> - For more information about configuring the Log Forwarder, refer to section *Appendix A: PEP Server Configuration File*.
> - After installing the MS SQL Database Protector, in the *pepserver.cfg* file, if you set the *Logging configuration* parameter as *mode=drop*, then you must drop the SQL objects and recreate them.

18. Verify that the **Logforwarder** service is in *Running* state.

3.1.3 Installing the PEP Server

The PEP server is the connection between the ESA and the MS SQL Database Protector. The PEP server is responsible for accepting the policy that is deployed from the ESA. It also sends back the audit logs to the ESA. This section describes the steps to install the PEP server.

The following set of certificate files are downloaded in the `<installation_directory>\defiance_dps\data` directory:

- *authesa.plm*
- *CA.pem*
- *cert.key*
- *cert.pem*
- *certkeyup.bin*
- *keyinternal.plm*
- *pepserver.cfg*
- *pepserver.pid*

To Install the PEP server:

1. Navigate to the directory where you have extracted the installation files.
2. To install the PEP server, run the *PepServerSetup_Windows_x64_9.2.0.0.x.exe* file.
   The **Welcome to the Protegrity PEP Server Setup Wizard** appears.

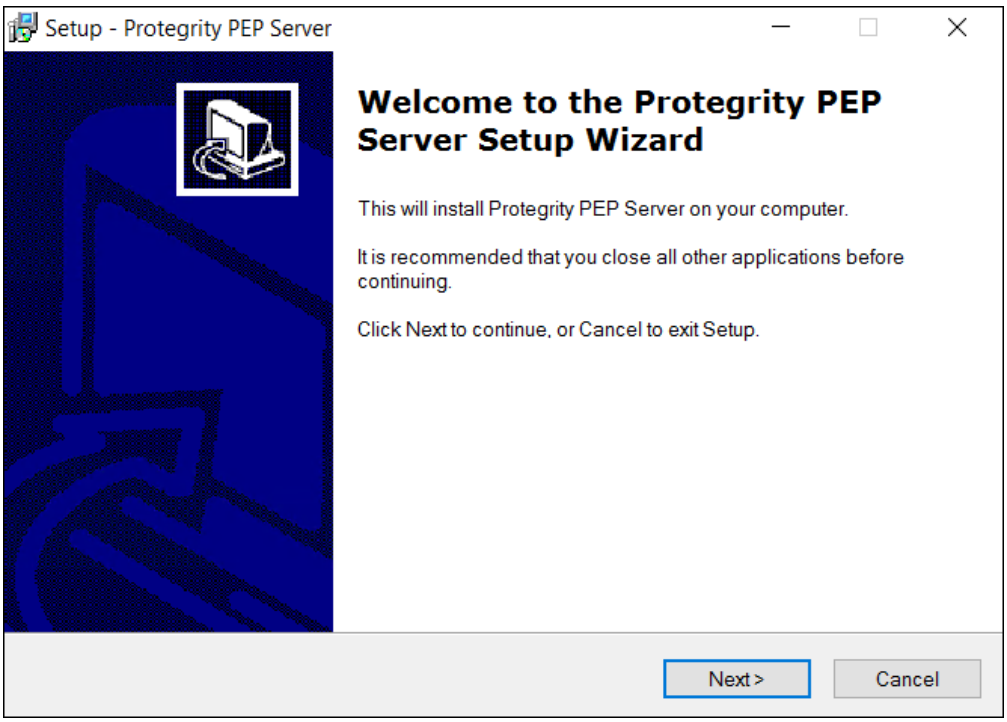

Figure 3-8: Welcome to the Protegrity PEP Server Setup Wizard

3. Click **Next**.
   The **ESA Connectivity Information** screen appears.

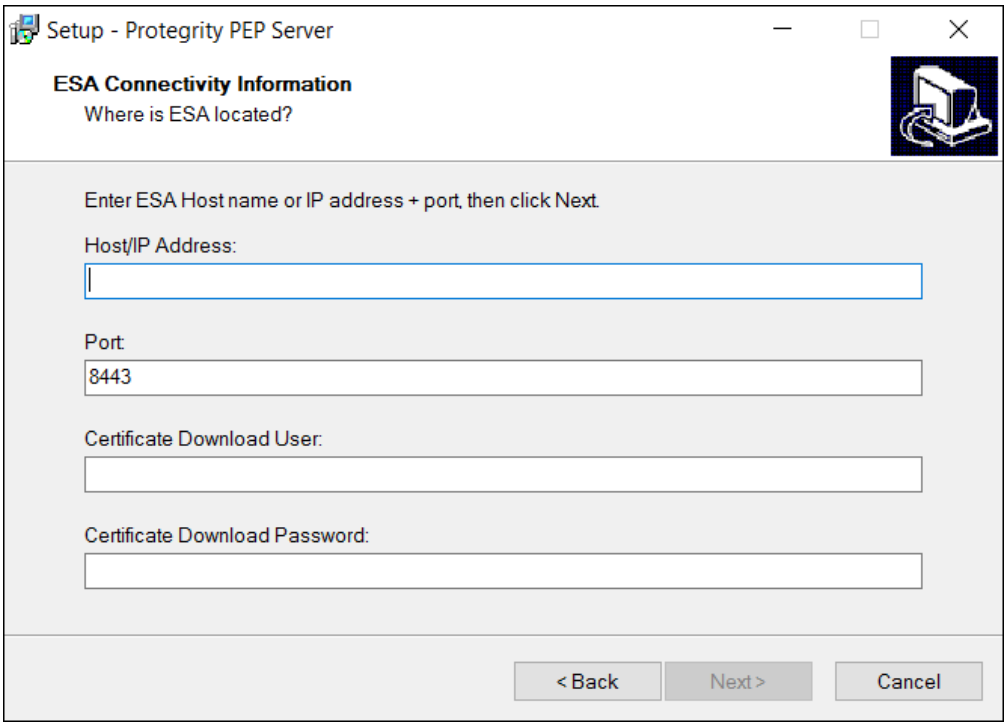

Figure 3-9: ESA Connectivity Information screen

4. In the **Host/IP Address** box, enter the host name or the IP address of the ESA.

> **Caution:** Ensure that the ESA is up and running with the *HubController* service. The *HubController* service must be in *Running* state to enable the downloading of certificates automatically.

5. In the **Port** box, enter the port number of the ESA.

   > **Note:** You can retain the default port number of the ESA, 8443.

6. In the **Certificate Download User** box, enter the username of the ESA.

   > **Note:** It is recommended to use *Admin* user.

7. In the **Certificate Download Password** box, enter the password for the respective username.
8. Click **Next**.
   The **Select Destination Location** screen appears.

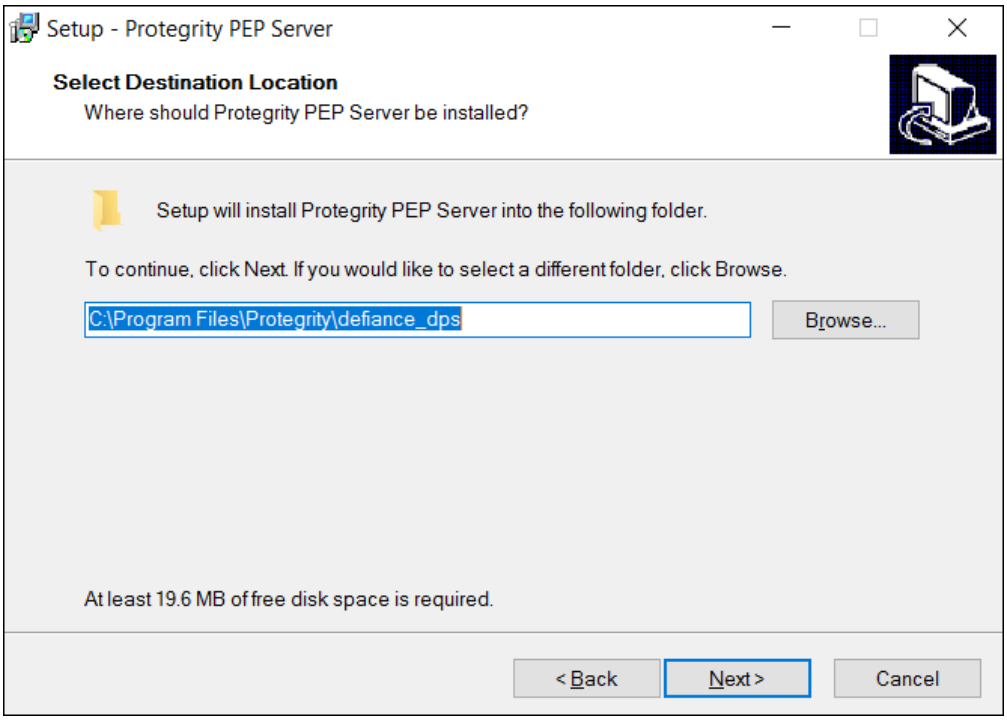

Figure 3-10: Select Destination Location screen

9. To change the default installation directory for the PEP server, perform the following steps.
   a. Click **Browse**.
   b. Select the required installation directory where you want to install the PEP server.
   c. Click **OK**.

   > **Note:** The default directory is `C:\Program Files\Protegrity\defiance_dps` where the PEP server is installed.

10. Click **Next**.
    The **Ready to Install** screen appears.

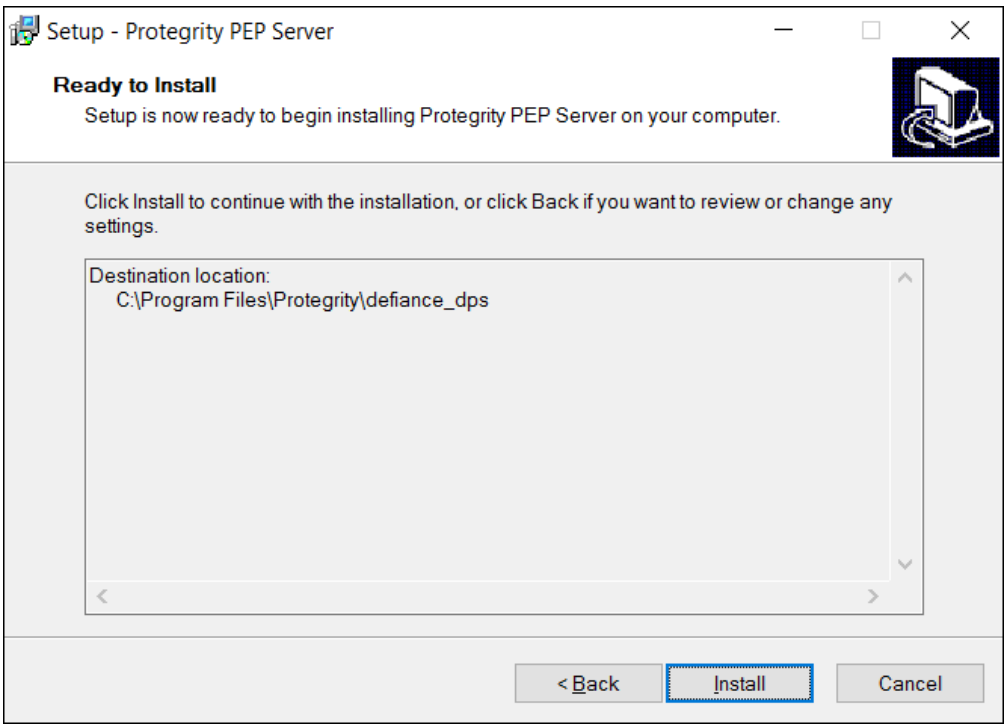

Figure 3-11: Ready to Install screen

11. Click **Install**.
    The installer completes the installation of the PEP server successfully in the specified location and the **Completing the Protegrity PEP Server Setup Wizard** screen appears.

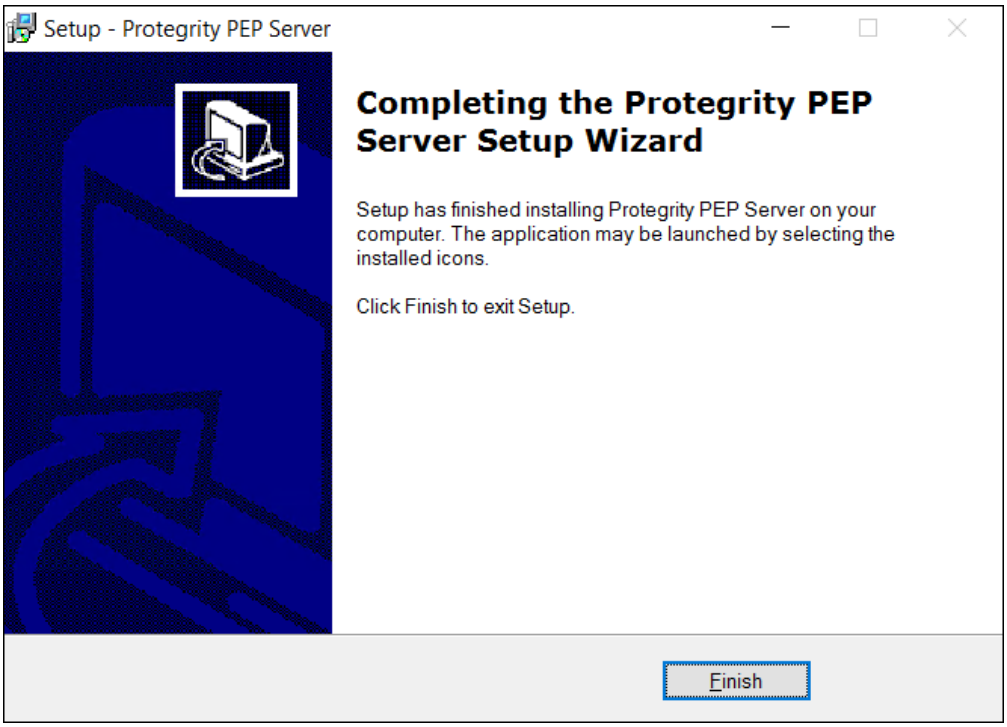

Figure 3-12: Completing the Protegrity PEP Server Setup Wizard

12. To exit the installation wizard, click **Finish**.
    The **Completing the Protegrity PEP Server Setup Wizard** exits.

> **Note:**

> After the PEP server is installed successfully, ensure that the PEP server service is in *Running* state. The PEP server fetches the policy in the protector shared memory. If the PEP server service is not running, then the policy will not be deployed in the ESA.

13. To verify the PEP server service state, perform the following step.
    a. Navigate to **Start** > **Control Panel** > **System and Security** > **Administrative Tools** > **Services**.
14. Verify that the **Protegrity PEP Server** service is in *Running* state.

3.1.3.1 Installing the PEP for MS SQL Server

The PEP for MS SQL Server runs security operations such as protect, unprotect data. This section describes the steps to install the PEP for MS SQL Server.

After the PEP for MS SQL Server are installed successfully, the following scripts are downloaded in the `<install_dir>\Protegrity\Database Protector\pep\sqlscripts\sqlserver` directory:

- *0.0.DropObjects.sql*
- *1.0.CreateAssembly.sql*
- *2.0.CreateFunctions.sql*

To Install the PEP for MS SQL Server:

1. Navigate to the directory where you have extracted the installation files.
2. To install the PEP for MS SQL server, run the *PepSQLServerSetup_Windows_x64_9.2.0.0.x.exe* file.
   The **Welcome to the Protegrity Data Security Platform – SQL Server UDF Setup Wizard** appears.

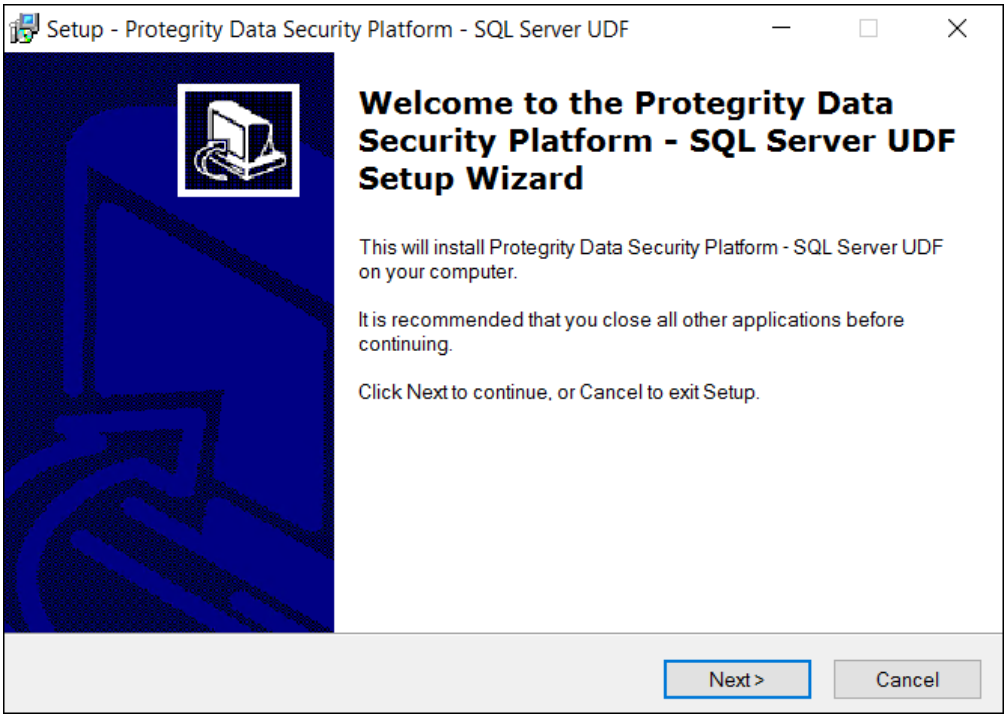

Figure 3-13: Welcome to the Protegrity Data Security Platform - SQL Server UDF Setup Wizard

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

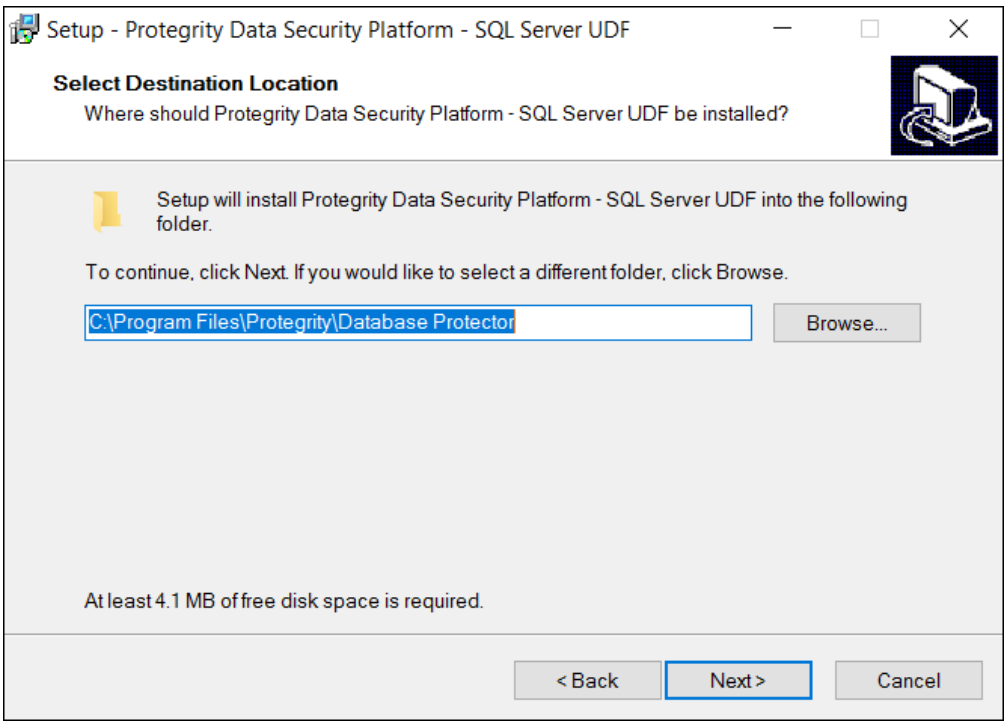

Figure 3-14: Select Destination Location screen

4. To change the default installation directory for the PEP for MS SQL Server, perform the following steps.
   a. Click **Browse**.
   b. Select the required installation directory where you want to install the PEP for MS SQL Server.
   c. Click **OK**.

> **Note:** The default directory is `C:\Program Files\Protegrity\Database Protector` where the PEP for MS SQL Server is installed.

5. Click **Next**.
   The **Ready to Install screen** appears.

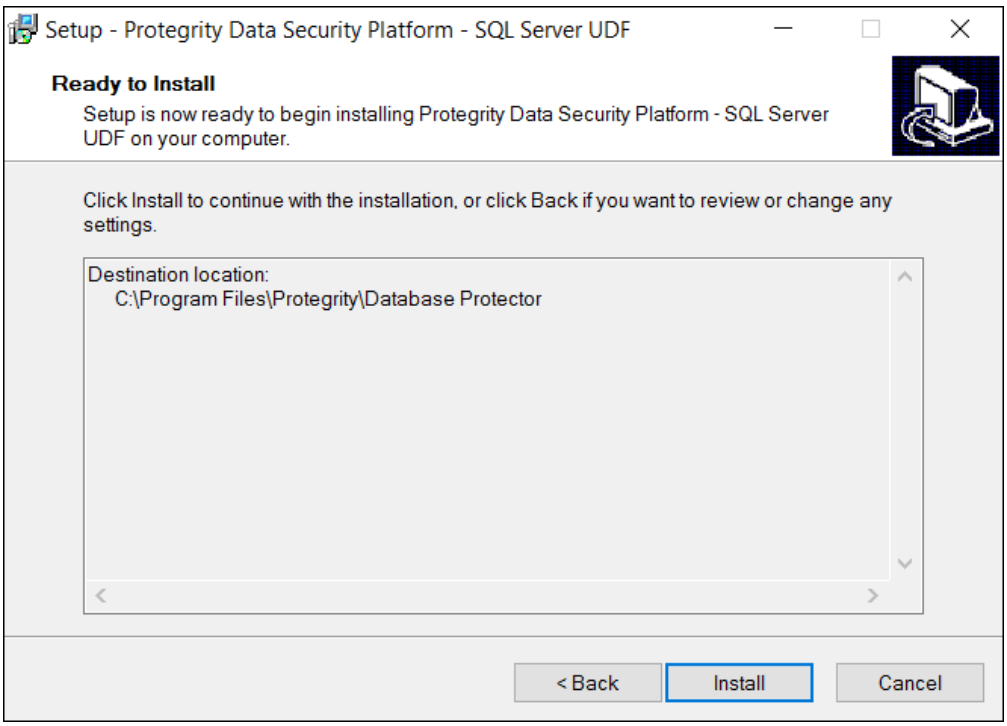

Figure 3-15: Ready to Install screen

6. Click **Install**.
The installer completes the installation of the PEP for MS SQL Server successfully in the specified location and the **Completing the Protegrity Data Security Platform - SQL Server UDF Setup Wizard** screen appears.

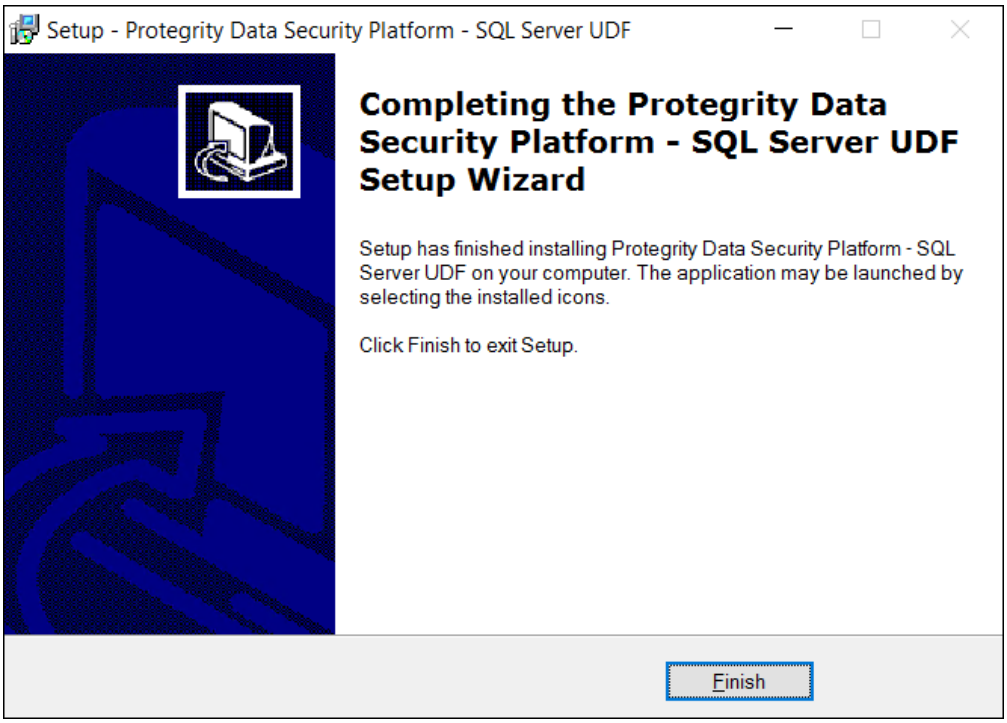

Figure 3-16: Completing the Protegrity Data Security Platform - SQL Server UDF Setup Wizard

7. Click **Finish**.
The **Completing the Protegrity Data Security Platform - SQL Server UDF Setup Wizard** exits.

3.1.3.2 Starting the PEP Server

To start the PEP Server:

1. In Windows, navigate to **Start** > **Control Panel** > **Administrative Tools** > **Services**.
2. Select the **Protegrity PEP Server** service.
3. To start the PEP server, right-click the Protegrity **PEP Server service** and click **Start**.
   The PEP server starts.

3.1.4 Configuring Settings for MS SQL Database Protector

Some configuration settings are required to be setup for the MS SQL Database Protector. These settings are done automatically by the installation process and do not require manual intervention. The following table describes these settings.

Table 3-1: Settings for MS SQL Server

Setting	Description
Communication ID	Set the *Communication ID* to *0* in the • *pepserver.cfg* file, and • **HKEY_LOCAL_MACHINE** > **SOFTWARE** > **Protegrity** > **Defiance DPS** > **SQL CLR** registry entry This becomes the MS SQL Server default setting.
Domain Name	Using the LDAP member-source component, update the registry value in **HKEY_LOCAL_MACHINE** > **SOFTWARE** > **Protegrity** > **Defiance DPS** > **SQL CLR** This helps the Administrator include domain names with every user name. This makes the user name unique.

Note:

- Truncating the user names could lead to a security vulnerability and could result in user names, without the domain names, being treated as duplicate.
- It is recommended not to truncate the domain name as it is insecure. If the SQL Server instance is configured to perform windows authentication, then the mixed mode authentication should be disabled.
- A Windows authenticated user must provide the user name with the domain or host name prepended.

Important: It is necessary to secure the connection between any client application and a SQL Server instance. The *TRUSTWORTHY* property for the MS SQL database is used to indicate whether the SQL Server instance trusts the database and its contents.

Earlier, while running the *1.0.CreateAssembly.sql* script during installation of the MS SQL Database Protector, the *TRUSTWORTHY* property was set to *ON* in the *ALTER DATABASE* statement. Keeping the *TRUSTWORTHY* property set to *ON*, increases security risk. It is recommended to keep the *TRUSTWORTHY* property set to *OFF* to avoid malicious threats when the database is connected to the server. However, if the *TRUSTWORTHY* database property is set to *OFF* while running the *1.0.CreateAssembly.sql* script, then the installation fails with the following error:

```
CREATE ASSEMBLY for assembly 'XCPepConnector' failed because assembly 'XCPepConnector'
is not authorized for PERMISSION_SET = UNSAFE. The assembly is authorized when either
of the following is true: the database owner (DBO) has UNSAFE ASSEMBLY permission and
the database has the TRUSTWORTHY database property on; or the assembly is signed with
```

a certificate or an asymmetric key that has a corresponding login with UNSAFE ASSEMBLY permission.

It is recommended to avoid changing the *TRUSTWORTHY* property setting. An alternative method to mitigate this issue is that a certificate can be created for the MS SQL database using the signed *dll* from Protegrity. From this certificate a certificate-based login can be created for the database. An authorized certificate signed by a trusted source can validate the secured connection between the SQL Server instance and the database. A login is created with the certificate to connect the database securely with the server.

For more information about how to create a certificate-based login for the MS SQL database using the signed *dll* from Protegrity, refer to the section *Creating Certificate-Based Login*.

For more information about configuring the *TRUSTWORTHY* property and creating a certificate, refer to the sections *TRUSTWORTHY Database Property* and *Create a certificate for package signing* respectively, in Microsoft's website.

3.1.5 Installation of User Defined Functions (UDFs)

This section provides information about installing the **User Defined functions (UDFs)** for the MS SQL database protector. You can install the UDFs with or without a certificate-based login. To know more about certificate-based login, refer to the section, *Understanding the Certificate-Based Login*.

3.1.5.1 Creating Certificate-Based Login

This section describes the steps to create create certificate-based login for the MS SQL Server Database using signed *dll* from Protegrity.

To create certificate-based login:

1. Create a certificate for the database using the signed *dll* from Protegrity to authenticate the identity of the server.

   For example:

   ```
   CREATE CERTIFICATE <Certificate_name>
   FROM EXECUTABLE FILE = '<install_dir>\Protegrity\DefianceDPS\pep\XCPepConnector.dll';
   GO
   ```

2. Create a login bound to the certificate and grant the *UNSAFE ASSEMBLY* permissions.

   For example:

   ```
   CREATE LOGIN <Login_name>
   FROM CERTIFICATE <Certificate_name>
   GO

   GRANT UNSAFE ASSEMBLY TO <Login_name>;
   GO
   ```

3.1.5.2 Creating the User Defined Functions (UDFs)

This section describes the steps to create the UDFs for the MS SQL Server database protector.

Before you begin

- If the *TRUSTWORTHY* database property is OFF, then ensure that a certificate-based login is created from the signed *dll* file provided by Protegrity.
- For certificate-based login, perform the steps in the section, *Creating Certificate-Based Login*.

- You must create a certificate-based login before the creating the UDFs.
- If the *TRUSTWORTHY* database property is ON, then perform the steps to create the UDFs without certificate-based login.

To create the UDFs for MS SQL Server database protector:

1. To connect to the database, login as the privileged user with the *CREATE ASSEMBLY* permissions.

   Note: In MS SQL Server, the *sa* login is disabled by default. You must enable it to login with *sa* user for connecting to the database.

2. To run the scripts, navigate to the `<install_dir>\Protegrity\Database Protector\pep\sqlscripts\sqlserver` directory.
3. To execute the registered assembly, run the *CreateAssembly.sql* script.

```
USE [master]
GO

sp_configure 'clr enable',1
RECONFIGURE
GO

alter database [master] set trustworthy on
GO

if exists(SELECT name FROM sys.assemblies WHERE name = 'XCPepConnector')
 DROP ASSEMBLY [XCPepConnector]
GO

CREATE ASSEMBLY [XCPepConnector]
AUTHORIZATION [dbo]
FROM 'C:\Program Files\Protegrity\Database Protector\pep\XCPepConnector.dll'
WITH PERMISSION_SET = UNSAFE
GO
```

4. To create the standard UDFs, run the *CreateFunctions.sql* script.

3.1.6 Uninstalling the MS SQL Database Protector

This section describes the procedures to uninstall the MS SQL Database Protector.

To Uninstall MS SQL Database Protector:

- Uninstall the PEP for MSSQL
- Uninstall the PEP Server
- Uninstall the Log Forwarder
- Delete the installation directories.

3.1.6.1 Uninstalling the PEP server

This section describes the procedure to uninstall the PEP server.

To Uninstall the PEP server:

1. To open the Services Manager, navigate to **Start** > **Control Panel** > **System and Security** > **Administrative Tools** > **Services**.

   > **Note:** Alternatively, you can open the Services Manager, from the **Run** dialog box. From the *Windows* menu, navigate to **Start** > **Run**. In the **Run** window, type *services.msc* and then, click **OK**.

   The **Services** window appears.
2. To select the PEP server from the list of services, select **Protegrity PEP Server**.
3. To stop the PEP server service, navigate to **Action** > **Stop**.
4. Select **Stop**.
   The PEP server service is stopped.
5. To uninstall the PEP server, perform the following steps.
   a. From the Windows menu, navigate to **Start** > **Control Panel** > **Programs** > **Programs and Features**.
   b. From the list of programs, under the *Name* column, select **Protegrity PEP Server**.
   c. Click **Uninstall**.

   > **Note:** Alternatively, navigate to the `..\defiance_dps` directory. To uninstall the PEP server, select *unins000* file and then double-click it.

   The PEP server is uninstalled and the *defiance_dps* directory is deleted.

3.2 Configuring access to perform queries on MS SQL Database from ESA

A user must be granted access and permissions to the certain tables such that they can query the database for members and groups.

The following privilege rights need to be granted to the user defined in the member source configuration on the ESA, who will be performing the queries:

- Select access to *MASTER.SYSLOGINS*
- Select access to *SYSUSERS*

3.3 Configuring the *Delete* Trigger

If users, without the delete privilege, use the *Delete* trigger to remove selected data from the database, the trigger seemingly completes the activity without any error or warning message. If the trigger result is verified, you notice that the selected data is still available in the database.

To add an error or a warning message, the *Delete* trigger should be edited to include the ELSE section after the WHILE loop. The following section contains a sample snippet.

```
IF @RESULT = 0
   WHILE @@FETCH_STATUS = 0 AND @RESULT = 0
     BEGIN
       DELETE [dbo].[test2_ENC]
         WHERE [id_test2]=@id_test2
         FETCH NEXT FROM del_cursor INTO @name,@age,@id_test2
     END
```

```
ELSE
   RAISERROR ('Permission denied', 16, 1) WITH NOWAIT
```

> **Note:** The *Delete* trigger fails, if users, that have the delete privilege but do not have the select or unprotect privilege. Use the *Delete* trigger, where the output is set to give an exception. The same settings work if the output is set to NULL.

3.4 Upgrading the Database Protector

You must uninstall the previous version of the Database Protector before upgrading.

Before you start the upgrade, ensure that the cached audit log files have been moved to the ESA and copy the PEP server configuration file to a temporary directory.

3.4.1 Preparing the System for Upgrade

➤ To prepare the system for Upgrade:

1. Stop all database activity, such as protection or unprotection of data, for the Database Protector.
2. Click **Start** from Windows.
3. Navigate to **Run**, type *services.msc*, and click **OK**.
   The *Services* screen appears.
4. Right-click the *Protegrity PEP Server* service and select **Stop**.
5. Take a backup of the *pepserver.cfg* configuration file in a temporary directory.
   The *pepserver.cfg* configuration file is located in the *<INSTALL_DIRECTORY>\Defiance DPS\data* directory.

3.4.2 Dropping all UDF Functions

➤ To drop all UDF Functions:

1. Login to the database as the superuser.
2. Execute the *dropobject* script, which is located in the *<INSTALL_DIRECTORY>\Database Protector\pep\sqlscripts\sqlserver* directory.
   The script to drop the standard UDF functions is *DropObjects.sql*.
3. Backup and rename the *Defiance DPS* directory. Remove these directories, only if the upgrade is successful.

3.4.3 Upgrading the Database Protector

Upgrading the Database Protector involves installing the latest version of the protector and restarting the PEP server.

➤ To upgrade the Database Protector:

Before you begin

> **Note:**

> Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.
>
> For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Policy Management Guide 9.1.0.0*.

1. Install the new version of the Database Protector.
   For more information about installing a new version of the Database Protector, refer to the *Installation Guide 9.1.0.0*.
2. If the `pepserver.cfg` file has been customized, then add the same information in the new configuration file.

3.4.4 Recreating the UDF Functions

➤ To re-create the UDF Functions:

1. Login to the database as the superuser.
2. Execute the *createassembly* and *createfunction* scripts located in the *<install_directory>\Database Protector\pep\sqlscripts\sqlserver* directory.
   The script to execute the registered assembly is *CreateAssembly.sql*. The script to create the standard UDF functions is *CreateFunctions.sql*.

3.4.5 Restarting the PEP Server

➤ To start the PEP server and restart the MS SQL Server service:

1. Click **Start** from Windows.
2. Navigate to **Run**, type *services.msc*, and click **OK**.
   The *Services* screen appears.
3. Right-click the **Protegrity PEP Server** service and select **Start**.
4. Right-click the **SQL Server (MS SQL Server)** service and select **Restart**.
5. Deploy the policies from the ESA to the Database Protector.

The upgraded Database Protector is now ready to protect, unprotect, or reprotect data.

3.5 MS SQL Server Protector Example

3.5.1 MS SQL Server Protector Example – Encryption

This is a sample for protection and unprotection of data using encryption. Every time a user tries to perform an operation, an Access Check procedure is called to verify that the user has access. Two variables, CCN and RATING, are encrypted and decrypted in this example.

```
-----------------------------------------------------------------------
-- Protegrity User Defined Functions sample script.
--
```

```
-- NOTE: Please change the following 'tags' before executing the script:
--       '<data element>'      : Data element name for encryption or tokenization.
-- Copyright (c) 2013 Protegrity USA, Inc. All rights reserved
--
----------------------------------------------------------------------
  CREATE TABLE SAMPLE (
    ID      INTEGER                 NOT NULL,
    CCN     VARCHAR(32)             NOT NULL,
    FNAME   VARCHAR(32)             NOT NULL,
    LNAME   VARCHAR(32)             NOT NULL,
    RATING  INTEGER                 NOT NULL,
    REFN    INTEGER                 NOT NULL,
    constraint sample_pk primary key(ID)
  ) ;

INSERT INTO SAMPLE values( 1, 'PTY_IVP_FPRTEST_CCN', 'PTY_IVP_FPRTEST_FNAME',
'PTY_IVP_FPRTEST_LNAME', 123456789, 987654321);


BEGIN TRAN T1
-- Checks if user is valid
BEGIN
BEGIN TRAN T2
           DECLARE @check INT,
           @ERROR INT
           SET @ERROR = 0

           exec @check = master.dbo.xp_pty_insert_check '<data element>'
           if @check != 0 or @@error != 0
           BEGIN
               PRINT 'Unauthorized access in dataelement1 - the process will be stopped'
               SET @ERROR = 1
           END

           IF @ERROR = 0
               COMMIT TRAN T2
END

-- End of user check


ALTER TABLE [dbo].[SAMPLE] DROP CONSTRAINT sample_pk


exec sp_rename '[dbo].[SAMPLE]', 'SAMPLE_BAK','OBJECT'

CREATE TABLE [dbo].[SAMPLE_PTY]([ID] [INT] NOT NULL,
[CCN] VARBINARY(48) NOT NULL,
[FNAME] [VARCHAR](32)  NOT NULL,
[LNAME] [VARCHAR](32)  NOT NULL,
[RATING] VARBINARY(16) NOT NULL,
[REFN] [INT] NOT NULL) ON [PRIMARY]

GO


CREATE VIEW [dbo].[SAMPLE]([ID], [CCN], [FNAME], [LNAME], [RATING], [REFN]) AS SELECT
           [ID],
           CONVERT(VARCHAR(32),master.dbo.pty_select([CCN],'<data element>',' ',0)),
           [FNAME],
           [LNAME],
           CONVERT(INT,master.dbo.pty_select([RATING],'<data element>','0',0)),
           [REFN]
FROM [dbo].[SAMPLE_PTY]
GO


-- Create INSERT trigger

CREATE TRIGGER [dbo].[SAMPLE_INS] ON [dbo].[SAMPLE]
INSTEAD OF INSERT AS
BEGIN
SET NOCOUNT ON
```

```
DECLARE
@ID INT,
@CCN VARCHAR(32),
@CCN_PTYRYPTED VARBINARY(80),
@FNAME VARCHAR(32),
@LNAME VARCHAR(32),
@RATING VARCHAR(11),
@RATING_PTYRYPTED VARBINARY(59),
@REFN INT,
@RESULT INT,
@ERROR INT
SET @ERROR = 0
SET @RESULT = 0

DECLARE ins_cursor CURSOR FAST_FORWARD FOR SELECT
           [ID],
           CONVERT(VARCHAR(32),[CCN]),
           [FNAME],
           [LNAME],
           CONVERT(VARCHAR(11),[RATING]),
           [REFN]
FROM inserted

OPEN ins_cursor
FETCH NEXT FROM ins_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN

WHILE @@FETCH_STATUS = 0 AND @ERROR = 0
BEGIN

           EXEC @RESULT = master.dbo.xp_pty_insert @CCN_PTYRYPTED OUTPUT, @CCN, '<data
element>',0
           IF (@RESULT != 0)
             SET @ERROR = 1
           EXEC @RESULT = master.dbo.xp_pty_insert @RATING_PTYRYPTED OUTPUT, @RATING, '<data
element>',0
           IF (@RESULT != 0)
             SET @ERROR = 1


           IF (@ERROR = 0)
           INSERT INTO [dbo].[SAMPLE_PTY]([ID],[CCN],[FNAME],[LNAME],[RATING],[REFN])
           VALUES(@ID,
           @CCN_PTYRYPTED,
           @FNAME,
           @LNAME,
           @RATING_PTYRYPTED,
           @REFN)
           ELSE
           BEGIN
           RAISERROR ('Error occured during encryption process', 16, 1) WITH NOWAIT
           SET @ERROR = 1
           END
           FETCH NEXT FROM ins_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN
END
CLOSE ins_cursor
DEALLOCATE ins_cursor
SET NOCOUNT OFF
END
GO


-- Create UPDATE trigger

CREATE TRIGGER [dbo].[SAMPLE_UPD] ON [dbo].[SAMPLE]
INSTEAD OF UPDATE AS
BEGIN
  SET NOCOUNT ON
  DECLARE
    @ID INT,
    @IDOld INT,
    @CCN VARCHAR(32),
    @CCN_PTYRYPTED VARBINARY(80),
    @CCN_UPD CHAR(1),
```

```
    @FNAME VARCHAR(32),
    @LNAME VARCHAR(32),
    @RATING VARCHAR(11),
    @RATING_PTYRYPTED VARBINARY(59),
    @RATING_UPD CHAR(1),
    @REFN INT,
    @RESULT INT,
    @ERROR INT
    SET @ERROR = 0
    SET @RESULT = 0

  BEGIN
    DECLARE upd_cursor CURSOR FAST_FORWARD FOR SELECT [ID],
      CONVERT(VARCHAR(32),[CCN]),
      [FNAME],
      [LNAME],
      CONVERT(VARCHAR(11),[RATING]),
      [REFN]
    FROM inserted

    DECLARE upd_cursor2 CURSOR FAST_FORWARD FOR SELECT [ID] FROM deleted

    OPEN upd_cursor
    FETCH NEXT FROM upd_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN

    OPEN upd_cursor2
    FETCH NEXT FROM upd_cursor2 INTO @IDOld

    IF UPDATE([CCN]) SET @CCN_UPD = 'T'
    IF UPDATE([RATING]) SET @RATING_UPD = 'T'

    WHILE @@FETCH_STATUS = 0 AND @ERROR = 0
    BEGIN
        EXEC @RESULT = master.dbo.xp_pty_update @CCN_PTYRYPTED OUTPUT,
        @CCN, '<data element>',@CCN_UPD,0
        IF (@RESULT != 0)
           SET @ERROR = 1
        EXEC @RESULT = master.dbo.xp_pty_update @RATING_PTYRYPTED OUTPUT,
        @RATING, '<data element>',@RATING_UPD,0
           IF (@RESULT != 0)
             SET @ERROR = 1
           IF (@ERROR = 0)
           BEGIN
             IF UPDATE([ID])
                BEGIN
                  UPDATE [dbo].[SAMPLE_PTY] SET [ID]=@ID WHERE
                  [ID]=@IDOld
                END
             IF UPDATE([CCN])
                BEGIN
                  UPDATE [dbo].[SAMPLE_PTY] SET [CCN]=@CCN_PTYRYPTED
                  WHERE [ID]=@IDOld
                END
             IF UPDATE([FNAME])
                BEGIN
                  UPDATE [dbo].[SAMPLE_PTY] SET [FNAME]=@FNAME WHERE
                  [ID]=@IDOld
                END
             IF UPDATE([LNAME])
                BEGIN
                  UPDATE [dbo].[SAMPLE_PTY] SET [LNAME]=@LNAME WHERE
                  [ID]=@IDOld
                END
             IF UPDATE([RATING])
                BEGIN
                  UPDATE [dbo].[SAMPLE_PTY] SET
                  [RATING]=@RATING_PTYRYPTED WHERE [ID]=@IDOld
                END
             IF UPDATE([REFN])
                BEGIN
                  UPDATE [dbo].[SAMPLE_PTY] SET [REFN]=@REFN WHERE
                  [ID]=@IDOld
           END
```

```
    END
    ELSE
      BEGIN
        RAISERROR ('Error occured during encryption process', 16, 1) WITH
        NOWAIT
        SET @ERROR = 1
      END

      FETCH NEXT FROM upd_cursor INTO
       @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN
      FETCH NEXT FROM upd_cursor2 INTO @IDOld
    END
    CLOSE upd_cursor
    DEALLOCATE upd_cursor
    CLOSE upd_cursor2
    DEALLOCATE upd_cursor2
  END
  SET NOCOUNT OFF
END
GO

-- Create DELETE trigger

CREATE TRIGGER [dbo].[SAMPLE_DEL] ON [dbo].[SAMPLE]
  INSTEAD OF DELETE AS
  BEGIN
    SET NOCOUNT ON
    DECLARE
      @ID INT,
      @CCN VARCHAR(32),
      @CCN_PTYRYPTED VARBINARY(80),
      @FNAME VARCHAR(32),
      @LNAME VARCHAR(32),
      @RATING VARCHAR(11),
      @RATING_PTYRYPTED VARBINARY(59),
      @REFN INT,
      @RESULT INT
    SET @RESULT = 0

    DECLARE del_cursor CURSOR FAST_FORWARD FOR SELECT [ID],
       CONVERT(VARCHAR(32),[CCN]), [FNAME], [LNAME],
       CONVERT(VARCHAR(11),[RATING]), [REFN]
    FROM deleted

    OPEN del_cursor
    FETCH NEXT FROM del_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN
    EXEC @RESULT = master.dbo.xp_pty_delete_check '<data element>',0
    IF @RESULT = 0
      WHILE @@FETCH_STATUS = 0 AND @RESULT = 0
      BEGIN
         DELETE [dbo].[SAMPLE_PTY]
         WHERE [ID]=@ID
         FETCH NEXT FROM del_cursor INTO
            @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN
      END
    ELSE
      RAISERROR ('Permission denied', 16, 1) WITH NOWAIT
    CLOSE del_cursor
    DEALLOCATE del_cursor
    SET NOCOUNT OFF
  END
GO


IF @@TRANCOUNT = 2
  GOTO error
INSERT INTO [dbo].[SAMPLE] ([ID],[CCN],[FNAME],[LNAME],[RATING],[REFN]) SELECT [ID],[CCN],
[FNAME],[LNAME],[RATING],[REFN] FROM [dbo].[SAMPLE_BAK]
error:
GO

-- ----------------------------------------------------------------------
-- ---------------------------- End of script for [dbo].[SAMPLE]----------
```

```
-- ------------------------------------------------------------------------

ALTER TABLE [dbo].[SAMPLE_PTY] ADD CONSTRAINT sample_pk PRIMARY KEY CLUSTERED ([ID] ASC) ON
[PRIMARY]

GO

DROP TABLE [dbo].[SAMPLE_BAK]

IF @@TRANCOUNT = 1
  BEGIN
    PRINT 'User access check OK'
    COMMIT TRAN T1
  END
ELSE IF @@TRANCOUNT = 2
  BEGIN
    PRINT 'User access check FAILED. Rollback...'
    COMMIT TRAN T2
    ROLLBACK TRAN T1
  END
ELSE IF @@TRANCOUNT = 0
  BEGIN
    PRINT 'Script execution FAILED...'
  END
```

3.5.2 MS SQL Server Protector Example – Tokenization

The following is an example for protecting and unprotecting data using tokenization. Every time a user tries to perform an operation, an Access Check procedure is called to verify that the user has access. Two variables, CCN and RATING, are tokenized, detokenized, and reprotected in this example.

```
------------------------------------------------------------------------
-- Protegrity User Defined Functions sample script.
--
-- NOTE: Please change the following 'tags' before executing the script:
--       '<dataelement1>'      : Data element name for varchar tokenization.
--       '<dataelement2>'      : Data element name for integer tokenization.
-- Copyright (c) 2013 Protegrity USA, Inc. All rights reserved
--
------------------------------------------------------------------------
  CREATE TABLE SAMPLE (
    ID      INTEGER                 NOT NULL,
    CCN     VARCHAR(32)             NOT NULL,
    FNAME   VARCHAR(32)             NOT NULL,
    LNAME   VARCHAR(32)             NOT NULL,
    RATING  INTEGER                 NOT NULL,
    REFN    INTEGER                 NOT NULL,
    constraint pk_sample primary key(ID)
  ) ;

INSERT INTO SAMPLE values( 1, 'PTY_IVP_FPRTEST_CCN', 'PTY_IVP_FPRTEST_FNAME',
'PTY_IVP_FPRTEST_LNAME', 12345, 987654);

BEGIN TRAN T1
-- Checks if user is valid
BEGIN
BEGIN TRAN T2
           DECLARE @check INT,
           @ERROR INT
           SET @ERROR = 0

           exec @check = master.dbo.xp_pty_insert_check '<dataelement1>'
           if @check != 0 or @@error != 0
           BEGIN
               PRINT 'Unauthorized access in <dataelement1> - the process will be stopped'
               SET @ERROR = 1
           END
           exec @check = master.dbo.xp_pty_insert_check '<dataelement2>'
           if @check != 0 or @@error != 0
           BEGIN
               PRINT 'Unauthorized access in <dataelement2> - the process will be stopped'
```

```
                SET @ERROR = 1
            END

            IF @ERROR = 0
                COMMIT TRAN T2
END

-- End of user check


exec sp_rename '[dbo].[SAMPLE]', 'SAMPLE_BAK','OBJECT'

CREATE TABLE [dbo].[SAMPLE_PTY]([ID] [INT] NOT NULL,
[CCN] VARCHAR(48) NOT NULL,
[FNAME] [VARCHAR](32) NOT NULL,
[LNAME] [VARCHAR](32) NOT NULL,
[RATING] INT NOT NULL,
[REFN] [INT] NOT NULL) ON [PRIMARY]

GO

CREATE VIEW [dbo].[SAMPLE]([ID], [CCN], [FNAME], [LNAME], [RATING], [REFN]) AS SELECT
            [ID],
            CONVERT(VARCHAR(32),master.dbo.pty_select2([CCN],'<dataelement1>',' ',0)),
            [FNAME],
            [LNAME],
            master.dbo.pty_selectint([RATING],'<dataelement2>','0',0),
            [REFN]
FROM [dbo].[SAMPLE_PTY]
GO



-- Create INSERT trigger

CREATE TRIGGER [dbo].[SAMPLE_INS] ON [dbo].[SAMPLE]
INSTEAD OF INSERT AS
BEGIN
SET NOCOUNT ON
DECLARE
@ID INT,
@CCN VARCHAR(32),
@CCN_ENCRYPTED VARCHAR(80),
@FNAME VARCHAR(32),
@LNAME VARCHAR(32),
@RATING INT,
@RATING_ENCRYPTED INT,
@REFN INT,
@RESULT INT,
@ERROR INT
SET @ERROR = 0
SET @RESULT = 0

DECLARE ins_cursor CURSOR FAST_FORWARD FOR SELECT
            [ID],
            [CCN],
            [FNAME],
            [LNAME],
            [RATING],
            [REFN]
FROM inserted

OPEN ins_cursor
FETCH NEXT FROM ins_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN

WHILE @@FETCH_STATUS = 0 AND @ERROR = 0
BEGIN

            EXEC @RESULT = master.dbo.xp_pty_tpe_insert @CCN_ENCRYPTED OUTPUT, @CCN,
'<dataelement1>',0
            IF (@RESULT != 0)
              SET @ERROR = 1
            EXEC @RESULT = master.dbo.xp_pty_tpe_int_insert @RATING_ENCRYPTED OUTPUT, @RATING,
```

```
'<dataelement2>',0
           IF (@RESULT != 0)
             SET @ERROR = 1


           IF (@ERROR = 0)
           INSERT INTO [dbo].[SAMPLE_PTY]([ID],[CCN],[FNAME],[LNAME],[RATING],[REFN])
           VALUES(@ID,
           @CCN_ENCRYPTED,
           @FNAME,
           @LNAME,
           @RATING_ENCRYPTED,
           @REFN)
           ELSE
           BEGIN
           RAISERROR ('Error occured during encryption process', 16, 1) WITH NOWAIT
           SET @ERROR = 1
           END
           FETCH NEXT FROM ins_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN
END
CLOSE ins_cursor
DEALLOCATE ins_cursor
SET NOCOUNT OFF
END
GO



-- Create UPDATE trigger

CREATE TRIGGER [dbo].[SAMPLE_UPD] ON [dbo].[SAMPLE]
INSTEAD OF UPDATE AS
BEGIN
SET NOCOUNT ON
DECLARE
@ID INT,
@IDOld INT,
@CCN VARCHAR(32),
@CCN_ENCRYPTED VARCHAR(80),
@CCN_UPD CHAR(1),
@FNAME VARCHAR(32),
@LNAME VARCHAR(32),
@RATING VARCHAR(11),
@RATING_ENCRYPTED INT,
@RATING_UPD CHAR(1),
@REFN INT,
@RESULT INT,
@ERROR INT
SET @ERROR = 0
SET @RESULT = 0

BEGIN
DECLARE upd_cursor CURSOR FAST_FORWARD FOR SELECT
           [ID],
           [CCN],
           [FNAME],
           [LNAME],
           [RATING],
           [REFN]
FROM inserted

DECLARE upd_cursor2 CURSOR FAST_FORWARD FOR SELECT
           [ID]
FROM deleted

OPEN upd_cursor
FETCH NEXT FROM upd_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN

OPEN upd_cursor2
FETCH NEXT FROM upd_cursor2 INTO @IDOld

IF UPDATE([CCN]) SET @CCN_UPD = 'T'
IF UPDATE([RATING]) SET @RATING_UPD = 'T'
```

```
WHILE @@FETCH_STATUS = 0 AND @ERROR = 0
BEGIN
          EXEC @RESULT = master.dbo.xp_pty_tpe_update @CCN_ENCRYPTED OUTPUT, @CCN,
'<dataelement1>',@CCN_UPD,0
          IF (@RESULT != 0)
            SET @ERROR = 1
          EXEC @RESULT = master.dbo.xp_pty_tpe_int_update @RATING_ENCRYPTED OUTPUT, @RATING,
'<dataelement2>',@RATING_UPD,0
          IF (@RESULT != 0)
            SET @ERROR = 1


          IF (@ERROR = 0)
          BEGIN
                    IF UPDATE([ID])
                    BEGIN
                    UPDATE [dbo].[SAMPLE_PTY] SET [ID]=@ID WHERE [ID]=@IDOld
                    END
                    IF UPDATE([CCN])
                    BEGIN
                    UPDATE [dbo].[SAMPLE_PTY] SET [CCN]=@CCN_ENCRYPTED WHERE [ID]=@IDOld
                    END
                    IF UPDATE([FNAME])
                    BEGIN
                    UPDATE [dbo].[SAMPLE_PTY] SET [FNAME]=@FNAME WHERE [ID]=@IDOld
                    END
                    IF UPDATE([LNAME])
                    BEGIN
                    UPDATE [dbo].[SAMPLE_PTY] SET [LNAME]=@LNAME WHERE [ID]=@IDOld
                    END
                    IF UPDATE([RATING])
                    BEGIN
                    UPDATE [dbo].[SAMPLE_PTY] SET [RATING]=@RATING_ENCRYPTED WHERE
[ID]=@IDOld
                    END
                    IF UPDATE([REFN])
                    BEGIN
                    UPDATE [dbo].[SAMPLE_PTY] SET [REFN]=@REFN WHERE [ID]=@IDOld
                    END
          END
          ELSE
          BEGIN
          RAISERROR ('Error occured during encryption process', 16, 1) WITH NOWAIT
          SET @ERROR = 1
          END

          FETCH NEXT FROM upd_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN
          FETCH NEXT FROM upd_cursor2 INTO @IDOld
END
CLOSE upd_cursor
DEALLOCATE upd_cursor
CLOSE upd_cursor2
DEALLOCATE upd_cursor2
END
SET NOCOUNT OFF
END
GO



-- Create DELETE trigger

CREATE TRIGGER [dbo].[SAMPLE_DEL] ON [dbo].[SAMPLE]
INSTEAD OF DELETE AS
BEGIN
SET NOCOUNT ON
DECLARE
@ID INT,
@CCN VARCHAR(32),
@CCN_ENCRYPTED VARCHAR(80),
@FNAME VARCHAR(32),
@LNAME VARCHAR(32),
```

```
@RATING VARCHAR(11),
@RATING_ENCRYPTED INT,
@REFN INT,
@RESULT INT
SET @RESULT = 0

DECLARE del_cursor CURSOR FAST_FORWARD FOR SELECT
           [ID],
           [CCN],
           [FNAME],
           [LNAME],
           [RATING],
           [REFN]
FROM deleted

OPEN del_cursor
FETCH NEXT FROM del_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN

EXEC @RESULT = master.dbo.xp_pty_delete_check '<dataelement1>',0
IF @RESULT = 0
EXEC @RESULT = master.dbo.xp_pty_delete_check '<dataelement2>',0
IF @RESULT = 0

WHILE @@FETCH_STATUS = 0 AND @RESULT = 0
BEGIN
           DELETE [dbo].[SAMPLE_PTY]
           WHERE [ID]=@ID

           FETCH NEXT FROM del_cursor INTO @ID,@CCN,@FNAME,@LNAME,@RATING,@REFN
END
ELSE
RAISERROR ('Permission denied', 16, 1) WITH NOWAIT
CLOSE del_cursor
DEALLOCATE del_cursor
SET NOCOUNT OFF
END
GO


IF @@TRANCOUNT = 2
  GOTO error
INSERT INTO [dbo].[SAMPLE] ([ID],[CCN],[FNAME],[LNAME],[RATING],[REFN]) SELECT [ID],[CCN],
[FNAME],[LNAME],[RATING],[REFN] FROM [dbo].[SAMPLE_BAK]
error:
GO


-- ------------------------------------------------------------------------
-- ------------------- End of script for [dbo].[SAMPLE]-------------------
-- ------------------------------------------------------------------------



ALTER TABLE [dbo].[SAMPLE_PTY] ADD CONSTRAINT sample_pk PRIMARY KEY CLUSTERED ([ID] ASC) ON
[PRIMARY]



GO

DROP TABLE [dbo].[SAMPLE_BAK]

IF @@TRANCOUNT = 1
BEGIN
PRINT 'User access check OK'
COMMIT TRAN T1
END
ELSE IF @@TRANCOUNT = 2
BEGIN
PRINT 'User access check FAILED. Rollback...'
COMMIT TRAN T2
ROLLBACK TRAN T1
END
```

```
ELSE IF @@TRANCOUNT = 0
BEGIN
PRINT 'Script execution FAILED...'
END
```

3.6 Impersonating a user in SQL Server

In this release, if you are using Microsoft SQL Server 2008 or a later version, you can connect to a server using single login with privileges to impersonate another user using the *EXEC AS USER* statement.

The following list provides information about user roles involved during execution of the *EXEC AS* statement:

- **Proxy User**: User who wants to access privileges of the impersonated user. This user is not to be confused with the *ProxyUser* that exists in the ESA.
- **Privileged User**: User who is impersonated by the proxy user.

When a middle-tier application is used with SQL Server, this functionality lets you connect to data from a service account with a pooled connection, and then execute the SQL statements on behalf of the privileged user.

Consider that you are a proxy user without privileges to perform protect/unprotect operations, but intend to protect/unprotect data using a data element. In such cases, the *EXEC AS* statement enables the proxy user to perform the protect/unprotect operation on behalf of the privileged user who possesses this capability.

For more information about the *EXEC AS USER* statement in SQL, refer to *https://msdn.microsoft.com/en-us/library/ms181362.aspx*.

3.6.1 Steps to Impersonate a User

➤ To impersonate a user and protect using a data element:

1. Login to the server as the proxy user, USER1.
2. Execute the following command to change the user execution to privileged user, USER2.

   ```
   EXECUTE AS USER = 'USER2'
   ```
3. Perform the protect/unprotect functionality on the data using a data element.
4. Disconnect the current session and initiate the session as USER1.

Similarly, you can use the *EXEC AS* statement to impersonate any privileged user to perform functions as per the user's capabilities.

> **Note:** The user impersonation settings are effective only for that session.

> **Note:** If a proxy user impersonates a privileged user and performs any operation, then entries in the Audit logs are displayed as performed by the privileged user, and not the proxy user.

3.7 MS SQL DB Protector UDFs

The APIs and UDFs provided with the MS SQL Protector can be used to accommodate custom requirements.

For more information about the UDFs and their related parameters, refer to the section *MS SQL DB Protector Functions* in the *APIs, UDFs, and Commands Reference Guide 9.1.0.0*.

Chapter 4

Netezza Database Protector

4.1 Introduction

The Protegrity Database Protector for IBM Netezza (Netezza DB Protector) has been optimized to work with SPUs that can only be accessed from the host in a fast, parallel, and multi-node IBM Netezza appliance.

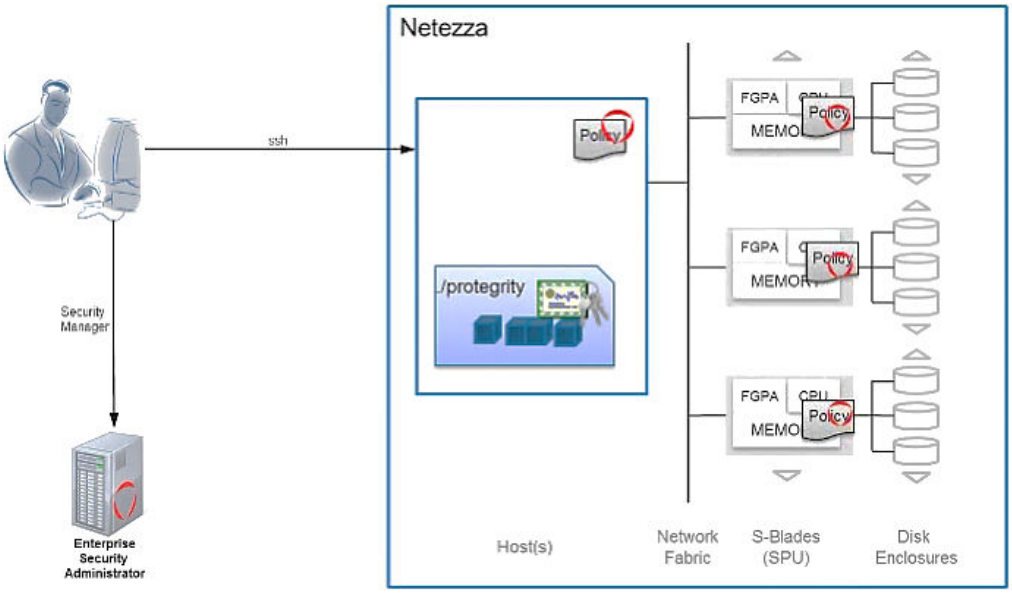

Figure 4-1: Netezza DB Protector Setup

For this section, we have considered the default directories that are available to you in the package. However, you can change the names to suit your company's requirements. For example, the default Netezza DB Protector location is */nz/export/tools/protegrity* but the location can be customized for your current Netezza installation depending on the partition shared between the Host and SPUs.

The Netezza DB Protector creates two different libraries, one for the host called *pepnetezza_host.plm* and another for the SPUs *pepnetezza_spu<n>.plm*.

Netezza DB Protector supports varchar, integer, real, and date data types.

> **Note:**
> The File Protector (FP) is certified for version 6.6.4.

For more information on these data types, refer to *Protegrity Protection Methods Reference Guide 9.1.0.0*.

The following figure shows the task flow specifics for setting up the Netezza DB Protector.

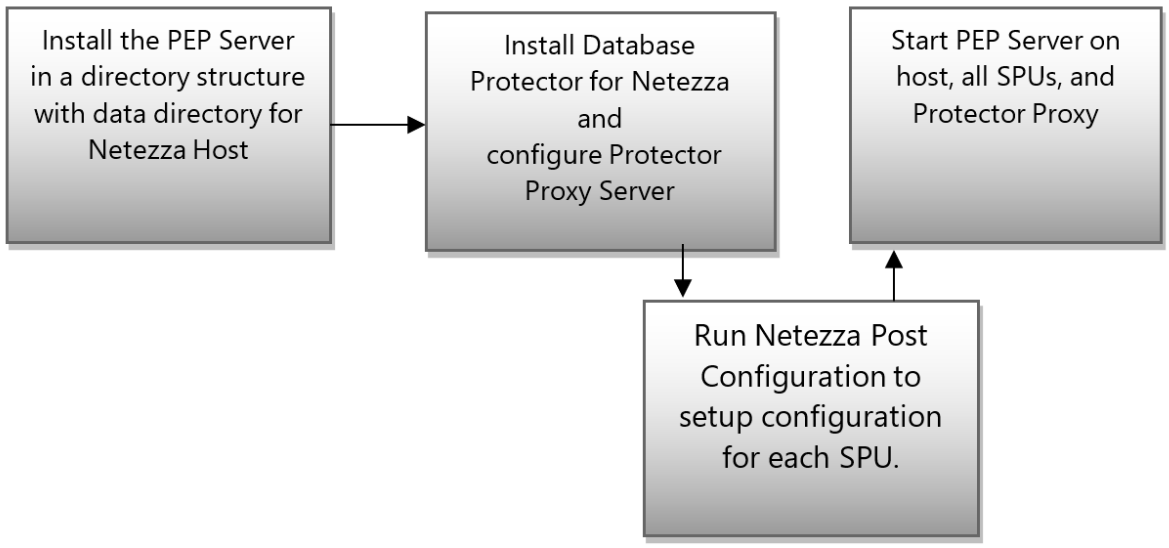

Figure 4-2: Netezza DB Protector Setup Task Flow

4.1.1 Netezza Proxy Servers

The SPUs in Netezza database architecture are connected only to the host. None of the SPUs can be directly accessed by ESA or can reach back to ESA. Proxy servers are used to provide an indirect means of communicating with ESA.

4.2 Requirements

Netezza DB Protector works with ESA which is bundled in the package. As for hardware requirements, we have mentioned the configuration for different databases.

4.2.1 Hardware Requirements

Netezza DB Protector mainly comprises of the UDFs and PEP Server databases. The table below provides the hardware requirements for the same.

Configuration	Memory
PEP Server	50MB per node

Configuration	Memory
PEP Server repository	50MB
Shared Memory	≥ 500MB

Keep another point in mind – when installing the PEP Server, you need to enter the maximum data size to be allocated by the UDFs. This value is 500 by default but can be increased depending on your requirement. When you calculate the data size, add the space for the overheads. For example:

- For data that would be tokenized using non-length preserving tokens, add an overhead of 6% approximately to the original data size.
- For AES-encrypted data with blocks of 16 bytes, an additional 16 bytes is required for CRC or IV.

For more information on PEP Server installation, refer to *Installation Guide 9.1.0.0*.

4.3 Installing and Uninstalling Netezza Database Protector

This section discusses the details of all the steps required for installing and uninstalling the Netezza Database Protector. The following steps broadly describe how Netezza Database Protector is installed.

- Ensure that the prerequisites for installing the Netezza Database Protector are met.
- Download and extract the Netezza Database Protector package to the right directories.
- Install the PEP Server.
- Install the Protector Proxy.
- Install the UDFs.
- Configure SPUs for PEP Server and Protector Proxy.
- Start the PEP Server.

4.3.1 Prerequisites

Ensure the following prerequisites are met before installing Netezza DB Protector on a Netezza setup.

1. The ESA appliance is installed, configured, and running.
2. You have the IP address or host name of the ESA noted down.
3. You have root access to the operating system.
4. The Netezza host and SPUs are installed, configured, and running.
5. You have DBA rights to the Netezza database.
6. Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.

   For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Protegrity Policy Management Guide 9.1.0.0*.

4.3.1.1 Prerequisite tasks before installing Netezza

These are some of the other tasks which, if completed, help you during installation and configuration.

- Log in to the Netezza file system as *nz* user with the following command.

```
# su -nz
```

- Check and note the Netezza version available in your machine to verify that it is compatible with the Netezza Database Protector that you are about to install.
- Check and note the path to the directories that are shared through NFS between the Netezza Host and all SPUs. Following are the directories:
  - Directory from where files are exported by Netezza Hosts: `/nz/export/tools/`
  - Directory where these files are mounted on Netezza SPU: `/var/opt/nz/host/tools`
- Create a directory on the Netezza Host in the shared directory.

```
$ mkdir /nz/export/tools/protegrity
```

- The SPUs are not accessible directly by the PEP server. Therefore, use the the allowed servers option, which lets you add multiple servers, that is part of the Data Store creation process on the ESA to deploy policies and Proxy servers for role_member retrieval and to collect audit records from each SPU node.

  For more information about Data Store creation, refer to section *Working with Data Stores* in the *Protegrity Policy Management Guide 9.1.0.0*.

4.3.2 Downloading and Extracting Protector Package

Before you begin
Copy the package, ***DatabaseProtector_<OS>-<arch>_<Netezza Distribution>-<bit size>_<build number>.tgz***, that you have received from Protegrity to a temporary directory in the Netezza host machine. One example of the Netezza Database Protector package file could be ***DatabaseProtector_RHEL-5-64_x86-64_Netezza-32_7.2.0.xx.tgz***.

To Extract the Netezza Database Protector Tar File on the Netezza Host:

1. Copy the Netezza Database Protector package, received earlier to the shared directory, ***/nz/export/tools/protegrity*** that you have already created on the Netezza host.
2. Extract the package, ***DatabaseProtector_<OS>-<arch>_<Netezza Distribution>-<bit size>_<build number>.tgz*** to the same directory.

After extraction, make sure that the following tar files are present. These files are required for installation:

- ***PepServerSetup_Linux_<bit size>_<build-number>.sh***
- ***ProxySetup_Linux_<bit size>_<build-number>.sh***
- ***PepNetezza<n.n>Setup_Linux_<bit size>_<build-number>.sh***
- ***NetezzaPostConfig.sh***

> **Note:** For information about the installing Protector proxy, refer to section *Protector Proxy*.

4.3.3 Installing Netezza Database Protector on Netezza Host

4.3.3.1 Install DPS

To install DPS:

1. To install the PEP server in */nz/export/tools/protegrity* directory, navigate to the directory and run the following command.

   ```
   # ./PepServerSetup_Linux_<bit size>_<build number>.sh
   ```

   Enter the ESA host name when prompted.
2. Install PEP UDF functions on the Netezza Host.

   ```
   # ./PepNetezza<n.n>Setup_Linux_<bit size>_<build number>.sh
   ```

   Enter the maximum data size that must be allocated by the database UDFs.

   > **Note:**
   > The default value is 500 characters. You must modify the default value in this step as per your requirements for maximum character length.

3. Change the access rights to the PEP server directory to *755*.
4. Install the Protector Proxy.

   ```
   # ./ProxySetup_Linux_<bit size>_<build number>.sh -esa <ESA IP> -dir /nz/export/tools/
   protegrity
   ```

   For information about the installing Protector proxy, refer to section *Protector Proxy Installation*.
5. Run this command to setup post-configuration.

   ```
   # ./Netezza<n.n>_PostConfigSetup_Linux_<bit size>_<build number>.sh
   ```

6. When prompted, enter the database name where the UDT functions reside.

4.3.3.2 Create PEP UDFs

Before you begin
Return to Netezza user, *nz*, and execute these steps.

To create PEP UDFs:

1. Launch *nzsql* and create the database **`<database name>`**.

   ```
   # nzsql
   SYSTEM(ADMIN+ => create database <database name>;
   SYSTEM(ADMIN) => \q
   ```

2. Change the directory to **`/nz/export/tools/protegrity/defiance_dps/pep/sqlscripts/netezza`** and run the following SQL script to create PEP functions in the database.

   ```
   # nzsql -d <database name> –f createobjects.sql
   ```

To verify that all functions are installed in the database, run the following command.

```
# nzsql -c 'show function' -d <database name> | grep -i PTY
```

3. Change directory to the **`/nz/export/tools/protegrity/defiance_dps/data`** directory.

4.3.4 Configuring Netezza Database Protector

4.3.4.1 Configure PEP Server Configuration Files

To configure PEP server configuration files:

1. To configure the PEP server configuration file, open the *pepserver.cfg* file located in the `/nz/export/tools/protegrity/defiance_dps/data` directory.
2. Edit the *pepserver.cfg* file based on your requirements.

4.3.4.2 Configure PEP Server and Post Configuration Files

This section describes the series of steps to configure the PEP server and the post configuration files.

Before you begin
All the post configuration files for Netezza are available in the`/nz/export/tools/protegrity/defiance_dps/postconfigure` directory and its subdirectories.

To configure PEP server and post configuration files:

1. Change the directory to `/nz/export/tools/protegrity/defiance_dps/postconfigure/scripts` directory.
2. Execute the SQL script to create the Protegrity user-defined table functions (UDTFs) in the database PROTEGRITY. This script verifies the existence of SPUs and creates data directories for all SPUs.

   ```
   # ./post_config.sh
   ```

   The output message displays the existing SPUs and the created data directories.
3. If you must increase the maximum size in bytes for the shared memory segment, then set the *SHMMAX* parameter on the Netezza host.

   The following snippet displays a sample of this setting. It increases the *SHMMAX* value to 500 MB.

   ```
   SHMMAX= cat /proc/sys/kernel/shmmax
   echo "kernel.shmmax=$SHMMAX" >> /etc/sysctl.conf
   echo 500000000 > /proc/sys/kernel/shmmax
   sysctl -w kernel.shmmax=500000000
   ```
4. From the */nz/export/tools/protegrity/defiance_dps/postconfigure/scripts* directory, execute the following script to check the added event handlers.

   ```
   ./event_show.sh
   ```

   The following output message appears:

   ```
   Field Name      Value
   --------------- ------------
   Name             PtyStartup
   On               yes
   Event Type       Sys State Changed
   ```

```
Event Args Expr  $previousState != online && $currentState == online
Notify Type      Run Command
Destination      /nz/export/tools/protegrity/defiance_dps/postconfigure/scripts/p
CC
Message          NPS system $HOST went online at $eventTimestamp $eventSource.
Body Text
Call Home        no
Aggregate Count  0
```

> **Note:** The Protector proxy must be installed on the same node where the PEP server is running in the */nz/export/tools/protegrity* directory.

4.3.5 Verifying Netezza Database Protector Installation and Configuration

This section is not mandatory but helps you to verify the installation and configuration steps before you start Netezza DB Protector.

➤ To verify Netezza DB Protector installation and configuration:

1. The directory structure and file contents of the Netezza Host and SPUs should be similar to the following.

   ```
   cd /nz/export/tools/protegrity/defiance_dps/data_spu<n>
   ```

2. The SPU data subdirectories mode should be *rwx* for '*others*'. Modify this mode, if required.

   # chmod –R o+rwx data_spu*

3. All SPU hostnames should be listed on the text file using the following command.

   # /nz/export/tools/protegrity/defiance_dps/postconfigure/scripts/list_spus.sh

4. Create the user-defined table functions in the database by performing the following steps.

   a. Change the directory to */nz/export/tools/protegrity/defiance_dps/postconfigure/sqlscripts* and execute the following SQL script:

      # nzsql -d <database name> –f create_pty_control.sql

   b. To verify that the table functions have been created, run the following command:

      # nzsql -d <database name> –c 'show function' | grep –e PTY_CONTROL –e PTY_HOST

5. Edit the *ptyudtfctrl.sh* script file and check if the specified database connection is accurate.

   ```
   cd /nz/export/tools/protegrity/defiance_dps/postconfigure/scripts

   vi ptyudtfctrl.sh
   ```

4.3.6 Using Netezza Database Protector

After the Netezza DB Protector is installed, a specific package of User Defined Table Function (UDTF) is used to start, stop, and show the status of the PEP server since the ESA cannot directly access the SPU. The function, *pty_control* is called, which executes OS commands on all connected SPUs. A limitation of this function is that it cannot generate error handling messages, and to do so an analysis of the PEP server log files is required.

In a Netezza system, the ESA cannot directly access the OS of the SPUs. To start and stop the PEP server for each SPU, the UDTF function *PTY_CONTROL* must be called. This UDTF executes OS commands on all connected SPUs. There is a limitation on error handling that can be provided so that error messages caused by start or stop operations can be handled by normal analysis of the PEP server log files.

When the Netezza Database Protector is installed, the default location is `/nz/export/tools/protegrity` but the location can be customized for the Netezza installation.

A normal installation of the IBM Netezza Database Protector creates the following directory structure:

```
${DEFAULT_PROTEGRITY_PATH}/defiance_dps
|
|-- bin
|-- data
|-- data_spu0101
|-- data_spu0102
|-- ...
|-- data_spuXXYY
|-- pep
|-- postconfigure
```

The *pty_control* table function must be forced to process across all SPUs.

This can be done by running the script *ptyudtfctrl.sh*, where specific SQL statements are executed to start, stop, or check the status of the PEP servers across all SPUs. The following example shows the SQL statement to start the PEP servers across all SPUs.

```
SELECT f.* FROM _v_dual_dslice AS vds,
(SELECT MIN(ds_id) AS mds FROM _v_dslice GROUP BY ds_prihwid) AS vs,
TABLE ( pty_control(vds.dsid, 'START',null) )
AS f WHERE vs.mds = vds.dsid
```

During the post-configuration installation, another overall main start and stop script, *ptyctrl.sh*, is created and saved in the `defiance_dps/bin` directory.

This script includes the *ptyudtfctrl.sh* script to start up the PEP servers across all SPUs and also start or stop scripts for the PEP server and Proxy servers on the Netezza Host.

The status of a node can also be checked by running the following script with a parameter:

`ptyctrl.sh start/stop/status`

This script, *ptyctrl.sh*, should normally be used when starting, stopping, and obtaining the status of PEP servers, Protector Proxy on the master and all SPUs of a Netezza multi-node system.

During post-configuration after installation, a script named *ptyctrl.sh* is created in the */nz/export/tools/protegrity/defiance_dps/bin* directory. This script can start up, show the status, and stop all the DPS servers of the Netezza DB Protector.

1. Change the directory to `/nz/export/tools/protegrity/defiance_dps/bin` directory.
2. Start all servers on the Netezza Host and SPUs using the following command.

   ```
   # ./ptyctrl.sh start
   ```

   The following output message appears.

   ```
   Server status...
   Service is running as PID=17595
   pepserver -dir /nz/export/tools/protegrity/defiance_dps/data is not running!
    HOSTNAME | CODE |       MESSAGE
   ----------+------+---------------------
    spu0101  |    0 | PEP Server running.
   (1 row)
   ```

Your Netezza DB Protector is now up and running.

4.3.6.1 Create Policy in Policy Management

The Protector requires policies to ensure that unauthorized users do not access the data and that the correct data is protected and unprotected.

For more information about creating policies, refer to the *section 6 Creating and Deploying Policies* in the *Protegrity Policy Management Guide 9.1.0.0*.

4.4 Uninstallation

4.4.1 Uninstalling the UDFs

➤ To Uninstall the UDFs:

1. Login to the Netezza File System.
2. Navigate to the `/nz/export/tools/protegrity/defiance_dps/pep/sqlscripts/netezza` directory.
3. Login to the database.
4. Execute the following query to drop UDF.
   ```
   # nzsql –d <database name> -f dropobjects.sql
   ```
5. Navigate to the `~/defiance_dps/postconfigure/sqlscripts` directory and execute the following query.
   ```
   # nzsql -d <database name> -f drop_pty_control.sql
   ```
6. Navigate to the `~/defiance_dps/postconfigure/scripts` directory and execute the `./event_delete.sh` command.
7. To verify that the UDFs have been removed, run the following command.
   ```
   # nzsql -d <Database Name, default: PROTEGRITY> -c 'show function' | grep -i PTY
   ```
   The output must display zero rows.

4.4.2 Uninstalling the PEP Server

➤ To uninstall the PEP server:

1. Stop the PEP server by executing the *ptyctrl.sh* command.
2. Remove the `/nz/export/tools/protegrity/defiance_dps/` directory.
   The `/nz/export/tools/protegrity/defiance_dps/` directory must be empty.

> **Note:** For information about uninstalling the Protector proxy, refer to the section *Installing and Uninstalling Protector Proxy*.

4.5 Netezza DB Protector UDFs

The APIs and UDFs provided with the Netezza Protector can be used to accomodate custom requirements.

Detailed information about the UDFs provided and their related parameters are provided in the *APIs, UDFs, and Commands Reference Guide 9.1.0.0*.

4.6 Netezza Database Protector Examples

The following two sections have an example each for encryption and tokenization using Netezza Protector. These two examples are also available with the package in the location */nz/export/tools/protegrity/defiance_dps/pep/sqlscripts/netezza*. The flow of the example has been mentioned as comments within the code.

4.6.1 Encryption Example

```
-- Protegrity User Defined Functions sample script.
--
-- NOTE: Please change the following 'tags' before executing the script:
-- '<data element>' : Data element name for encryption.
-- Copyright (c) 2013 Protegrity USA, Inc. All rights reserved
--
-----------------------------------------------------------------------
--DROP TABLE SAMPLE_PTY;
--DROP TABLE SAMPLE_BAK;
--DROP VIEW SAMPLE;
-----------------------------------------------------------------------
--
-- SAMPLE: Table with no protection. Contains sample data.
-----------------------------------------------------------------------
CREATE TABLE SAMPLE (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL
);
INSERT INTO SAMPLE values ('PTY_IVP_FPRTEST_CCN', 'test', 123456789, 987654321, date
'2013-02-15', date '2013-02-15');
ALTER TABLE SAMPLE RENAME TO SAMPLE_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE_PTY: Same as SAMPLE but with protection added for
-- columns, which are encrypted when the
-- table is loaded from SAMPLE_BAK.--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE_PTY (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(48) NOT NULL,
RATING INTEGER NOT NULL,
REFN VARCHAR(16) NOT NULL,
BIRT DATE NOT NULL,
LUPD VARCHAR(16) NOT NULL
);
INSERT INTO SAMPLE_PTY("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") SELECT
CCN,
PTY_VARCHARENC(LNAM,'<data element>'),
RATING,
PTY_INTEGERENC(REFN,'<data element>'),
BIRT,
PTY_DATEENC(LUPD,'<data element>')
FROM SAMPLE_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE_: This is a view that shows how data is unprotected using the UDFs.
-- Data is selected from the 'SAMPLE_PTY' table.
-- The name of this view is the same as the original table
--
-----------------------------------------------------------------------
CREATE VIEW SAMPLE ("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") AS SELECT
CCN,
PTY_VARCHARDEC(LNAM,'<data element>'),
RATING,
PTY_INTEGERDEC(REFN,'<data element>'),
BIRT,
```

```
PTY_DATEDEC(LUPD,'<data element>')
FROM SAMPLE_PTY;
```

4.6.2 Tokenization Example

```
-----------------------------------------------------------------------
-- Protegrity User Defined Functions sample script.
--
-- NOTE: Please change the following 'tags' before executing the script:
-- - <data element1> - dataelement used for protecting varchar
-- - <data element2> - dataelement used for protecting integer
-- - <data element3> - dataelement used for protecting date
-- Copyright (c) 2013 Protegrity USA, Inc. All rights reserved
--
-----------------------------------------------------------------------
--DROP TABLE SAMPLE_PTY;
--DROP TABLE SAMPLE_BAK;
--DROP VIEW SAMPLE;
-----------------------------------------------------------------------
--
-- SAMPLE: Table with no protection. Contains sample data.
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL
);
INSERT INTO SAMPLE values ('PTY_IVP_FPRTEST_CCN', 'test', 123456789, 987654321, date
'2013-02-15', date '2013-02-15');
ALTER TABLE SAMPLE RENAME TO SAMPLE_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE_PTY Same as SAMPLE but with protection added for
-- columns, which are tokenized when the
-- table is loaded from SAMPLE_BAK.
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE_PTY (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL
);
INSERT INTO SAMPLE_PTY("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") SELECT
CCN,
PTY_VARCHARINS(LNAM,'<data element1>'),
RATING,
PTY_INTEGERINS(REFN,'<data element2>'),
BIRT,
PTY_DATEINS(LUPD,'<data element3>')
FROM SAMPLE_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE Same as SAMPLE_PTY. But data is detokenized
-- when SAMPLE is loaded from SAMPLE_PTY.
--
-----------------------------------------------------------------------
CREATE VIEW SAMPLE ("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") AS SELECT
CCN,
PTY_VARCHARSEL(LNAM,'<data element1>'),
RATING,
PTY_INTEGERSEL(REFN,'<data element2>'),
BIRT,
PTY_DATESEL(LUPD,'<data element3>')
FROM SAMPLE_PTY;
```

Chapter 5

Oracle Database Protector

This section describes about the Oracle Database Protector and provides information about setting up, installing, uninstalling, and upgrading the Oracle Database Protector.

5.1 Setup Requirements and User Privileges for Windows and UNIX

The Oracle Database Protector (Oracle DB Protector) provides guidelines to secure access to sensitive data in Oracle databases, that include the Oracle Database on a single node, Oracle Real Application Clusters (RAC), and the Oracle Exadata appliance.

Supported Versions

The following versions of Oracle are supported:

Oracle	Detailed version number
Oracle 12	12c
Oracle 18	18c
Oracle 19	19c
Oracle Exadata	12c, 18c, and 19c

Environment Specifications

Before setting up the Database Protector, ensure that your configuration meets the minimum requirements. The following table describes the environment requirements.

Table 5-1: Environment Requirements

Configuration	Free Disk Space on Windows	Free Disk Space on UNIX
PEP server	50 MB per node	50 MB per node
User Defined Functions (UDFs) and Procedures	10 MB	10 MB

Configuration	Free Disk Space on Windows	Free Disk Space on UNIX
RAM	4 GB	4 GB

Note: For systems under heavy load with auditing turned on, the space required for the PEP server (20 MB per node) can be higher.

5.1.1 User Privileges

The Oracle Database Protector installation can be broadly divided into installing the the PEP server and installing the UDFs. The PEP server installation establishes the connection between the ESA and the Database Protector, while the UDFs use the policies to enforce protection on the data.

User for retrieving users from Oracle Database

For policies to be defined in the ESA, users can be imported from any of the multiple sources such as Active Directory (AD), file, or an Oracle database. If you want to pull users from an Oracle database, a membersource must be created. The following information applies if the users must be pulled from an Oracle database.

To retrieve users from the Member Source Server, you must either create a functional database user with create session granted or use an existing user with create session granted, and then grant the following two specific grants:

- Grant select on sys.dba_roles to *protegrity*
- Grant select on sys.dba_role_privs to *protegrity*

Where, *protegrity* is the functional user created.

User for installing UDFs

After the PEP server is installed, the UDFs can be installed on the Oracle Database server. A functional database user with the following are the privilege rights must be created.

- Grant unlimited tablespace to USER1
- Grant create session to USER1
- Grant select any table to USER1
- Grant create library to USER1
- Grant create procedure to USER1
- Grant drop public synonym to USER1
- Grant create public synonym to USER1
- Grant create table to USER1
- Grant create view to USER1

Where, USER1 is the functional user created.

5.2 Prerequisites

Ensure that the following prerequisites are met:

- The Enterprise Security Administrator (ESA) and the Protegrity Storage Unit (PSU) appliances are installed, configured, and running.

  For more information about the installing, configuring, and using the Protegrity Storage Unit (PSU) for storing logs in the Audit Store, refer to the *Protegrity Storage Unit Guide 9.1.0.0*.

  For more information about the connection between the ESA, PSU, and the Protector, refer to the section *12.1 General Architecture* in the *Protegrity Installation Guide 9.1.0.0*.

- The IP address or host name of the ESA and the PSU is noted.
- The Oracle Database is installed and configured.

  For more information about the Oracle Database, refer to the *Oracle® website* for assistance.

- Ensure that Policy Management (PIM) has been initialized on the ESA. The initialization of PIM ensures that cryptographic keys for protecting data and the policy repository have been created.

  For more information about initializing the PIM, refer to *section 4 Initializing the Policy Management* in the *Policy Management Guide 9.1.0.0*.

- Download and save the Oracle DB Protector, *DatabaseProtector_<OS>-<arch>_<Oracle distribution>-64_<version>.tgz*, made available by Protegrity.
- Even if it is not mandatory, take a backup of the databases where the Oracle DB Protector and the UDFs would be installed.
- Access to the server as the *root* user, should be available to you.
- Access to the Oracle database as the *sysdba* superuser, should be available to you.

5.3 Configuring the Environment Variables for Oracle Database

The Oracle DB Protector can be installed by the root user and Oracle admin user. This section discusses the installation using the root user. Wherever possible, the oracle commands for Oracle admin user would be provided as well.

To use the Oracle DB Protector, you must configure environment variables as per the Oracle version that you are using.

5.3.1 Configuring the *extproc.ora* Environment Variable

The *extproc.ora* file can be found in the `$ORACLE_HOME` directory. Execute the following command to identify the location of the *extproc.ora* file.

```
find . -name extproc.ora 2>/dev/null
```

Add the following environment variable in the *extproc.ora* file that enables you to use the Protegrity UDFs.

```
SET EXTPROC_DLLS=ANY
```

5.4 Installing the Oracle Database Protector

This section provides information to set up the Oracle Database Protector.

Ensure that the following order of installation is followed.

Order of Installation	Description
1	*Setting up the Environment Specification*
2	*Verifying the User Privileges*
3	*Configuring the Environment Variables*
4	*Prerequisites*
5	*Installing the Log Forwarder*
6	*Installing the PEP server*

5.4.1 Preparing the Server

This initial installation is always done on the Oracle server machine.

To prepare the server:

1. Login to the server as the *root* user.
2. To install the Oracle DB Protector, select either your own directory or the default directory.
   For example, if you select the default directory `/opt`, then create `/opt/protegrity`, if it is not present.

   > **Note:** Grant recursive 755 permissions for the `/opt/protegrity` directory. If the directory does not have these permissions, then the Protegrity UDFs cannot be installed and the following error is returned for the *.plm* file in the `~/defiance_dps/pep` directory.
   > `File not found`

3. Map the policies with the required Database users and groups to enable them to use the UDFs.
   Before the Oracle Database UDFs are installed, the following tasks must be completed:
   - Install the PEP server
   - Find the status of the PEP server

5.4.2 Installing the Log Forwarder

This section describes the steps to install the Log Forwarder.

To install the Log Forwarder:

1. Save the Oracle DB Protector package, *DatabaseProtector_<OS>-<arch>_<Oracle distribution>-64_<version>.tgz*, in the `/opt/protegrity` directory.
2. Login to the server as the *root* user.
3. Navigate to the `/opt/Protegrity` directory and unpack the installation package using the following command.
   ```
   # tar zxf DatabaseProtector_<OS>-<arch>_<Oracle distribution>-64_<version>.tgz
   ```
   For example, if the name of the package is *DatabaseProtector_LINUX-ALL-64_x86-64_Oracle-ALL-64_9.1.0.0.xx.tgz*, then run the following command.

   ```
   # tar zxf DatabaseProtector_LINUX-ALL-64_x86-64_Oracle-ALL-64_9.1.0.0.xx.tgz
   ```

   > **Note:** If you are using the Oracle RAC or the Oracle Exadata appliance, then run the command on all the Oracle Database servers.

   Ensure that the following self-extracting shell executables are available:

   - *LogforwarderSetup_Linux_x64_<version>.sh*
   - *PepServerSetup_Linux_x64_<version>.sh*
   - *PepOracleSetup_Linux_x64_<version>.sh*
4. Run the Log Forwarder installer using the following command. Follow the screen prompts that appear during the installation process.

```
./LogforwarderSetup_Linux_x64_<version>.sh
```

Caution: It is mandatory to install the Log Forwarder component before installing the PEP server component to ensure that the Oracle Database Protector is configured correctly.

Note: If you are using the Oracle RAC or the Oracle Exadata appliance, then run the command on all the Oracle Database servers.

5. Enter the Elasticsearch host name or the IP address.
   The *Logforwarder* is installed successfully in the `/opt/protegrity/fluent-bit` directory.
6. Start the Log Forwarder by running the following command.
   ```
   ./logforwarderctrl start
   ```

Note: If you want to change the authentication, then refer to the section *Configuring Security for the Log Forwarder* in the *Audit Store Guide 9.1.0.0*.

5.4.3 Installing the PEP Server

This section describes the steps to install the PEP server.

To install the PEP server:

1. Run the PEP server installer using the following command. Follow the screen prompts that appear during the installation process.
   ```
   ./PepServerSetup_Linux_x64_<version>.sh
   ```

Caution: Ensure that the ESA is installed and running with the *HubController* service status as *Running*, to enable downloading of the certificates automatically.

Note: If you are using Oracle RAC or Oracle Exadata appliance, then run the command on all the Oracle Database servers.

2. Enter the ESA hostname or IP Address.
3. Enter the credentials for downloading the certificates.
   The certificates are successfully downloaded and stored in the the `/opt/protegrity/defiance_dps/data` directory and the PEP server is installed successfully in the `/opt/protegrity/defiance_dps` directory.
4. Set the *EXTPROC_DLLS* parameter in the *extproc.ora* file located in the `$ORACLEHOME/hs/admin/` directory on the Oracle Database server to the following value.
   ```
   SET EXTPROC_DLLS=ANY
   ```

Note: You can modify the PEP server configuration parameters as per your requirement. For more information about the usage of the parameters of the PEP Server configuration file *pepserver.cfg*, refer to the section *Appendix A: PEP Server Configuration File* in the *Protegrity Installation Guide 9.1.0.0*.

5.4.4 Granting Permissions to the Required Files and Directories

To Grant Permissions to the Required Files and Directories:

Run the following command to grant access to the required files and directories.

```
chmod -R 755 /opt/protegrity/
```

5.4.5 Finding Status of the PEP Server

To Find the Status of the PEP Server:

1. Run the following command to find the status of the PEP server.
   ```
   /opt/protegrity/defiance_dps/bin/pepsrvctrl status all
   ```
2. If the PEP server is not running, then start the PEP server using the following command.
   ```
   /opt/protegrity/defiance_dps/bin/pepsrvctrl start
   ```

> **Note:** If you are using Oracle RAC or Oracle Exadata appliance, then run the command on all the Oracle Database servers.

> **Note:** To check the PEP server logs on the ESA or the PSU, login to the ESA Web UI and navigate to **Analytics** > **Forensics**.
> For more information about the Forensics, refer to the section *Working with Forensics* in the *Protegrity Analytics Guide 9.1.0.0*.

5.4.6 UDF Installation

This section provides detailed information on the configuration of User Defined Functions (UDFs) and Extended Store (XP) procedures defined for the type of the installed database.

5.4.6.1 Prerequisites

Ensure that the following prerequisites are met before installing the UDFs:

- Verify the required User Privileges.
  For more information about User Privileges, refer to section *User Privileges*.
- Configure the environment variables based on the Oracle version.
  For more information about configuring the environment variables, refer to section *Configuring the Environment Variables for Oracle Database*.
- The *Logforwarder* is installed.
  For more information about installing the *Logforwarder*, refer to section *Installing the Log Forwarder*.
- The PEP server is installed.
  For more information about installing the PEP server, refer to section *Installing the PEP Server*.

5.4.6.2 Oracle DB Protector UDFs

This topic describes how to create UDFs for Oracle databases from the command line interface.

5.4.6.2.1 Installing UDFs for Oracle Database Protector

This topic describes how to create UDFs for an Oracle database.

Before you begin
Before running the *createobjects.sql* script, configure the *extproc.ora* configuration file, depending on the version of the Oracle database.

For more information about the configuration files, refer to section *Configuring the Environment Variables for Oracle Database*.

> **Note:** If you are using an Oracle RAC or Exadata appliance, then install the UDFs on any one Oracle Database server only.

➤ To install UDFs for Oracle Database Protector:

1. Run the Oracle Database Protector installer using the following command to install the PEP UDFs.
   ```
   ./PepOracleSetup_Linux_x64_<version>.sh
   ```
   > **Note:** If you are using Oracle RAC or Oracle Exadata appliance, then run the command on all the Oracle Database servers.
2. Connect to the database as the *oracle* user with the database owner credentials.
   > **Note:** You can create UDFs using the *oracle* user only. The Oracle DBAs should switch to this user mode by running the following command.
   > ```
   > su – oracle
   > ```
3. Navigate to the `/opt/protegrity/defiance_dps/pep/sqlscripts/oracle` directory.
4. Run the following command to install the UDFs.
   ```
   sqlplus User1/Password1 @createobjects.sql
   ```
   > **Note:** The *User1* and *Password1* are the credentials of the database owner.
   > The symbol “\” is used for Windows and “/” for UNIX environments.

5.4.6.2.2 Checking the Installation of UDFs

➤ To check that all UDFs have been installed:

1. Login to the database *<Oracle UDFs Database>*.
   ```
   su - oracle
   ```
2. Run the following command to get a list of all the installed Oracle DB Protector UDFs.
   ```
   select PROCEDURE_NAME from user_procedures order by 1
   ```
3. To verify that the installation is successful, run either of these Oracle DB UDFs:
   - ```
     select pty.whoami() from dual;
     ```
     Name of the user is displayed.
   - ```
     select pty.getversion() from dual;
     ```
     Version number of the protector is displayed.

4. Logout from the user *oracle*.

5.4.6.2.3 Parallel Execution in the Oracle Database Protector UDFs

Parallel execution reduces the response time for data-intensive operations on large databases. This section describes about the parallel execution of SQL queries in the Oracle database with the Protegrity UDFs. The *PARALLEL_ENABLE* clause has been added in the Protegrity UDFs that enable the SQL queries to be executed in parallel using an SQL hint (PARALLEL) in the queries.

The following code snippet is an example where the *PARALLEL_ENABLE* clause is added in the *ins_varchar2* UDF used for protecting VARCHAR data.

```
FUNCTION ins_varchar2(dataelement IN CHAR, inval IN VARCHAR2, SCID IN BINARY_INTEGER ) RETURN
VARCHAR2 PARALLEL_ENABLE;
```

For more information about the UDFs that have the parallel execution enabled, refer to the section *Oracle User Defined Functions and Procedures* in the *APIs, UDFs, and Commands Reference Guide 9.1.0.0.*

The following code snippet is an example that shows how to achieve parallelism while executing a query using the *PARALLEL* hint.

```
select /*+ PARALLEL(4) */ pty.ins_varchar2('TE_A_S23_L0R0_N',contact_name,0)  from employees;
```

For more information about the methods to enable parallel execution, refer to Oracle's official website.

5.4.6.3 Installing UDFs for Oracle XA Protector

This topic describes additional setup steps required for the Oracle XA protector.

To install the UDFs for Oracle XA protector:

1. Navigate to the `defiance_dps/xcoracle` directory.
2. Run the following command from the $ prompt.
   ```
   loadjava -resolve -verbose -user USER/PASSWORD xcoracle.jar com/protegrity/
   xcoracle/XCOracle.properties
   ```
3. Navigate to the directory where the PEP server is installed. The following files are available in the directory:

Directory	Files
`/defiance_dps/xcoracle`	*xcoracle.jar*
`/defiance_dps/xcoracle/scripts/`	*createobjects.sql*
	dropobjects.sql
	allowsocket.sql grant_java.sql
`/defiance_dps/xcoracle/com/protegrity/xcoracle`	*XCOracle.properties*

4. Run the *allowsocket.sql* script by the *SYS* user.
5. Run the *grant_java.sql* script to allow all users to execute the Java functions by the Oracle user running XC java (for example, it can be JAVAUSR, used for the Oracle XA setup).
6. Run the *createobjects.sql* script by the PEP user (JAVAUSR in our example).

7. Ensure that one of the XC listener ports is activated (uncommented) in the *pepserver.cfg* file as shown in the following code snippet.

```
# By default none of these are active.
# To activate you need to remove the comment of one or both of these rows
listener = tcp, 15910, xc
```

8. Restart the PEP server if you have made any changes to the configuration file.

5.4.7 Enterprise User Security (EUS) in Oracle Database

The Enterprise User Security (EUS) is an important component of Oracle database that allows you to centrally manage the database users across the enterprise. Enterprise users are the users that are defined and managed in a directory. Each enterprise user has an unique identity across the enterprise. The Enterprise User Security relies on the Oracle Identity Management infrastructure, that uses an LDAP-compliant directory service to centrally store and manage the users.

Protegrity supports the following authentication methods:

- Password-based authentication
- SSL-based authentication
- Kerberos-based authentication

In the following list, the type of user is followed by the value returned:

- Password-authenticated enterprise user: nickname (same as the login name)
- Password-authenticated database user: the database username (same as the schema name)
- SSL-authenticated enterprise user: the DN in the user's PKI certificate
- SSL-authenticated external user: the DN in the user's PKI certificate
- Kerberos-authenticated enterprise user: kerberos principal name
- Kerberos-authenticated external user: kerberos principal name; same as the schema name

The Oracle database protector supports the retrieval of the user information using the *AUTHENTICATED_IDENTITY* parameter that returns the identity used in the authentication.

5.4.7.1 Using the EUS Feature

This section describes how to use the EUS feature. The instructions and examples provided in the section use the Kerberos based authentication.

> **Note:**
> - Ensure that the user name in the ESA policy contains only the user name and does not include the domain name. For example, USER1.
> - Currently, only one domain name is supported.

To use the EUS feature:

1. To create a kerberos ticket for the enterprise user, execute the following command:

```
okinit <username>
```

The command prompts for the password of the enterprise user.

```
[oracle@db ~]$ okinit USER1
Kerberos Utilities for Linux: Version 18.0.0.0.0 - Production on 15-DEC-2021 06:07:06

Copyright (c) 1996, 2017 Oracle. All rights reserved.

Configuration file : /u01/app/oracle/product/18.0.0/dbhome_1/network/admin/kerberos/
krb5.conf.
Password for USER1@TESTLAB.COM:
```

2. Type the password.
3. Press ENTER.
4. To verify whether the authentication ticket is generated successfully, execute the following command:

```
oklist
```

5. Press ENTER.

   The command displays the authentication ticket details.

```
[oracle@db ~]$ oklist

Kerberos Utilities for Linux: Version 18.0.0.0.0 - Production on 15-DEC-2021 06:07:37

Copyright (c) 1996, 2017 Oracle. All rights reserved.

Configuration file : /u01/app/oracle/product/18.0.0/dbhome_1/network/admin/kerberos/
krb5.conf.
Ticket cache: FILE:/tmp/krb5cc_54321
Default principal: USER1@TESTLAB.COM

Valid starting Expires Service principal
12/15/21 06:07:06 12/15/21 16:07:06 krbtgt/TESTLAB.COM@TESTLAB.COM
renew until 12/16/21 06:07:06
[oracle@db ~]$
```

6. To login to the Oracle database, execute the following command:

```
sqlplus /@<database_name>
```

7. Press ENTER.

   The SQL prompt appears.

```
[oracle@db ~]$ sqlplus /@orcl

SQL*Plus: Release 18.0.0.0.0 - Production on Wed Dec 15 06:09:45 2021
Version 18.3.0.0.0

Copyright (c) 1982, 2018, Oracle. All rights reserved.

Last Successful login time: Wed Dec 15 2021 05:20:43 -05:00

Connected to:
Oracle Database 18c Enterprise Edition Release 18.0.0.0.0 - Production
Version 18.3.0.0.0

SQL>
```

8. To verify the authentication method, execute the following command:

```
SQL> select sys_context('USERENV','AUTHENTICATION_METHOD') from dual;
```

9. Press ENTER.

   The command displays the authentication method.

```
SQL> select sys_context('USERENV','AUTHENTICATION_METHOD') from dual;
```

```
SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD')
--------------------------------------------------------------------------------
KERBEROS
```

10. Execute a protect operation.

    For example:

```
SQL> select pty.ins_encrypt('AES128', 'Original data', 0) from dual;

PTY.INS_ENCRYPT('AES128','ORIGINALDATA',0)
--------------------------------------------------------------------------------
3713D5C1E058701568115B28885707CA

SQL>
```

11. Execute an unprotect operation.

    For example:

```
SQL> select pty.sel_decrypt('AES128', pty.ins_encrypt('AES128', 'Protegrity', 0) , 0)
from dual;

PTY.SEL_DECRYPT('AES128',PTY.INS_ENCRYPT('AES128','PROTEGRITY',0),0)
--------------------------------------------------------------------------------
Protegrity

SQL>
```

5.4.7.2 Retrieving User Information using the *AUTHENTICATED_IDENTITY* Parameter

This section describes how to use the *AUTHENTICATED_IDENTITY* parameter to retrieve the information of an enterprise user.

➤ To retrieve the user information using the *AUTHENTICATED_IDENTITY* parameter:

To fetch the information of an enterprise user, run the following query:

```
select sys_context( 'userenv', 'AUTHENTICATED_IDENTITY' ) from dual;
```

> **Note:** The *AUTHENTICATED_IDENTITY* parameter holds the information of the enterprise user and returns the identity that is used in the authentication.

5.5 Uninstalling the Oracle Database Protector

This section demonstrates the procedures to uninstall the Oracle Database Protector.

5.5.1 Uninstalling the UDFs

This section describes the procedures to uninstall the Database Protector UDFs.

➤ To uninstall the UDFs:

1. Login to the Oracle Database server.
2. Navigate to the *~/defiance_dps/pep/sqlscripts/oracle* directory.

3. Login to the Oracle database, with the same user that installed the UDFs, by running the following command.
   sqlplus USER1/Password1 @dropobjects.sql

5.5.2 Uninstalling the PEP Server

This section describes the steps to uninstall the PEP server.

➤ To uninstall the PEP server:

1. Stop the PEP server by running the following command.
   /opt/protegrity/defiance_dps/bin/pepsrvctrl stop
2. Remove the *~/defiance_dps/bin* directory.
3. Remove the *~/defiance_dps/data* directory.
4. Remove the *~/defiance_dps/pep* directory.
5. To verify that the PEP server has been removed, check for the existence of the *~/defiance_dps/bin*, *~/defiance_dps/data*, and *~/defiance_dps/pep* directories.
   The directories should not exist.

5.5.3 Uninstalling the Log Forwarder

This section describes the steps to uninstall the Log Forwarder.

➤ To uninstall the Log Forwarder:

1. Stop the Log Forwarder by running the following command.
   /opt/protegrity/fluent-bit/bin/logforwarderctrl stop
2. Navigate to the */opt/protegrity* directory.
3. Remove the *fluent-bit* directory.

5.6 Upgrading the Database Protector

You must uninstall the previous version of the Database Protector before upgrading.

Before you start the upgrade, ensure that the cached audit log files have been moved to the ESA and copy the PEP server configuration file to a temporary directory.

5.6.1 Preparing System for Upgrade

➤ To Prepare System for Upgrade:

1. Stop all the database activity, such as protection or unprotection of data, for the Database Protector.
2. Wait for few minutes to allow the PEP server to move all the cached audit log files to the ESA.
3. Run the following command to stop the PEP server from the CLI.

```
/opt/protegrity/defiance_dps/bin]# .pepsrvctrl stop all
```

For more information about the PEP server, refer to the section *Installing and Uninstalling the PEP Server* in the *Installation Guide 9.1.0.0*.

4. Backup the PEP server configuration file in a temporary directory. The configuration file *pepserver.cfg* is located in the `../defiance_dps/data/` directory.

5.6.2 Dropping all UDF Functions

The UDF Functions for each database must be dropped before upgrading to the latest database protector.

➤ To drop all UDF Functions:

1. Login to the database with a username that is the owner of the UDF functions.
2. Run the *dropobject* script which is located in `../defiance_dps/pep/sqlscripts/postgres` directory.
   The script to drop the standard UDF functions is *dropobjects.sql*.

```
-rw-r--r-- 1 root root 7005 Feb 10 21:06 createobjects.sql
-rw-r--r-- 1 root root 1534 Jan 24 02:23 dropobjects.sql
-rw-r--r-- 1 root root 3032 Jan 24 02:23 sample_enc.sql
-rw-r--r-- 1 root root 3104 Jan 24 02:23 sample_tok.sql
-rw-r--r-- 1 root root 2609 Jan 24 02:23 testscript.sql
```

Figure 5-1: Drop UDF Functions

> **Note:** Some database platforms include specific UDFs for decimal, Unicode, and other data types. If these were also installed, then run the corresponding *dropobjects* script.

3. Remove the `../defiance_dps` directory.

5.6.3 Upgrading the Database Protector

Upgrading the Database Protector involves the installing of the latest version of the protector and restarting the PEP server.

➤ To upgrade the Database Protector:

Before you begin

Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.

For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Policy Management Guide 9.1.0.0*.

1. Install the new version of the Database Protector.
   For the steps, refer to the section *Setup Overview of Windows and Unix*.
2. If the `pepserver.cfg` file has been customized by you, then add the same information in the new configuration file.

5.6.4 Recreating the UDF Functions

After upgrading the database protector, you can recreate the UDF Functions that were dropped.

To re-create the UDF Functions:

1. Login to the database with a username that is the owner of the UDF functions.
2. Execute the *createobjects.sql* script located in `../defiance_dps/pep/sqlscripts/<Database Name>` directory. The script to create the standard UDF functions is `createobjects.sql`.

> **Note:** Some database platforms include specific UDFs for decimal, Unicode, and other data types. If these were also installed, then run the corresponding *createobjects* script.

5.6.5 Starting the PEP Server

The PEP server must be started to ensure that the configurations are applied to the Database Protector.

To start the PEP server:

1. Run the following command to start the PEP server:
   ```
   /opt/protegrity/defiance_dps/bin]# ./pepsrvctrl start
   ```
2. Deploy the policies from the ESA to the Database Protector.

The upgraded Database Protector is now ready to protect, unprotect, or reprotect data.

5.7 Oracle User Defined Functions and Procedures

The APIs and UDFs provided with the Oracle Protector can be used to accommodate custom requirements.

> **Note:**
> The Oracle Database Protector supports username only up to 30 characters in length for protect and unprotect operations.

For more information about the UDFs, refer to the *APIs, UDFs, and Commands Reference Guide 9.1.0.0*.

5.8 Oracle Database Protector Example

In this section, two examples of usage of the Protector is provided, for encryption and for tokenization. Go through the same and replace the values as required. These examples are also included in the database package.

5.8.1 Encryption Example

```
-----------------------------------------------------------------------
-- Defiance DPS User Defined Functions.
```

```
-- Copyright (c) 2011 Protegrity USA, Inc. All rights reserved
--
-- NOTE make sure that follwing is replaced with encryption data element
-- before executing script:
-- - <data element> - dataelement used for encryption
--
-----------------------------------------------------------------------
-----------------------------------------------------------------------
--
-- SAMPLE1 - Two tables and one view is created as follows:
--
-- Run the sample_enc.sql job to verify protect / unprotect of datatypes
-- VARCHAR, DATE and INTEGER.
--
-----------------------------------------------------------------------
--DROP TABLE SAMPLE1_BAK;
--DROP TABLE SAMPLE1_PTY;
--DROP VIEW SAMPLE1;
-----------------------------------------------------------------------
--
-- SAMPLE1 Base table with no protection
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1 (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL,
ID INTEGER NOT NULL, PRIMARY KEY(ID)
);
INSERT INTO SAMPLE1 values ('PTY_IVP_FPRTEST_CCN', 'PTY_IVP_FPRTEST_LNAME', 123456789,
987654321, date '2013-02-15', date '2013-02-15', 1);
ALTER TABLE SAMPLE1 RENAME TO SAMPLE1_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE1_PTY Same as SAMPLE1 but with protection added for
-- columns, which are encrypted / tokenized when the
-- table is loaded from SAMPLE1_BAK.
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1_PTY (
CCN VARCHAR(32) NOT NULL,
LNAM RAW(48) NOT NULL,
RATING INTEGER NOT NULL,
REFN RAW(48) NOT NULL,
BIRT DATE NOT NULL,
LUPD RAW(48) NOT NULL,
ID INTEGER NOT NULL, PRIMARY KEY(ID)
);
INSERT INTO SAMPLE1_PTY(CCN, LNAM, RATING, REFN, BIRT, LUPD, ID) SELECT
CCN,
PTY.INS_ENCRYPT_VARCHAR2('<data element>', LNAM, 0),
RATING,
PTY.INS_ENCRYPT_INTEGER('<data element>', REFN, 0),
BIRT,
PTY.INS_ENCRYPT_DATE('<data element>', LUPD, 0),
ID
FROM SAMPLE1_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE1 Same as SAMPLE1_PTY. But data is decrypted /detokenized
--
--
-----------------------------------------------------------------------
CREATE VIEW SAMPLE1 (CCN, LNAM, RATING, REFN, BIRT, LUPD, ID) AS SELECT
CCN,
PTY.SEL_DECRYPT_VARCHAR2('<data element>', LNAM, 0),
RATING,
PTY.SEL_DECRYPT_INTEGER('<data element>', REFN, 0),
BIRT,
PTY.SEL_DECRYPT_DATE('<data element>', LUPD, 0),
```

```
ID
FROM SAMPLE1_PTY;
```

5.8.2 Tokenization Example

```
-----------------------------------------------------------------------
-- Defiance DPS User Defined Functions.
-- Copyright (c) 2011 Protegrity USA, Inc. All rights reserved
--
-- NOTE make sure that following is replaced with encryption data element
-- before executing script:
-- - <data element1> - dataelement use for tokenizing varchar
-- - <data element2> - dataelement use for tokenizing integer
-- - <data element3> - dataelement use for tokenizing date
--
-----------------------------------------------------------------------
-----------------------------------------------------------------------
--
-- SAMPLE_PTY - Two tables and one view is created as follows:
--
-- Run the sample.sql job to verify protect / unprotect of datatypes
-- VARCHAR, DATE and INTEGER.
--
-----------------------------------------------------------------------
--DROP TABLE SAMPLE1_BAK;
--DROP TABLE SAMPLE1_PTY;
--DROP VIEW SAMPLE1;
-----------------------------------------------------------------------
--
-- SAMPLE1 Base table with no protection
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1 (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL,
ID INTEGER NOT NULL, PRIMARY KEY(ID)
);
INSERT INTO SAMPLE1 values ('PTY_IVP_FPRTEST_CCN', 'PTY_IVP_FPRTEST_LNAME', 123456789,
987654321, date '2013-02-15', date '2013-02-15', 1 );
ALTER TABLE SAMPLE1 RENAME TO SAMPLE1_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE1_PTY Same as SAMPLE1 but with protection added for
-- columns, which are tokenized when the
-- table is loaded from SAMPLE_BAK.
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1_PTY (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL,
ID INTEGER NOT NULL, PRIMARY KEY(ID)
);
INSERT INTO SAMPLE1_PTY(CCN, LNAM, RATING, REFN, BIRT, LUPD, ID) SELECT
CCN,
PTY.INS_VARCHAR2('<data element1>', LNAM, 0),
RATING,
PTY.INS_INTEGER('<data element2>', REFN, 0),
BIRT,
TO_DATE( PTY.INS_VARCHAR2('<data element3>', TO_CHAR(LUPD,'YYYY-MM-DD'), 0), 'YYYY-MM-DD' ),
ID
FROM SAMPLE1_BAK;
-----------------------------------------------------------------------
```

```
--
-- SAMPLE1 Same as SAMPLE1_PTY. But data is decrypted /detokenized
--
--
-------------------------------------------------------------------------
CREATE VIEW SAMPLE1 (CCN, LNAM, RATING, REFN, BIRT, LUPD, ID) AS SELECT
CCN,
PTY.SEL_VARCHAR2('<data element1>', LNAM, 0),
RATING,
PTY.SEL_INTEGER('<data element2>', REFN, 0),
BIRT,
PTY.SEL_VARCHAR2('<data element3>', TO_CHAR(LUPD,'YYYY-MM-DD'), 0),
ID
FROM SAMPLE1_PTY;
```

Chapter 6

Greenplum Database Protector

This section describes about the Greenplum Database Protector and provides information about setting up, installing, uninstalling, and upgrading the Greenplum Database Protector.

The Greenplum Database protector setup and installation can be classified as installation on a single node or multiple nodes.

- *Setting up Greenplum Database Protector on a Single Node*
- *Setting up Greenplum Database Protector on Multiple Nodes*

6.1 Setup Overview for UNIX

Before setting up the Database Protector, ensure that your configuration meets the minimum requirements. The following table describes the environment requirements.

Table 6-1: Environment Requirements

Configuration	Free Disk Space on UNIX
PEP server	50 MB per node
User Defined Functions (UDFs) and Procedures	10 MB
RAM	4 GB

> **Note:** For systems under heavy load with auditing turned on, the space required for the PEP server (20 MB per node) can be higher.

These are the main tasks for installing the Protegrity Greenplum Database Protector.

- Check the prerequisites for installing the Greenplum Database Protector are met
- Download and extract the Greenplum Database Protector package
- Install the PEP server in the database
- Install the Greenplum Database Protector

- Start the PEP server
- To verify the installation, perform the following tasks:
  - Create and deploy policy
  - Protect a table
  - Review Audit logs for successful protect operation.

The Greenplum Database Protector secures access to sensitive data on the GPDB or Greenplum databases.

6.2 Prerequisites

Ensure the following prerequisites are met:

- The Greenplum Database is installed and configured.

  For more information, refer to *Greenplum® Database 4.0 Installation Guide* for assistance.

- Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.

  For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Protegrity Policy Management Guide 9.1.0.0*.

- Download and save the Greenplum Database Protector, *DatabaseProtector_<OS>-<arch>_<Greenplum distribution>-64_x.x.x.x.tgz*, made available by Protegrity.
- Even if it is not mandatory, take a backup of the databases where the Greenplum Database Protector and UDFs would be installed.
- Access as the *root* user, should be available to you.
- Access as the *gpadmin* superuser for Greenplum database, should be available to you.

6.3 Installing the Greenplum Database Protector on a Single Node

The Greenplum Database Protector can be installed by the *root* user and Greenplum *gpadmin* user. This section discusses the installation using the *root* user. Wherever possible, the *gpadmin* commands would be provided as well.

6.3.1 Creating the Directory to Install the Greenplum Database Protector

The `/opt/protegrity` directory is the default directory to install the Greenplum Database Protector. If the default directory is not present, then create the directory and grant *755* permissions.

6.3.2 Installing the PEP Server

➤ To Install the PEP Server:

1. Upload the Greenplum Database Protector, *DatabaseProtector_<OS>-<arch>_<Greenplum distribution>-64_x.x.x.x.tgz*, to a temporary directory on the Greenplum server machine.
2. Login to Greenplum server machine as the *root* user.
3. Navigate to the temporary directory, created in Step 1, and unpack the installation package.
   ```
   tar xvzf DatabaseProtector_RHEL-6-64_x86-64_Greenplum-4.3-64_x.x.x.x.tgz
   ```
   The following files are extracted:

- *PepServerSetup_<OS>_x64_x.x.x.x.sh*
- *PepGreenplum<x.x>Setup_Linux_x64_x.x.x.x.sh*
- *U.S.Patent.No.6,321,201.Legend.txt*

4. Navigate to the `/opt/protegrity` directory.
5. Run the following PEP server installer first in this location. Follow the screen prompts that appear during the installation process.
   ```
   ./PepServerSetup_Linux_x64_x.x.x.x.sh
   ```
6. Enter the ESA IP address, as requested.
7. Without changing the location, that is `/opt/protegrity` directory, run the Greenplum Database Protector installer. Follow the screen prompts that display during the installation process.
   ```
   ./PepGreenplum<x.x>Setup_Linux_x64_x.x.x.x.sh
   ```

6.3.3 Granting Permissions to the Required Files and Directories

To Grant Permissions to the Required Files and Directories:

Run the following command to grant access to required files and directories.

```
chmod -R 755 /opt/protegrity/
```

6.3.4 Finding Status of the PEP Server

Before the Greenplum Database UDFs are installed, it is recommended to check the status of the PEP server.

To Find Status of PEP Server:

1. Run the following command to find the status of the PEP server.
   ```
   bash /opt/protegrity/defiance_dps/bin/pepsrvctrl status
   ```
2. If the PEP server is not running, then start the PEP server using the following command.
   ```
   bash /opt/protegrity/defiance_dps/bin/pepsrvctrl start
   ```

6.4 Installing and Uninstalling Greenplum Database Protector on Multiple Nodes

The Greenplum Database Protector can be installed both by *root* user and Greenplum *gpadmin* user. This section discusses the installation using the *root* user. Wherever possible, the *gpadmin* commands would be provided as well.

6.4.1 Creating the Directory to Install the Greenplum Database Protector on all the Nodes

The `/opt/protegrity` directory is the default directory to install the Greenplum Database Protector. If the default directory is not present, then create the directory and grant *755* permissions.

> **Note:** If you select the default `/opt/protegrity` directory, then create the `/opt/protegrity` directory on all the nodes.

6.4.2 Installing the PEP Server

To Install the PEP Server:

1. Copy the Greenplum Database Protector, *DatabaseProtector_<OS>-<arch>_<Greenplum distribution>-64_x.x.x.x.tgz*, to a temporary directory on the master.
2. Login to the master node as the *root* user.
3. Navigate to the temporary directory, created in Step 1, and extract the installation package.
   ```
   tar xvzf DatabaseProtector_RHEL-6-64_x86-64_Greenplum-4.3-64_x.x.x.x.tgz
   ```
   The following files are extracted:
   - *PepServerSetup_<OS>_x64_x.x.x.x.sh*
   - *PepGreenplum<x.x>Setup_Linux_x64_x.x.x.x.sh*
   - *U.S.Patent.No.6,321,201.Legend.txt*
4. Navigate to the `/opt/protegrity` directory.
5. Run the following PEP server installer first in this location. Follow the screen prompts that appear during the installation process.
   ```
   ./PepServerSetup_Linux_x64_x.x.x.x.sh
   ```
6. Enter the ESA IP address, as requested.
7. Without changing the location, that is `/opt/protegrity` directory, run the Greenplum Database Protector installer. Follow the screen prompts that display during the installation process.
   ```
   ./PepGreenplum<x.x>Setup_Linux_x64_x.x.x.x.sh
   ```

6.4.3 Granting Permissions to the Required Files and Directories on all the Nodes

To Grant Permissions to the Required Files and Directories on all the Nodes:

Run the following command to ensure that the UDFs are working on all the nodes and access is available to all the files and directories.

```
chmod -R 755 /opt/protegrity/
```

6.4.4 Copying All Files and Directories from the Source Directory to the Destination Directory

To Copy all the Files and Directories from the Source Directory to the Destination Directory:

Run the following command to copy all the files and directories from the source directory to the destination directory.

```
bash -r -h sdw1 -h sdw2 -h sdw3 -h sdw4 -h sdw5 -h sdw6 /opt/protegrity/* =:/opt/protegrity/
```

Here source is the `/opt/protegrity` directory and destination comprises of all the segment nodes, for example, *sdw1* through *sdw6*.

6.4.5 Finding Status of the PEP Server on all Nodes

Before you begin
Before the Greenplum Database Protector UDFs are installed, it is recommended to check the status of the PEP server.

To Find Status of the PEP Server on all the Nodes:

1. Run the following command to login to the *gpadmin* user.
   ```
   su - gpadmin
   ```
2. Run the following command to find the status of the PEP server on all the nodes.
   ```
   bash /opt/protegrity/defiance_dps/bin/pepsrvctrl status
   ```
3. If the PEP server in any node is not running, then start the PEP server using the following command.
   ```
   bash /opt/protegrity/defiance_dps/bin/pepsrvctrl start
   ```

6.5 Installing UDFs of Greenplum Database Protector

This section describes how to create UDFs for Greenplum Database Protector.

To install UDFs for Greenplum Database Protector:

1. Connect to the database as *gpadmin* user with database owner credentials.

   Note: You can create UDFs using the *gpadmin* user only. The Greenplum DBAs should switch to this user mode by running the `su - gpadmin` command.

2. Navigate to the `/opt/protegrity/defiance_dps/pep/sqlscripts/postgres` directory.
3. Run the following command to install the UDFs.
   ```
   psql –d <Greenplum UDFs Database> –f createobjects.sql
   ```

   Note: The command line interface can also be used to create the Greenplum Database Protector UDFs.
   ```
   psql -h mdw -d biadpdev -a -f /opt/protegrity/defiance_dps/pep/sqlscripts/
   postgres/createobjects.sql -U gpadmin
   ```

Name	Type
mdw	Master in Greenplum Database
biadpdev	Database
gpadmin	Username
`/opt/protegrity/defiance_dps/pep/sqlscripts/postgres/createobjects.sql`	Script used to create the UDFs in the database

6.5.1 Verifying the Installation of UDFs

To check that all UDFs have been installed:

1. Login to the database *<Greenplum UDFs Database>*.
   ```
   psql –d <Greenplum UDFs Database>
   ```
2. Run the following command to get a list of all the installed Greenplum DB Protector UDFs.
   ```
   <Greenplum UDFs Database> = # \df pty*
   ```
3. Run either of the following Greenplum DB UDFs to verify that the installation is successful.
   - ```
     select pty_whoami();
     ```
     Name of the user is displayed.
   - ```
     select pty_getversion();
     ```
     Version number from the PEP server file is displayed.
4. Log out from *gpadmin*.

6.6 Uninstalling the Greenplum Database Protector

To uninstall the Greenplum Database Protector, the UDFs and the PEP server need to be uninstalled or removed from the main node. You should be logged in as the *root* user.

6.6.1 Uninstalling the UDFs

To Uninstall the UDFs:

1. Login to the Greenplum file system.
   ```
   # su - gpadmin
   ```
2. Navigate to the `/opt/protegrity/defiance_dps/pep/sqlscripts/postgres` directory.
3. Run the following query to uninstall the UDFs.
   ```
   # psql -d <database name> -f dropobjects.sql
   ```
4. Delete the `~/defiance_dps/pep` directory.

6.6.2 Uninstalling the PEP Server

To uninstall the PEP Server:

1. Navigate to the `~/defiance_dps/bin` directory.
2. Run the following command to stop the PEP server on all the segments.
   ```
   ./pepsrvctrl stop all
   ```
3. Delete the `<install_directory>/defiance_dps` directory.

6.7 Upgrading the Database Protector

You must uninstall the previous version of the Database Protector before upgrading.

Before you start the upgrade, ensure that the cached audit log files have been moved to the ESA and copy the PEP server configuration file to a temporary directory.

6.7.1 Preparing System for Upgrade

To Prepare System for Upgrade:

1. Stop all the database activity, such as protection or unprotection of data, for the Database Protector.
2. Wait for few minutes to allow the PEP server to move all the cached audit log files to the ESA.
3. Run the following command to stop the PEP server from the CLI.
   ```
   /opt/protegrity/defiance_dps/bin]# .pepsrvctrl stop all
   ```
   For more information about the PEP server, refer to the section *Installing and Uninstalling the PEP Server* in the *Installation Guide 9.1.0.0*.
4. Backup the PEP server configuration file in a temporary directory. The configuration file *pepserver.cfg* is located in the `../defiance_dps/data/` directory.

6.7.2 Dropping all UDF Functions

The UDF Functions for each database must be dropped before upgrading to the latest database protector.

To drop all UDF Functions:

1. Login to the database with a username that is the owner of the UDF functions.
2. Run the *dropobject* script which is located in `../defiance_dps/pep/sqlscripts/postgres` directory.
   The script to drop the standard UDF functions is *dropobjects.sql*.

```
-rw-r--r-- 1 root root 7005 Feb 10 21:06 createobjects.sql
-rw-r--r-- 1 root root 1534 Jan 24 02:23 dropobjects.sql
-rw-r--r-- 1 root root 3032 Jan 24 02:23 sample_enc.sql
-rw-r--r-- 1 root root 3104 Jan 24 02:23 sample_tok.sql
-rw-r--r-- 1 root root 2609 Jan 24 02:23 testscript.sql
```

Figure 6-1: Drop UDF Functions

> **Note:** Some database platforms include specific UDFs for decimal, Unicode, and other data types. If these were also installed, then run the corresponding *dropobjects* script.

3. Remove the `../defiance_dps` directory.

6.7.3 Upgrading the Database Protector

Upgrading the Database Protector involves the installing of the latest version of the protector and restarting the PEP server.

To upgrade the Database Protector:

Before you begin

Starting from the version 7.2.0 release, if you are installing the ESA for the first time, ensure that the Policy Management is initialized prior to installing the protector.

For more information about initializing the Policy Management, refer to section *Initializing the Policy Management* in the *Policy Management Guide 9.1.0.0*.

1. Install the new version of the Database Protector.
   For the steps, refer to the section *Setup Overview of Windows and Unix*.
2. If the `pepserver.cfg` file has been customized by you, then add the same information in the new configuration file.

6.7.4 Recreating the UDF Functions

After upgrading the database protector, you can recreate the UDF Functions that were dropped.

To re-create the UDF Functions:

1. Login to the database with a username that is the owner of the UDF functions.
2. Execute the *createobjects.sql* script located in `../defiance_dps/pep/sqlscripts/<Database Name>` directory.
   The script to create the standard UDF functions is `createobjects.sql`.

   Note: Some database platforms include specific UDFs for decimal, Unicode, and other data types. If these were also installed, then run the corresponding *createobjects* script.

6.7.5 Starting the PEP Server

The PEP server must be started to ensure that the configurations are applied to the Database Protector.

To start the PEP server:

1. Run the following command to start the PEP server:
   ```
   /opt/protegrity/defiance_dps/bin]# ./pepsrvctrl start
   ```
2. Deploy the policies from the ESA to the Database Protector.

The upgraded Database Protector is now ready to protect, unprotect, or reprotect data.

6.8 Greenplum Database Protector UDFs

The APIs and UDFs provided with the Greenplum Database Protector can be used to accommodate custom requirements.

For more information about the UDFs, refer to the *APIs, UDFs, and Commands Reference Guide 9.1.0.0*.

6.9 Greenplum Database Protector Example

The following two sections have an example each for encryption and tokenization using Greenplum Database Protector. These two examples are also available with the package in the location, Important points have been mentioned as comments within the code.

6.9.1 Greenplum DB Protector Example – Encryption

```
-----------------------------------------------------------------------
-- DPS User Defined Functions.
-- Copyright (c) 2013 Protegrity USA, Inc. All rights reserved
--
-- NOTE make sure that following is replaced before executing the script:
-- - <data element1> - dataelement used for protecting varchar
-- - <data element2> - dataelement used for protecting integer
-- - <data element3> - dataelement used for protecting date
--
-----------------------------------------------------------------------
-----------------------------------------------------------------------
--
-- SAMPLE1 - Two tables and one view is created as follows:
--
-- Run the sample_tok.sql job to verify protect / unprotect of datatypes
-- VARCHAR, DATE and INTEGER.
--
-----------------------------------------------------------------------
--DROP VIEW SAMPLE1;
--DROP TABLE SAMPLE1_BAK;
--DROP TABLE SAMPLE1_PTY;
-----------------------------------------------------------------------
--
-- SAMPLE1 Base table with no protection
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1 (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL
);
INSERT INTO SAMPLE1 values ('PTY_IVP_FPRTEST_CCN', 'PTY_IVP_FPRTEST_LNAME', 123456789,
987654321, '2013-02-15', '2013-02-15');
RENAME TABLE SAMPLE1 to SAMPLE1_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE1_PTY Same as SAMPLE1 but with protection added for
-- columns, which are encrypted / tokinized when the
-- table is loaded from SAMPLE1_BAK.
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1_PTY (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL
);
INSERT INTO SAMPLE1_PTY("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") SELECT
PTY.INS_VARCHAR(0, '<data element1>', CCN, 0), LNAM,
PTY.INS_INTEGER(0, '<data element2>', RATING, 0), REFN,
CAST(PTY.INS_VARCHAR(0, '<data element3>', CAST(BIRT AS VARCHAR(32)) ,0) AS DATE),
LUPD
FROM SAMPLE1_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE1 Same as SAMPLE1_PTY. But data is decrypted /detokenized
```

```
-- when SAMPLE1 is loaded from SAMPLE1_PTY.
--
-----------------------------------------------------------------------
CREATE VIEW SAMPLE1 ("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") AS SELECT
PTY.SEL_VARCHAR(0, '<data element1>', CCN, 0), LNAM,
PTY.SEL_INTEGER (0, '<data element2>', RATING, 0), REFN,
CAST(PTY.SEL_VARCHAR(0, '<data element3>', CAST(BIRT AS VARCHAR(32)), 0) AS DATE),
LUPD
FROM SAMPLE1_PTY;
```

6.9.2 Greenplum DB Protector Example – Tokenization

```
-----------------------------------------------------------------------
-- DPS User Defined Functions.
-- Copyright (c) 2013 Protegrity USA, Inc. All rights reserved
--
-- NOTE make sure that follwing is replaces before executing script:
-- - <data element1> - dataelement used for protecting varchar
-- - <data element2> - dataelement used for protecting integer
-- - <data element3> - dataelement used for protecting date
--
-----------------------------------------------------------------------
-----------------------------------------------------------------------
--
-- SAMPLE1 - Two tables and one view is created as follows:
--
-- Run the sample_tok.sql job to verify protect / unprotect of datatypes
-- VARCHAR, DATE and INTEGER.
--
-----------------------------------------------------------------------
--DROP VIEW SAMPLE1;
--DROP TABLE SAMPLE1_BAK;
--DROP TABLE SAMPLE1_PTY;
-----------------------------------------------------------------------
--
-- SAMPLE1 Base table with no protection
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1 (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL
);
INSERT INTO SAMPLE1 values ('PTY_IVP_FPRTEST_CCN', 'PTY_IVP_FPRTEST_LNAME', 123456789,
987654321, '2013-02-15', '2013-02-15');
RENAME TABLE SAMPLE1 to SAMPLE1_BAK;
-----------------------------------------------------------------------
--
-- SAMPLE1_PTY Same as SAMPLE1 but with protection added for
-- columns, which are encrypted / tokinized when the
-- table is loaded from SAMPLE1_BAK.
--
-----------------------------------------------------------------------
CREATE TABLE SAMPLE1_PTY (
CCN VARCHAR(32) NOT NULL,
LNAM VARCHAR(32) NOT NULL,
RATING INTEGER NOT NULL,
REFN INTEGER NOT NULL,
BIRT DATE NOT NULL,
LUPD DATE NOT NULL
);
INSERT INTO SAMPLE1_PTY("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") SELECT
PTY.INS_VARCHAR(0, '<data element1>', CCN, 0), LNAM,
PTY.INS_INTEGER(0, '<data element2>', RATING, 0), REFN,
CAST(PTY.INS_VARCHAR(0, '<data element3>', CAST(BIRT AS VARCHAR(32)) ,0) AS DATE),
LUPD
FROM SAMPLE1_BAK;
```

```
--------------------------------------------------------------------------
--
-- SAMPLE1 Same as SAMPLE1_PTY. But data is decrypted /detokenized
-- when SAMPLE1 is loaded from SAMPLE1_PTY.
--
--------------------------------------------------------------------------
CREATE VIEW SAMPLE1 ("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") AS SELECT
PTY.SEL_VARCHAR(0, '<data element1>', CCN, 0), LNAM,
PTY.SEL_INTEGER (0, '<data element2>', RATING, 0), REFN,
CAST(PTY.SEL_VARCHAR(0, '<data element3>', CAST(BIRT AS VARCHAR(32)), 0) AS DATE),
LUPD
FROM SAMPLE1_PTY;
```

Chapter 7

Teradata Database Protector

The Teradata protector has been optimized to work with the fast, parallel, multi-node Teradata systems. This protector is the fastest protection point available on the market for Teradata databases.

This section describes the following:

- Teradata Protector installation prerequisites
- Teradata Protector installation and configuration
- Teradata usage with proxy users by way of query bands and trusted sessions
- Teradata audit logs filtering using MRU settings
- Teradata Proxy node.

7.1 Configuring access to perform queries on Teradata Database from ESA

A user must be granted access and permissions to the certain tables such that they can query the database for members and groups.

The following privilege rights need to be granted to the user defined in the member source configuration on the ESA, who will be performing the queries:

- Select access to DBC.DBASE
- Select access to DBC.ROLEINFO
- Select access to DBC.RoleMembers

There are three basic types of queries performed in Teradata:

- **Retrieving the database users**

```
SELECT DBASE.DatabaseNameI FROM DBC.DBASE DBASE WHERE DBASE.ROWTYPE = 'U' ORDER BY
1;
```

For more information on this topic, refer to section *Fetching Users from Teradata*.

- **Retrieving the database roles/groups**

  ```
  SELECT RoleName,UPPER(GRANTEE) FROM DBC.RoleMembers ORDER BY RoleName;
  ```

- **Retrieving the database users that are members of a role/group**

  ```
  SELECT UPPER(GRANTEE) FROM DBC.RoleMembers ORDER BY GRANTEE;
  ```

7.2 Fetching Users from Teradata

The Database Protector supports user names that are up to 255 characters in length. However, as the Teradata platform, version 14.10 and higher, supports user name lengths of 128 characters only, the user name is thus limited to the value supported by the Teradata platform.

7.3 Teradata Query Bands and Trusted Sessions

When a middle-tier application is used together with the Teradata database, it typically logs on to the database as a permanent database user (application user) and establishes a connection pool. End-users that access the database through the middle-tier application are given all authorized database privileges and are audited based on that single application user.

For sites that require users to be individually identified, authorized, and audited, the middle-tier application can be configured to offer trusted sessions. Application users that access the database through a trusted session must be set up as proxy users and assigned one or more database roles, which determine their access rights in the database. When a proxy user requests database access, the application forwards the user identity and applicable role information to the database.

For more information about Teradata trusted sessions, refer to the Teradata document *Teradata Database Security Administration*.

The system uses a proxy user if the query band contains the reserved name PROXYUSER. In order for the proxy user to access sensitive data, UDFs and UDTs need to know the requestor of the data. They obtain this information from the query band parameters.

For more information about query bands, refer to the Teradata document *Teradata Database SQL Data Definition Language*.

If a proxy user is found among the query band parameters, then it is used in the authorization process instead of the regular database user (which could be a different user). This means that only the proxy user’s permissions apply. This is similar to how the Teradata permissions work for trusted sessions. The database permissions for the proxy user are used, and not the application user's permissions.

Before such a user can access the database, a Grant Connect through Access right should be given by the database administrator to the user. The following example provides the query to ensure that the user 'goldtown' can connect through.

The example below shows a database session where the PROXYUSER query band parameter is used. In this example, a switch is made from User1 (which is in the policy) to another user ('goldtown').

To run this example, the database administrator must first Grant Connect access to the user, 'goldtown'.

```
GRANT CONNECT THROUGH USER1
TO PERMANENT goldtown
WITHOUT ROLE;
```

The user *goldtown* can now access the database.

> **Note:** The UDF *getqueryband* is provided by Teradata.

```
select getqueryband();
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
getqueryband()
--------------------------------------------------------------------------------
select pty_varcharlatinenc('abcd','AES',123,0,0);
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
pty_varcharlatinenc('abcd','AES',123,0,0)
--------------------------------------------------------------------------------
E3AE49B5C44E4CE64CC7AB3A20F82325
SET QUERY_BAND='PROXYUSER=goldtown;' FOR SESSION;
*** Set QUERY_BAND accepted.
*** Total elapsed time was 1 second.
select getqueryband();
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
getqueryband()
--------------------------------------------------------------------------------
=S> PROXYUSER=goldtown;
select pty_varcharlatinenc('abcd','AES',123,0,0);
*** Failure 7504 in UDF/XSP/UDM SYSLIB.pty_varcharlatinenc: SQLSTATE U0001:
No such user
Statement# 1, Info =0
*** Total elapsed time was 1 second.
AUDIT TRACE:
Thu Dec 30 01:21:53.530 2010 goldtown AES 0 1 0 0 Insert, unknown user dbp 1
SET QUERY_BAND=NONE FOR SESSION;
*** Set QUERY_BAND accepted.
*** Total elapsed time was 1 second.
select pty_varcharlatinenc('abcd','AES',123,0,0);
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
pty_varcharlatinenc('abcd','AES',123,0,0)
--------------------------------------------------------------------------------
E3AE49B5C44E4CE64CC7AB3A20F82325
```

7.4 Teradata Audit Logs

The many nodes of Teradata generate an abundance of logs. The Teradata Database Protector and the Teradata Database filter logs based on certain parameters. The following two sections explain how the audit logs are filtered by the Protector and by the Database.

7.4.1 Audit Log Filtering by Protector

The Teradata database, in particular multi-node Teradata database produces duplicate logs. To filter out the Teradata logs, MRU settings can be applied.

The MRU helps to reduce the number of logs in several steps. Duplicates are removed on the node level and on a system level. This reduces the amount of data sent from the PEP server as well as the amount of data stored in the log repository (at the Log Server or Protegrity Data Management Server).

Each log event has some information from the operation that generated the event. In Teradata those key values are Session Number, Request Number and Database User ID. These values are read from the database at the time of UDF execution. The MRU uses those values to decide whether a record is a duplicate or not, given some rules set in the configuration of the MRU.

UDFs Filtering

The UDF does some filtering based on the session number and request number. It keeps a list of fixed size in memory. This list holds session numbers request numbers and data elements for which an audit record has been generated. It is a simple circular buffer, so once fully populated it starts overwriting from the beginning.

For one single query there is one audit record generated for each AMP Worker Task (AWT) utilized, regardless of number of rows accessed. This is because each UDF runs in its own space. The number of AMPs depends on Teradata system configuration and size.

The number of audit records generated also depends on how many nodes and AMPs were involved in the query. In some cases just one (or a few nodes) are involved in a query. Typically for tactical queries, that pulls back just a few records.

The UDF filtering reduces the number of audit records generated from number of rows encrypted/ decrypted to number of AMPs.

PEP Server Filtering

The PEP server reads audit records generated by the UDF from shared memory. This means that initially the PEP server can have, for example, 10 identical audit records.

These identical records are then further filtered by the MRU in the PEP server. The filtering takes place when the records have been read from shared memory and before stored in cache and sent to the Log Server. Duplicate records are discarded.

What make the records identical to the PEP server are session number, request number, time of arrival (time when the record was fetched) and Data Element. So audit records that belong to a database transaction must arrive at the PEP server within a specified time to be considered identical.

The Time Delta is configurable in the PEP server. For the PEP server this value is typically rather small (by default, 60 seconds). The PEP server filtering reduces the number of audit records from number of AMPs invoked to just 1.

In case of multiple protected columns there needs to be a single event for each data element accessed.

In cases where there are multiple columns protected with a single Data Element – the MRU provides a single event for ALL the columns. In cases where there are multiple columns protected each with different data element – the MRU provides a single event for each column.

Log Server / Data Management Server Filtering

The Log Server and Data Management Server use the same MRU component as the PEP server. This filtering reduces duplicate audit records that come from multiple PEP servers in a multi-node Teradata system. The filtering is done in a similar way as in the PEP server with some differences:

- The MRU also considers the Database User ID since the session number and request number cannot be considered unique when logs are coming from multiple Teradata systems.
- The Time Delta must be set higher than for the PEP Server. The audit records are transferred from the PEP servers to the Log Server/Data Management Server and this can create some time lag, especially in a large scale Teradata system. To account for this the time delta should be set quite high (up to two hours).

In the Log Server/Data Management Server the MRU is used to check for duplicates before any audit records are stored in the log repository.

7.4.2 Audit Log Filtering by Teradata Database

Typically any Database generates error and audit logs for most activities, which include protection, unprotection, or re-protection operations initiated by a Database Protector. In the case of Teradata Database, the number of logs multiply depending on the number of nodes. To manage this number, built-in checks have been added within Teradata Database architecture.

Therefore, Teradata Database which would otherwise produce logs for every activity of deterministic Protegrity Teradata Protector UDFs with constant parameters, generates logs and messages only for the first two executions of the UDF. After that Teradata optimizes calls to the UDF and returns a cached result. Since the UDF is not invoked, no audit record is generated. Hence no further logs are available for subsequent execution of the same UDF.

7.4.3 Configuring MRU Settings

You need to perform the required MRU configuration settings in the *pepserver.cfg* file and the *DMS.cfg* file to filter the audit logs.

The following snippet is the part of the *pepserver.cfg* file that you need to configure.

```
# ------------------------
# Log Server configuration
# ------------------------
# Teradata Log MRU List configuration parameters:
# Flag to set the override mode, when the list is at full capacity and no
# entries can be timed out. If the flag is set then the oldest entry be reused
# teradatamruoverridemode = ON
# Maximum MRU list size. To set this entry accurately one would need to
# estimate how many distinctive DB requests can happen in a specific time
# interval.
# Max value is 200,000. Default value is 200,000
# 0 will disable MRU.
# teradatamrumaxlistsize = 200000
```

The following snippet is the part of the *DMS.cfg* file that you need to configure.

```
<!-- TeraData MRU implementation -->
<TeraDataMRU enabled="1">
<MaxListSize>2000000</MaxListSize>
<TimeDelta>3600</TimeDelta>
</TeraDataMRU>
```

Configuring Time Delta

The Time Delta is expressed in seconds and is used to determine if a record added to the MRU is a duplicate or not.

When a record is added to the MRU it is tagged with a timestamp of when it was added. All time differences are calculated as current time minus the entry timestamp.

When subsequent entries are added:

- If an existing key (Session Number and Request Number) is found and the time difference is less than time delta, then the entry timestamp is updated.
- If an existing key is found but the time difference is larger than time, then the MRU does not consider it as a duplicate. The timestamp of the entry gets updated and the MRU indicates that this was a new audit record (i.e. NOT a duplicate).

If a new entry is about to be added to the MRU and the list is full, then the oldest entry in the list is a candidate to overwrite:

- If the time difference for the oldest entry is less than time, then it is reused, but only if “Override Mode” is set to TRUE.
- If the time difference for the oldest entry is larger than time delta, then it is reused.

Configuring Maximum List Size

The Max List Size is the maximum number of entries in the MRU list. It can also be used to disable the MRU by setting the value to zero.

> **Note:** For most of the configurations, it is recommended to change the default value to 20000 to increase performance.

If the max limit is reached, then the oldest entry is reused according to the override mode, and time delta settings.

The MRU has an API cleaning all entries in the list that have a time difference larger than time delta.

Configuring Override Mode

The Override Mode determines what to do when the MRU reaches maximum list size. If override mode is ON, then the oldest entry can be reused even if the time difference is less than time delta.

7.5 Teradata User Defined Type (UDT)

A user-defined type (UDT) is a named data type that is created in the database by the user. The UDTs have the ability to interact with other types by way of their transforms, orderings, casts, and instance methods.

The implementation for distinct UDTs enables a developer to drop any auto-generated component and define their own. This capability allows for maximum flexibility if the default behavior of the UDT is not good enough for a specific application.

Teradata automatically creates the To-SQL and From-SQL transform, the ordering, and the necessary casts for a distinct UDT once the CREATE TYPE statement is issued.

On the Database Protector installation, the */pep* directory is created with the following files:

- *generate_udt_scripts.sh* is an executable file that generates UDT scripts
- *pepteradata12udt.plm* is a library that contains protect and unprotect functions for UDP usage.

Generate_udt_scripts.sh generates UDT scripts using the following arguments:

```
labsxtd13-64:/opt/protegrity/defiance_dps/pep # ./generate_udt_scripts.sh
Defiance DPS UDT Scripts for Teradata
Usage: generate_udt_scripts udtname dataelement scid dbtype
udtname : UDT Name
dataelement: Data Element
scid : Security Coordinate ID
dbtype : Database data type, must be one of: bigint,date,float,integer,varchar
```

The following are some limitations for the UDT arguments:

- Udtname – any applicable name
- Dataelement – DE deployed to the PEP server
- Scid – applicable security coordinate (0 by default)
- Dbtype – one of bigint, date, float, integer, varchar.

7.5.1 Creating a UDT

➤ To create a UDT:

1. Execute *generate_udt_scripts.sh* that generates two scripts *create_<UDTNAME>.sql* and *drop_<UDTNAME>.sql*. They are customized for BTEQ so that basic error handling can be done.

   ```
   ./generate_udt_scripts.sh UDT_VARCHAR AES128 0 varchar
   ```

   It generates two scripts for creating a UDT - *create_UDT_VARCHAR.sql* and for dropping a UDT - *drop_UDT_VARCHAR.sql*.

   > **Note:** You are allowed to use data elements names in upper-case only.

2. Execute the *create_<UDTNAME>.sql* using BTEQ or any other client tool (such as Teradata SQL Assistant).

   Script creates UDT (under SYSUDTLIB database) and three UDFs (actually two for encryption and decryption and one Teradata support function).

   > **Note:** If you use nonzero *Communication ID*, then you should modify the *Communication ID* in script through all your *create_UDT* script.
   >
   > Note that you can create only one UDT for each data type. Creation of another UDT that uses the basic data type which is already used by any existing UDT results in a linker error.

3. Make some GRANTs using Teradata Administrator or manually executing SQL statement:

   a. Grant UDTUSAGE access (with GRANT option) to the just created UDT to public (including your DB).

      ```
      grant udtusage on sysudtlib to public with grant option
      ```

   b. Grant execute function for all UDT functions to public (with GRANT option).

4. Create a table specifying column(s) type as UDT.

   ```
   CREATE TABLE WITH_UDT (a INTEGER NOT NULL,b UDT_VARCHAR(50) NOT NULL, PRIMARY KEY(c1_pk))
   ```

   When the column(s) type is specified as UDT, the table can be used as it is, while protection/unprotection functions are performed “inside” the UDT.

7.6 Teradata Proxy Node

If you are dealing with multi-node Teradata environment, then not all nodes can be reached from the ESA, and not all nodes can reach back to the ESA. For such cases, it is required to use a Teradata Proxy node that provides an indirect means of communicating with the ESA.

Effective from Release 7.0, proxy daemon is provided for Teradata multi-node database solutions called, Protector Proxy.

> **Note:** For information about installing the Protector Proxy, refer to section *8.3 Protector Proxy Installation* in the *Installation Guide 9.1.0.0*.

7.7 Teradata User Defined Functions

For more information about Teradata user-defined functions, refer to section *Teradata User Defined Functions* in the *Protegrity APIs, UDFs, and Commands Reference Guide Release 9.1.0.0*.

7.8 Using Unicode Tokens with Teradata

If you are utilizing the Unicode PLM with the Teradata database protector, then ensure that the prerequisites mentioned in this section are met.

7.8.1 Prerequisites

Ensure that the following prerequisites are met before using Unicode tokens with Teradata:

- Set the the database name to the user, that was created using the Unicode character set in the Unicode script files, that are created after installation of the Teradata database protector.
- Set the session for the *Unicode* user before using the Unicode scripts.

  For more information about setting the session for the Unicode users, refer to section *Setting the Session for Unicode Users*.

- Set the character set as *Latin* for the *Unicode* user before using the Unicode scripts.

  For more information about setting the character set for the Unicode users, refer to section *Setting the Character Set for Unicode Users*.

If you need to uninstall the Unicode UDFs from the Teradata database protector, then run the *dropobjects.sql* file from the main node.

7.8.2 Setting the Session for Unicode Users

If you are using the Teradata database protector with a *Unicode* user, then you need to set the session for the *Unicode* user.

Before you begin
To set the Session:

1. Login to the Teradata database using the credentials for the *Unicode* user.
2. Run the following command to set the session for the *Unicode* user.
   ```
   .set session charset 'utf8';
   ```
3. Press ENTER.
   The Teradata database session is set for the *Unicode* user.

7.8.3 Setting the Character Set for Unicode Users

If you are using the Teradata database protector with a *Unicode* user, then set the character set as *Latin* for the *Unicode* user.

The following snippets displays the *PTY_GETDBSINFO()* and *PTY_CHECKSELACCESS()* functions in the *createobjects_u.sql* file, present in the `/opt/protegrity/defiance_dps/pep/sqlscripts/teradata` directory.

```
BT;
REPLACE FUNCTION PTY_GETDBSINFO()
RETURNS VARCHAR(500) CHARACTER SET LATIN
SPECIFIC pty_getdbsinfo
LANGUAGE C
NO SQL
PARAMETER STYLE TD_GENERAL
EXTERNAL NAME 'SP!/opt/protegrity/defiance_dps/pep/pepteradata1510u.plm';
ET;
```

```
BT;
REPLACE FUNCTION PTY_CHECKSELACCESS (DataElement1 VARCHAR(100) CHARACTER SET LATIN,
DataElement2 VARCHAR(100) CHARACTER SET LATIN,
DataElement3 VARCHAR(100) CHARACTER SET LATIN,
ResultLen INTEGER,
CommunicationID INTEGER )
RETURNS VARCHAR(3) CHARACTER SET LATIN
SPECIFIC pty_checkselaccess
LANGUAGE C
NO SQL
PARAMETER STYLE SQL
```

```
DETERMINISTIC
CALLED ON NULL INPUT
EXTERNAL NAME 'SP!/opt/protegrity/defiance_dps/pep/pepteradata1510u.plm';
ET;
```

> **Note:** Ensure that you append the text displayed in *red* color in the *createobjects_u.sql* file to set the character set as *Latin* for the *Unicode* user.

7.9 Teradata User Defined Functions and Procedures

The APIs and UDFs provided with the Teradata Protector can be used to accommodate custom requirements.

For more information about the UDFs, refer to section *Teradata User Defined Functions* in the *APIs, UDFs, and Commands Reference Guide 9.1.0.0*.

7.10 Teradata Database Protector Example

In this section, two examples of usage of the Protector is provided, for encryption and for tokenization. Go through the same and replace the values as required. These examples are also included in the database package.

7.10.1 Encryption Example

```
------------------------------------------------------------------------
-- Protegrity User Defined Functions sample script.
--
-- NOTE: Please change the following 'tags' before executing the script:
--       - REPLACE_DATABASE- Specify the testdatabase.
--       - <data element1> - dataelement used for protecting varchar
--       - <data element2> - dataelement used for protecting integer
--       - <data element3> - dataelement used for protecting date
--
-- This script should be run in BTEQ.
--
-- Copyright (c) 2015 Protegrity USA, Inc. All rights reserved
--
------------------------------------------------------------------------
DATABASE REPLACE_DATABASE;
.IF ERRORCODE != 0 THEN .QUIT 99

BT;
  DROP TABLE SAMPLE1_BAK;
ET;

BT;
  DROP TABLE SAMPLE1_PTY;
ET;

DROP VIEW SAMPLE1;

------------------------------------------------------------------------
--
-- SAMPLE1: Table with no protection. Contains sample data.
--
------------------------------------------------------------------------
BT;
  CREATE MULTISET TABLE SAMPLE1 (
    CCN     VARCHAR(32)               NOT NULL,
    LNAM    VARCHAR(32)               NOT NULL,
    RATING  INTEGER                   NOT NULL,
    REFN    INTEGER                   NOT NULL,
    BIRT    DATE FORMAT 'YYYY/MM/DD'  NOT NULL,
    LUPD    DATE FORMAT 'YYYY/MM/DD'  NOT NULL
```

```
  );
ET;

BT;
  INSERT INTO SAMPLE1 ('4000567834561233', 'PTY_IVP_FPRTEST_LNAME', 123456789, 987654321,
CAST( '2013/02/15' AS DATE ), CAST( '2013/02/15' AS DATE ));
ET;

BT;
RENAME TABLE SAMPLE1 to SAMPLE1_BAK;
ET;

--------------------------------------------------------------------------
--
-- SAMPLE1_PTY: This table is similar to SAMPLE1 will contain protected data.
--              The 'LNAM','REFN', and 'LUPD' columns now are of type VARBYTE.
--              Data is migrated from the 'SAMPLE1_BAK' table using UDF calls.
--
--------------------------------------------------------------------------
BT;
  CREATE MULTISET TABLE SAMPLE1_PTY (
    CCN     VARCHAR(32)               NOT NULL,
    LNAM    VARBYTE(48)               NOT NULL,
    RATING  INTEGER                   NOT NULL,
    REFN    VARBYTE(16)               NOT NULL,
    BIRT    DATE FORMAT 'YYYY/MM/DD'  NOT NULL,
    LUPD    VARBYTE(16)               NOT NULL
  );
ET;

BT;
  INSERT INTO SAMPLE1_PTY("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") SELECT
    "CCN",
    TESTDB.PTY_VARCHARLATINENC("LNAM",'<data element1>',48,0,0),
    "RATING",
    TESTDB.PTY_INTEGERENC("REFN",'<data element2>',16,0,0),
    "BIRT",
    TESTDB.PTY_DATEENC("LUPD",'<data element3>',16,0,0)
  FROM SAMPLE1_BAK;
ET;

--------------------------------------------------------------------------
--
--  SAMPLE1: This is a view that shows how data is unprotected using the UDFs.
--           Data is selected from the 'SAMPLE1_PTY' table.
--           The name of this view is the same as the original table
--
--------------------------------------------------------------------------
BT;
  CREATE VIEW SAMPLE1 ("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") AS SELECT
    "CCN",
    CAST(TESTDB.PTY_VARCHARLATINDEC("LNAM",'<data element1>',32,0,0) AS VARCHAR(32)),
    "RATING",
    CAST(TESTDB.PTY_INTEGERDEC("REFN",'<data element2>',0,0) AS INTEGER),
    "BIRT"
    ,
    CAST(TESTDB.PTY_DATEDEC("LUPD",'<data element3>',0,0) AS DATE)
  FROM SAMPLE1_PTY;
ET;
```

7.10.2 Tokenization Example

```
--------------------------------------------------------------------------
-- DPS User Defined Functions.
-- Copyright (c) 2015 Protegrity USA, Inc. All rights reserved
--
-- This script should be run in BTEQ
--
-- NOTE: Please change the following 'tags' before executing the script:
--       - TESTDB - database where the protegruty UFS's are installed
```

```
--      - REPLACEDB - Database where you have testdata
--      - <data element1> - dataelement used for protecting varchar
--      - <data element2> - dataelement used for protecting integer
--      - <data element3> - dataelement used for protecting date
--
------------------------------------------------------------------------
DATABASE REPLACE_DATABASE;
.IF ERRORCODE != 0 THEN .QUIT 99

------------------------------------------------------------------------
--
-- SAMPLE - Two tables and one view is created as follows:
--
-- Run the sample_tok.sql job to verify protect / unprotect of datatypes
-- VARCHAR, DATE and INTEGER.
--
------------------------------------------------------------------------
BT;
  DROP TABLE SAMPLE1_BAK;
ET;

BT;
  DROP TABLE SAMPLE1_PTY;
ET;

DROP VIEW SAMPLE1;

------------------------------------------------------------------------
--
-- SAMPLE1  Base table with no protection
--
------------------------------------------------------------------------
BT;
  CREATE MULTISET TABLE SAMPLE1 (
    CCN     VARCHAR(32)               NOT NULL,
    LNAM    VARCHAR(32)               NOT NULL,
    RATING  INTEGER                   NOT NULL,
    REFN    INTEGER                   NOT NULL,
    BIRT    DATE FORMAT 'yyyy-mm-dd'  NOT NULL,
    LUPD    DATE FORMAT 'yyyy-mm-dd'  NOT NULL
  );
ET;

BT;
  INSERT INTO SAMPLE1 ('4000567834561233', 'PTY_IVP_FPRTEST_LNAME', 123456789, 987654321,
CAST( '2013/02/15' AS DATE ), CAST( '2013/02/15' AS DATE ));
ET;

BT;
RENAME TABLE SAMPLE1 to SAMPLE1_BAK;
ET;

------------------------------------------------------------------------
--
-- SAMPLE1_PTY Same as SAMPLE1 but with protection added fo
--             columns, which are encrypted / tokinized when the
--             table is loaded from SAMPLE1_BAK.
--
------------------------------------------------------------------------
BT;
  CREATE MULTISET TABLE SAMPLE1_PTY (
    CCN     VARCHAR(32)               NOT NULL,
    LNAM    VARCHAR(32)               NOT NULL,
    RATING  INTEGER                   NOT NULL,
    REFN    INTEGER                   NOT NULL,
    BIRT    DATE FORMAT 'yyyy-mm-dd'  NOT NULL,
    LUPD    DATE FORMAT 'yyyy-mm-dd'  NOT NULL
  );
ET;

BT;
  INSERT INTO SAMPLE1_PTY("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") SELECT
    TESTDB.PTY_VARCHARLATININS("CCN",'<data element1>',32,0,0),
```

```
    "LNAM",
    TESTDB.PTY_INTEGERINS("RATING",'<data element2>',32,0,0),
    "REFN",
    CAST(TESTDB.PTY_VARCHARLATININS(CAST("BIRT" AS VARCHAR(32)),'<data element3>',32,0,0) AS
DATE),
    "LUPD"
  FROM SAMPLE1_BAK;
ET;

------------------------------------------------------------------------
--
--  SAMPLE1 Same as SAMPLE1_PTY. But data is decrypted /detokenized
--              when SAMPLE1 is loaded from SAMPLE1_PTY.
--
------------------------------------------------------------------------
BT;
  CREATE VIEW SAMPLE1 ("CCN", "LNAM", "RATING", "REFN", "BIRT", "LUPD") AS SELECT
    TESTDB.PTY_VARCHARLATINSEL("CCN",'<data element1>',32,0,0),
    "LNAM",
    TESTDB.PTY_INTEGERSEL ("RATING",'<data element2>',0,0),
    "REFN",
    CAST(TESTDB.PTY_VARCHARLATINSEL(CAST("BIRT" AS VARCHAR(32)),'<data element3>',32,0,0) AS
DATE),
    "LUPD"
  FROM SAMPLE1_PTY;
ET;
```

Chapter 8

DB2 Open Systems Database Protector

The DB2 Open Systems Database Protector secures access to sensitive data on DB2 Open Systems.

This section describes the tasks that you need to perform with these objects to set up and use the DB2 Open Systems Database Protector:

- UDFs installation
- Data Security Platform activation.

8.1 Set Up Requirements

Supported Versions

Linux Version: AIX 7.1 and RHEL 7

DB2: 9.5 or later

Environment Specifications

Before setting up the DB2 Database Protector, ensure that your configuration meets the minimum requirements. The following table shows the environment requirements.

Table 8-1: Environment Requirements

Configuration	Free Disk Space on UNIX
PEP server	50 MB
User Defined Functions (UDFs) and procedures	10 MB
RAM	4 GB

> **Note:** For systems under heavy load with auditing turned on, the space required for the PEP server (20 MB per node) can be higher.

8.2 Installing and Uninstalling the DB2 Database Protector

This section provides information about setting up, installing, and uninstalling the DB2 Database Protector.

The following table provides the order of installation for the DB2 Database Protector.

Table 8-2: DB2 Database Protector - Order of Installation

Order of installation	Description	Reference
1.	Install the ESA	*Installing ESA Appliance*
2.	Verify the Prerequisites	*Verifying the Prerequisites for Installing the DB2 Database Protector*
3.	Installation Preparation	*Creating the Directory to Install the DB2 Database Protector*
4.	Install the PEP server and the DB2 Database Protector	*Installing the PEP server and the DB2 Database Protector*

8.2.1 Verifying the Prerequisites for Installing the DB2 Database Protector

Ensure that the following prerequisites are met:

- The DB2 Database is installed and configured.
- Download and save the DB2 Database Protector, *DatabaseProtector_<OS>-<arch>_<DB2 distribution>-64_<version>.tgz*, made available by Protegrity.
- Even if it is not mandatory, take a backup of the databases where the DB2 Database Protector and UDFs would be installed.
- Access as the *root* user to the server, should be available to you.
- Access as the *database* user for the DB2 database, should be available to you.

8.2.2 Creating the Directory to Install the DB2 Database Protector

The `/opt/protegrity` directory is the default directory to install the DB2 Database Protector. If the default directory is not present, then create the directory and grant *755* permissions.

Run the following command to ensure that the UDFs are working and access is available to all the files and directories.

```
# chmod 755 /opt/protegrity/
```

8.2.3 Installing the PEP Server

To install the PEP server:

1. Login to the server as the *root* user.
2. Copy the DB2 Database Protector package, *DatabaseProtector_<OS>-<arch>_<DB2 distribution>-64_<version>.tgz*, to the `/opt/protegrity` directory.
3. Navigate to the `/opt/Protegrity` directory and unpack the installation package.

   For DB2 on AIX platform, run the following commands.

   ```
   # gunzip DatabaseProtector_<OS>-<arch>_<DB2 distribution>-64_<version>.tgz

   # tar -xvf DatabaseProtector_<OS>-<arch>_<DB2 distribution>-64_<version>.tar
   ```

   For example,

   ```
   # gunzip DatabaseProtector_AIX-7.2-64_PPC-64_DB2-11.1-64_7.1.0.xxx.tgz

   # tar -xvf DatabaseProtector_AIX-7.2-64_PPC-64_DB2-11.1-64_7.1.0.xxx.tar
   ```

For DB2 on Linux platform, run the following command.

```
# tar zxf DatabaseProtector_<OS>-<arch>_<DB2 distribution>-64_<version>.tgz
```

For example,

```
# tar zxf DatabaseProtector_RHEL-7-64_x86-64_DB2-11.5-64_7.1.0.xxx.tgz
```

The following files are extracted:

- *PepServerSetup_<OS>_<version>.sh*
- *PepDB2Setup_<OS>_<version>.sh*
- *U.S.Patent.No.6,321,201.Legend.txt*

4. Run the PEP server installer using the following command. Follow the screen prompts that display during the installation process.

   ```
   # ./PepServerSetup_Linux_x64_<version>.sh
   ```

   The following message appears on successful installation.

   ```
   Certificates successfully downloaded and stored in /opt/protegrity/defiance_dps/data.
   Protegrity PepServer installed in /opt/protegrity/defiance_dps.
   ```

> **Caution:** Ensure that the ESA is installed and running with the *HubController* service state as *Running*, to enable downloading of the certificates automatically.

8.2.3.1 Checking the Status of PEP Server

➤ To find the status of the PEP server:

1. Run the following command to find the status of the PEP server.

   ```
   # sh /opt/protegrity/defiance_dps/bin/pepsrvctrl status
   ```

2. If the PEP server is not running, then start the PEP server using the following command.

   ```
   # sh /opt/protegrity/defiance_dps/bin/pepsrvctrl start
   ```

8.2.4 Installing the UDFs for DB2

This section describes how to create UDFs for a DB2 database.

➤ To create UDFs for DB2:

1. Run the DB2 Database Protector installer using the following command to install the PEP UDFs. Follow the screen prompts that appear during the installation process.

   ```
   # ./PepDB2Setup_Linux_x64_<version>.sh
   ```

   The following message appears on successful installation.

   ```
   PEP installed in /opt/protegrity/defiance_dps.
   ```

2. Navigate to the */opt/protegrity/defiance_dps/pep/sqlscripts/db2* directory and open the script *createobjects.sql*.
3. In the *createobjects.sql* script, modify the *CONNECT TO REPLACE_DATABASE;* parameter by changing the *REPLACE_DATABASE* parameter to the database where you want to install the UDFs.
4. Login to DB2 using the database user having the required permissions.
   For example: *USER1*
   ```
   # su - user1
   ```
5. Navigate to the */opt/protegrity/defiance_dps/pep/sqlscripts/db2* directory.
6. Run the following command to install the UDFs.
   ```
   # db2 -tfcreateobjects.sql
   ```

For more information about the DB2 Database Protector UDFs, refer to section *DB2 Open Systems User Defined Functions* in the *APIs, UDFs, and Commands Reference Guide 9.1.0.0*.

8.2.5 Verifying the Installation of UDFs for DB2

This section describes the steps to check that all UDFs have been installed.

➤ To verify the installation of UDFs for DB2:

1. Login to DB2 using the database user having the required permissions.
   For example: *USER1*
   ```
   # su - user1
   ```
2. Connect to the database where the UDFs are installed.
3. Run either of the following DB2 Database UDFs to verify that the installation is successful.
   - ```
     SELECT pty.whoami() FROM SYSIBM.SYSDUMMY1;
     ```
     Name of the user is displayed.
   - ```
     SELECT pty.getversion() FROM SYSIBM.SYSDUMMY1;
     ```
     Version number of the PEP server is displayed.

8.2.6 Uninstalling the DB2 Database Protector

To uninstall the DB2 Database Protector, the UDFs and the PEP server need to be uninstalled and removed. You should be logged in as the *root* user.

8.2.6.1 Uninstalling the UDFs

➤ To Uninstall the UDFs:

1. Connect as a privileged user and navigate to the */opt/protegrity/defiance_dps/pep/sqlscripts/db2* directory.
2. Open the script *dropobjects.sql*.

3. In the *dropobjects.sql* script, modify the *CONNECT TO REPLACE_DATABASE;* parameter by changing the *REPLACE_DATABASE* parameter to the database where the UDFs are installed.
4. Login to DB2 using the database user having the required permissions.
   For example: *USER1*
   ```
   # su - user1
   ```
5. Navigate to the `/opt/protegrity/defiance_dps/pep/sqlscripts/db2` directory.
6. Run the following command to uninstall the UDFs.
   ```
   # db2 -tfdropobjects.sql
   ```
7. Delete the `/opt/protegrity/defiance_dps/pep/` directory.

8.2.6.2 Uninstalling the PEP Server

To uninstall the PEP Server:

1. Navigate to the `/opt/protegrity/defiance_dps/bin` directory.
2. Run the following command to stop the PEP server.
   ```
   # ./pepsrvctrl stop all
   ```
3. Delete the `/opt/protegrity/defiance_dps` directory including the sub-directories.

Chapter 9

Trino Protector

In an enterprise setup, data exists in multiple disparate locations. Traditionally, data has always existed in relational databases, such as, MS SQL, Oracle, Enterprise Data Warehouse (EDW), and so on. If the data resides in these structured environments, then it is difficult to analyze details, such as, market trends, customer sentiments, and so on, that help drive businesses.

Many organizations are moving towards the Hadoop ecosystem to store and manipulate data. They are moving towards scalable environments that not only store data, but also provide them an ability to use that data to predict user behavior and build strategies that create additional business value.

9.1 What is Trino Protector

Trino, a distributed SQL engine, lets you leverage its ability to query across multiple and disparate data sources to generate reliable insights.

In the Big Data environment, the Trino protector is used together with Hive to process queries and provide data warehousing capabilities.

> **Note:** The Trino protector currently supports the Hive environment only.

In this release, based on the Trino client query, the Trino Protector performs data security operations on sensitive data stored in HDFS. The user permissions and roles defined in the ESA data security policy define whether the user can view the protected data or clear data.

9.2 Trino Protector Architecture

This section provides a detailed understanding of the Trino Protector architecture.

The Trino Protector is installed on each node of the Hadoop cluster. The following image illustrates how the data security operation is performed when a data analyst queries HDFS using the Trino CLI to gain insight for data processing and analysis.

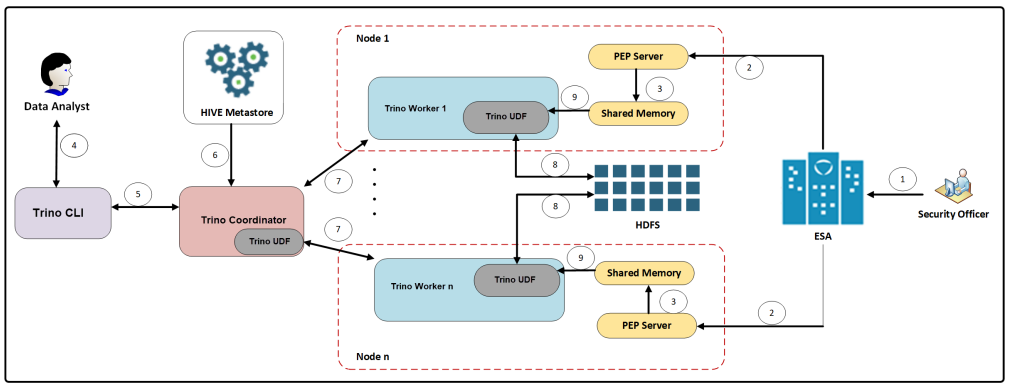

Figure 9-1: Trino Protector architecture

1. As part of the organizational setup, the ESA is installed as the central policy management unit. The Security officer is responsible for creating relevant data policies, assigning roles and permissions, and defining protection methods that the organization wants to deploy.
2. The Trino Protector is installed on each node of the Hadoop cluster and includes the PEP server that is responsible for pulling policies and other metadata from the ESA.
3. The PEP server pushes the policy metadata to the Shared memory. Whenever the Trino CLI client queries HDFS, the data is processed by the Trino Protector based on the policy metadata that resides in the Shared Memory.
4. Assuming the organization wants to analyze the data that resides in the HDFS for predicting user behavior and build better strategies, a Data Analyst queries using the Trino CLI. The query includes the Trino Protector UDFs to perform data security operations.
5. The query is forwarded to the Trino Co-ordinator node, which is responsible for parsing, analyzing, and further distributing it to the Trino Worker nodes.
6. The Trino Co-ordinator interacts with the Hive Metastore to retrieve metadata that is required to access the HDFS and plan distribution to the Trino Worker nodes.
7. After retrieving the metadata and analyzing the distribution pattern, the Trino Co-ordinator node distributes the query among the Trino Worker nodes.
8. The Trino Worker nodes fetch the data from HDFS based on the query.
9. The Trino Protector UDFs installed on each node enforce the data security policy and other metadata, such as, user roles and permissions, fetched from the shared memory on the data. Based on the policy information, data either appears protected or in clear to the Data Analyst. After processing the data, the Trino Co-ordinator node is responsible for collating the data and return the result of the query to the Data Analyst.

9.3 Set Up Requirements

Supported Versions

Linux Version: RHEL 6, RHEL 7, SLES 10, SLES 11

Trino: 351 or later

Environment Specifications

Before setting up the Trino Protector, ensure that your configuration meets the minimum requirements. The following table shows the environment requirements.

Table 9-1: Environment Requirements

Configuration	Free Disk Space on Linux
PEP server	10 MB per node

Configuration	Free Disk Space on Linux
PEP server repository	5 MB
User defined functions (UDFs) and procedures	10 MB

> **Note:** For systems under heavy load with auditing turned on, the space required for the PEP server (20 MB/node) can be higher.

9.4 Installing the Trino Protector

The process of installing the Trino Protector involves running the configurator script and running the installation script to install Trino on all the nodes in the cluster.

> **Note:** For more information about installing the Trino Protector, refer to the section *13.7.7 Installing and Uninstalling the Trino Protector* in the *Protegrity Installation Guide 9.1.0.0.*.

Appendix A

Supported Database Protectors Matrix

The below table lists the database protectors with the supported database version and platform details:

Protector	Supported Database Version	Supported Platforms
MS SQL Server Database Protector	MS SQL Server 2008	Windows 2003 x32
		Windows 2003 x64
		Windows 2008 x32
		Windows 2008 x32
		Windows 2012 x64
	MS SQL Server 2012	Windows 2008 x64
		Windows 2012 x64
	MS SQL Server 2014	Windows 2012 x64
Oracle Database Protector	Oracle 12c R1	AIX 6.1
		AIX 7.1
		OEL 6
		RHEL 6
		SLES 11
	Oracle 12c R2	AIX 6.1
		AIX 7.1
		AIX 7.2
		OEL 6
		OEL 7
		RHEL 6
		RHEL 7
		SLES 11
		SLES 12
		SLES 15
	Oracle 18c	AIX 7.1
		AIX 7.2
		OEL 6
		OEL 7

Protector	Supported Database Version	Supported Platforms
		RHEL 6
		RHEL 7
		SLES 12
	Oracle 19c	AIX 7.1
		AIX 7.2
		OEL 6
		OEL 7
		OEL 8
		RHEL 6
		RHEL 7
		RHEL 8
		SLES 12
		SLES 15
Teradata Database Protector	Teradata 14.10	SLES 10
		SLES 11
	Teradata 15.0	SLES 10
		SLES 11
	Teradata 15.10	SLES 10
		SLES 11
	Teradata 16.10	SLES 11
	Teradata 16.20	SLES 11
	Teradata 17.00	SLES 11
Greenplum Database Protector	Greenplum 4.2	RHEL 6
		RHEL 5
		SLES 10
		SLES 11
DB2/Open Systems Database Protector	DB2 9.1	AIX 5.3
		RHEL 4
	DB2 9.7	SLES 10
	DB2 9.7.2	RHEL 4
Trino Protector	351	RHEL 6
		RHEL 7
		SLES 10
		SLES 11