Protegrity Row Level Protector Guide 6.6.5

Created on: Aug 8, 2024

Copyright

Table of Contents

Chapter 1

Introduction to This Guide

This guide provides a summary of the Protegrity Row Level Protection components and its core functionality.

The guide describes installation of Protegrity Row Level Protection servers in a Linux environment. It also provides configuration instructions, and basic controlling, such as starting and stopping the system.

The guide also describes how to troubleshoot the Protegrity Row Level Protection.

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 Introduction* provides an overview about the Row Level Protector. In addition, it also contains information about the related functions and components.
- *Section 3 Row Level ProtectorInstallation* explains the installation procedure for the Row Level Protector.
- *Section 4 Row Level Troubleshooting* describes how to troubleshoot certain areas in the Row Level Protector.

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.

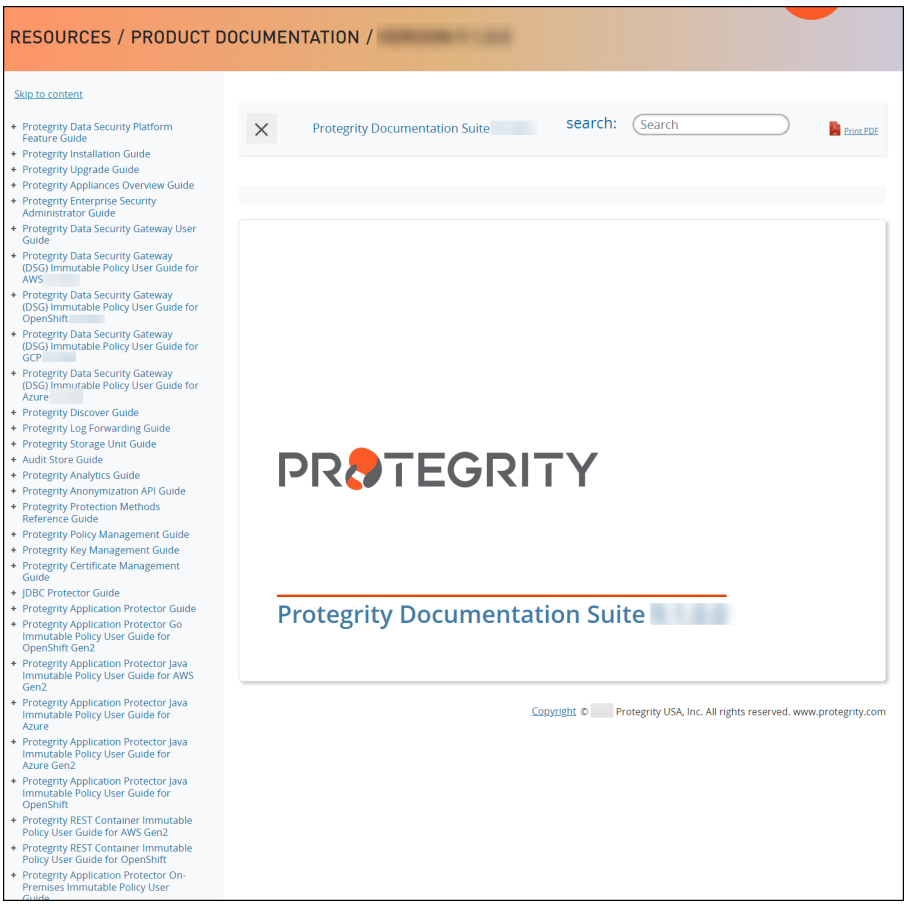

Figure 1-1: Documentation

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

Introduction to the Row Level Protector

Protegrity Row Level Protector is a complete software-based solution for handling row level access control. Row level access control is also provided by using Teradata Table Function UDFs.

To use this functionality, you must define some rule sets that are connected to a role that users can belong to. These roles are then used to verify and return valid data for the specific logged on user.

User verification can be done by using the logged on user ID, or by setting the proxy user value by the Teradata Query Band functionality.

The functionality that is supplied by the Row Level Protection can be illustrated by the table below, where the white cells show the data that specific user can view, the content of the grey cells and rows this user cannot view. Note that access control can be supplied down to cell level.

	Column A	Column B	Column C	Column D	Column E
Row 1					
Row 2					
Row 3					
Row 4					
Row 5					
Row 6					

2.1 Major Features Overview

Protegrity Row Level Protection provides two methods for providing row level access control and one method for column level access control. The column level access control can be used with either form of row level access control.

2.2 Row Level Access Control Functions

In order to provide row level access control, Protegrity provides table function UDFs. These functions return a table that can be used to join with base table data.

There are also helper functions that are regular UDFs and not table functions. They can either be used to validate version, logged on user or to verify if the logged on user has access to a specific rule set and role, this UDF returns a Boolean indicator that states if the customer has access or not.

All functions are prefixed with PTY_RLP.

2.3 Rules Updater Daemon

In order to apply the column and row level functionality, access control information must be shared between the managed access control tables and the UDFs.

You need to provide the functionality to populate and maintain the data in the intermediate table from the regular access control tables used in your application. This intermediate table is then be used by a daemon process called rlp_server, provided by Protegrity to propagate the information to the UDFs.

This daemon process is installed on each node in the system, and reads information using ODBC from the intermediate table that you have populated. This information is then published in shared memory. The connection to the database is handled using configuration files. The daemon process updates the shared memory periodically based on a configured value (daily).

2.4 Components of Protegrity Row Level Protection

The following figure shows the Protegrity Row Level Protection components (in red), and how they interact in the Teradata environment.

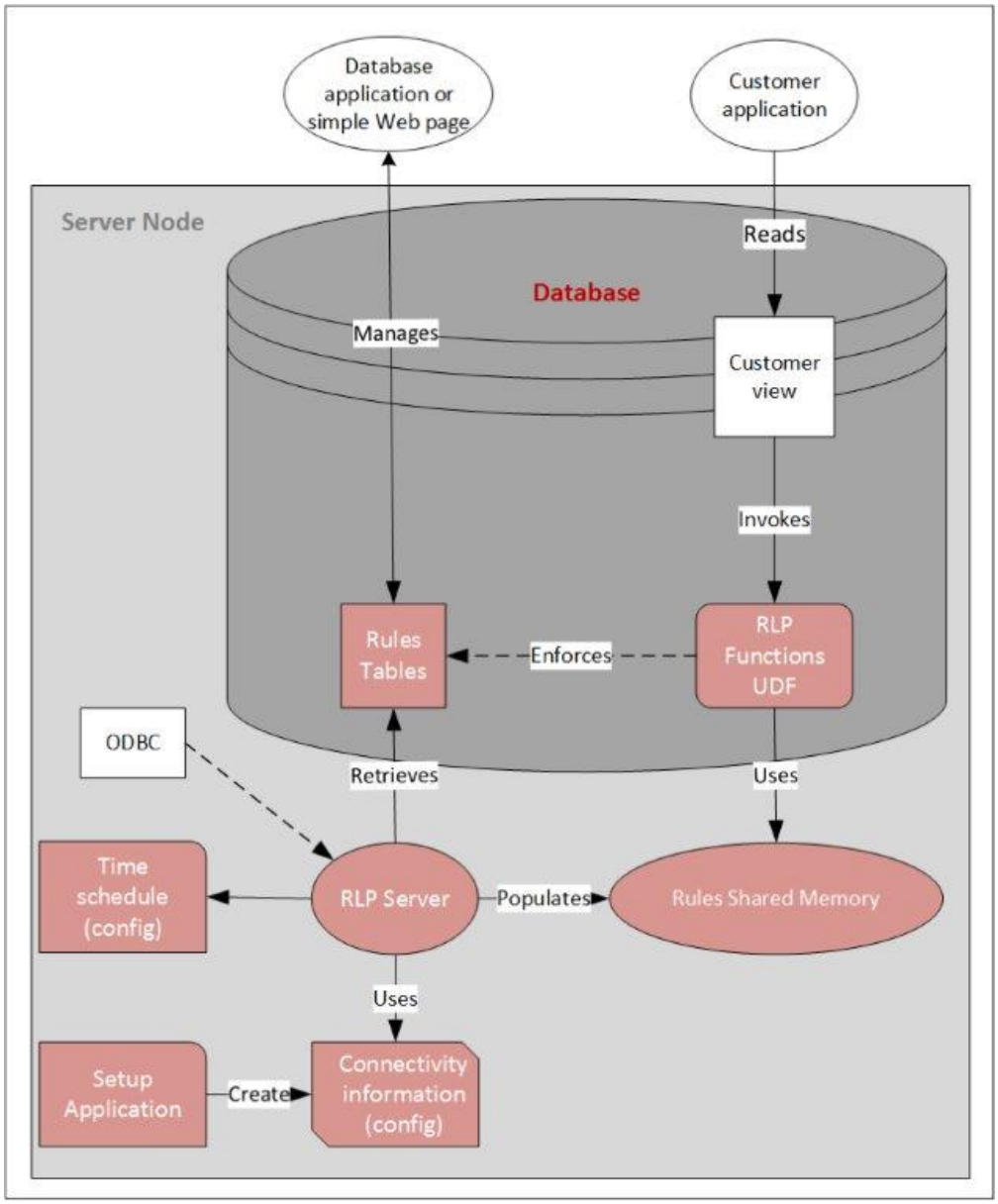

Figure 2-1: Protegrity Row Level Protection Architecture

2.5 Rules Updater (RLP_SERVER)

In order to provide the UDFs with information regarding the requested access validation, a daemon process needs to be executed on the Teradata nodes and provide shared memory/memories with information that is extracted from the access control table. The shared memory size should be 8Mb (each).

The shared memory information is read from the intermediate table and published in shared memory. One shared memory holds the row information and the other holds the column information.

The OS requirement for shared memory may have to be adjusted to facilitate the new shared memories, especially if the regular Protegrity DPS is in use as well.

2.6 Row Level Protection Functions

The row level functions can be executed in four different ways:

- Base functions (default)
- Equal comparison
- Regular expression comparison
- Customer specific functionality

When configured for other than base functions, then the non-base functions are used. Refer to section 3.3 Row Level Protection Functions Configuration and Usage for a more detailed description.

2.6.1 Base Functions

Function	Function Type	Description
PTY_RLP_ACCESS	Table function	This is a table function that takes a rule set id (data type is integer) as parameter and returns a table with up to 5 columns of valid data for the logged on user for that specific rule set id. The function is used to limit the row set returned and only return the rows the user has access to. This can be achieved via implementing this function in a view and join it together with base table data.
PTY_RLP_HAS_ACCESS	Regular function	This regular function takes two parameters, rule set id (integer) and role name (character) and the function is used to verify if the logged on user has access to the input values. The UDF returns an integer value of 1 indicating that the user has access and a value of 0 (zero) if no access is granted.
PTY_RLP_TRUSTED_USERS	Table function	The table function takes one parameter of type integer specifying the rule set to return users that have been granted access to it. It returns a table consisting of the user names of those that have been granted access to that rule set.
PTY_RLP_GETVERSION	Regular function	This function does not take any parameters and returns a character field with information on the version of the row level access control functions installed.
PTY_RLP_WHOAMI	Regular function	The function does not take any parameters and returns the valid user name of the user executing the function.

Note: User names are case sensitive. You must use the case consistently when providing user names in a table. Mixed casing is not supported.

2.6.2 Non-Base Functions

Function	Function Type	Description
PTY_RLP_ROW	Regular function	The function takes four parameters, and a 0 for no access and 1 for access granted when comparing the input parameters against the rules

Function	Function Type	Description
		in the table RLP_ROW that gives a match. If no match then it is 0 – no access.
PTY_RLP_COLUMN_VARCHARLATIN	Regular function	The function takes five parameters, of which the last is the column value. It returns the column value when comparing the input parameters against the rules in the table RLP_ROW that gives a match. If no match then it returns NULL – no access.
PTY_RLP_COLUMN_INTEGER	Regular function	The function takes five parameters, of which the last is the column value. It returns the column value when comparing the input parameters against the rules in the table RLP_ROW that gives a match. If no match then it returns NULL – no access.
PTY_RLP_COLUMN_FLOAT	Regular function	The function takes five parameters, of which the last is the column value. It returns the column value when comparing the input parameters against the rules in the table RLP_ROW that gives a match. If no match then it returns NULL – no access.
PTY_RLP_COLUMN_VARBYTE	Regular function	The function takes five parameters, of which the last is the column value. It returns the column value when comparing the input parameters against the rules in the table RLP_ROW that gives a match. If no match then it returns NULL – no access
PTY_RLP_COLUMN_SMALLINT	Regular function	The function takes five parameters, of which the last is the column value. It returns the column value when comparing the input parameters against the rules in the table RLP_ROW that gives a match. If no match then it returns NULL – no access.
PTY_RLP_COLUMN_DECIMAL8	Regular function	The function takes five parameters, of which the last is the column value. It returns the column value when comparing the input parameters against the rules in the table RLP_ROW that gives a match. If no match then it returns NULL – no access.
PTY_RLP_COLUMN_DECIMAL16	Regular function	The function takes five parameters, of which the last is the column value. It returns the column value when comparing the input parameters against the rules in the table RLP_ROW that gives a match, if no match then it returns NULL – no access.
PTY_RLP_ROW_TABLE_ACCESS	Table function	The function takes one parameter, the userid/customer id to use as comparison against the RLP_ROW table, if it is not supplied it uses the FNC_GetQueryBand to verify if “proxyuser” is set and if not it takes the logged on user. The function then returns a table that contains these columns from the RLP_ROW table where the column RLP_SHARE_IND is set to Y: • SHARING_ID, valid SHARING_ID that the userid/customer has access to. • FIELD1-4, the RLP_ACCESS_KEY is split up into 4 fields of 4 characters each. • CUSTOMER_ID, the function returns the RLP_MEMBER set in the RLP_ROW table.
PTY_RLP_ROW_TABLE_NOACCESS	Table function	The function takes one parameter, the userid/customer id to use as comparison against the RLP_ROW table, if it is not supplied it uses the FNC_GetQueryBand to verify if “proxyuser” is set and if not it takes the logged on user. The function then returns a table that contains these columns from the RLP_ROW table where

Function	Function Type	Description
		the column RLP_SHARE_IND is set to N: SHARING_ID, valid SHARING_ID that the userid/customer has access to. FIELD1-4, the RLP_ACCESS_KEY from the RLP_ROW table that is split up into 4 fields of 4 characters each. CUSTOMER_ID, the function returns the RLP_MEMBER set in the RLP_ROW table.

Note: User names are case sensitive. You must use the case consistently when providing user names in a table. Mixed casing is not supported.

Chapter 3

Row Level Protector Installation

For a successful installation of Protegrity Row Level Protector, we recommend that you read this initial section, otherwise valuable information may be missed. Although the automated installers make the installation and configuration process as simple and easy as possible, a successful deployment of Protegrity Row Level Protector requires the additional knowledge described below.

3.1 Setup Prerequisites

The following prerequisites must be met in order for Protegrity Row Level Protection to be installed correctly.

Rules Updater (RLP_SERVER)

- Approximately, 7.5 MB of free hard disk space is required. This includes space for the service and the key repository. The exact amount of disk space required depends somewhat on the number and size of the security policies.
- The rules updater component consumes a very small amount of CPU cycles on the host machine. A top-of-the-line processor is not required. At a minimum, the host machine should have a 2-GHz processor and 2GB of RAM memory.
- The host machine (node) must be running on SUSE Linux.
- The OS must be configured to have the availability to create two shared memories of 8Mb each.
- Teradata ODBC driver is installed on each node and configured with a DSN name pointing to the correct database where the rule tables exist.

Row Level Protection functions

- Approximately, 2.5 MB of free hard disk space is required.
- The Row Level Protection functions use the available CPU cycles and RAM memory on the host machine to perform access control operations. The host machine should have at least a 2.0 GHz processor and 2 GB of RAM memory.
- The host machine must be running on SUSE Linux with Teradata 13 or above.
- There must be enough space in the target installation database to be able to create the functions.

3.2 Setup Overview

Before you begin
Some of the installation and configuration steps for Protegrity Row Level Protection must be performed manually. This guide describes each step, as far as possible, whether it is automatic or manual. A complete setup of Protegrity Row Level Protection consists of the following steps:

1. **Install the rules updater**Identify the Teradata nodes that you want to host the rules updater. Run an automated installer on the host machine to set up the rules updater.
2. **Configure the rules updater** You optionally may change the rules updater from its default configuration by editing a configuration file. This allows you to modify the default ODBC DSN name, time interval for access control information update, and so on.
3. **Install and configure the Row Level Protection functions** Run the setup script to install the Row Level Protection functions.
4. **Start the Protegrity DPS rules update service**Start Row Level Protector service on each of the nodes. Check that the service is running. This startup procedure can later be added to the system startup procedures; then this Row Level Protector service automatically starts when the node is restarted.

3.2.1 Installing Rules Updater (RLP_SERVER)

Before you begin
Start by identifying the nodes that host the rules updater. The software components are packed within a single setup service, that you need to set up on the machine. Follow these steps:

1. Unpack the file with ***gunzip, tar*** commands and run the automated installer from the directory it has been unpacked into.

   A welcome screen displays.Follow the instructions on the welcome screen.

```
************************************************************
 Welcome to the Row Level Protector Server Setup Wizard
************************************************************

This will install or upgrade the Row Level Protector Server on your computer.
Do you want to continue? [yes or no]
```

Figure 3-1: Installer Welcome Screen

2. Choose the location on the target machine where you would like to install the Protegrity rules updater by entering a folder path.

```
************************************************************
 Welcome to the Row Level Protector Server Setup Wizard
************************************************************

This will install or upgrade the Row Level Protector Server on your computer.
Do you want to continue? [yes or no]
yes
Enter installation directory.
A new directory will be created in the installation directory.
[/opt/protegrity]:

Unpacking...
Extracting files...
```

Figure 3-2: Server Setup Wizard Screen

3. Enter the master key username and password. Note that the password has restrictions on length and the characters it must consist of.

```
Enter master key username
[]:
master
Enter master key password ( Please enter a password that is at least 8 characters
long and contains a mix of numeric and alphabetic characters. )
[]:
master1234
Confirm password
[]:
master1234
Master key created and saved as master.key
Key created and saved as repository.key
```

Figure 3-3: Master Key Entry Screen

4. Enter the certificate username and password. Note that the password has restrictions on length and the characters it must consist of.

```
Enter certificate username
[]:
cert
Enter certificate password ( Please enter a password that is at least 8 characters
long and contains a mix of numeric and alphabetic characters. )
[]:
cert1234
Confirm password
[]:
cert1234
CA certificate saved as cacertificate.crt
Certificate saved as servercertificate.crt
Certificate key saved as servercertificate.key
Certificate saved as clientcertificate.crt
Certificate key saved as clientcertificate.key
```

Figure 3-4: Certificate Entry Screen

5. Enter some information for the database connection and then the installation is finalized.

```
Enter the rule repository database user name
[]:
rlpuser
Enter the rule repository database password
[]:
rlpuserpw
Confirm password
[]:
rlpuserpw
Repository configuration created and saved as repository.cfg.

Enter the System DSN name(data source).
[DBC]:
rulerepos

Enter the database name of the rule repository
[]:
RULEDB

Protegrity Row Level Protection Server installed in /opt/protegrity/defiance_rlp.
```

Figure 3-5: Database configuration

> **Note:**

> It is recommended that the username of the rule repository database should be less than or equal to 107 characters long.
>
> If the username is more than the above-specified length, then it will not be able to connect to the database and thus will not update the rule repository while starting the RLP server.

6. Take a moment to review the files that were copied by the installer. In order to complete the remaining installation and configuration tasks, you need to understand what these files are used for. Navigate to the installation directory that you specified in Step 3 above and examine the file contents.

```
TDExpress14.10_Sles11:/opt/protegrity # ls -l /opt/protegrity/defiance_rlp/bin
total 11520
-rwxr-xr-x 1 root root 2871084 Jun 13 15:32 dpsadmin
-rwxr-xr-x 1 root root 3700648 Jun 13 15:32 dpsinit
drwxr-xr-x 2 root root    4096 Jun 27 05:23 repository
-rwxr-xr-x 1 root root 5189589 Jun 13 15:32 rlp_server
-rwxr-x--- 1 root root    4497 Jun 27 05:23 rlp_server_ctrl

TDExpress14.10_Sles11:/opt/protegrity # ls -l /opt/protegrity/defiance_rlp/data/
total 3244
-rw-r----- 1 root root     611 Jun 27 05:19 cacertificate.crt
-rw-r----- 1 root root      28 Jun 27 05:19 certkeyup.bin
-rw-r----- 1 root root     605 Jun 27 05:19 clientcertificate.crt
-rw-r----- 1 root root     624 Jun 27 05:19 clientcertificate.key
-rwxr-xr-x 1 root root   18907 Jun 13 15:32 dbodbc.plm
-rwxr-xr-x 1 root root  963945 Jun 13 15:32 dbsqlite.plm
-rw-r----- 1 root root      36 Jun 27 05:20 dbup.bin
-rw-r----- 1 root root      36 Jun 27 05:19 kekup.bin
-rwxr-xr-x 1 root root 2269445 Jun 13 15:32 keyinternal.plm
lrwxrwxrwx 1 root root      21 Jun 27 05:23 libodbc.so.1 -> /usr/lib64/libodbc.so
-rw-r----- 1 root root     129 Jun 27 05:19 master.key
-rw-r----- 1 root root     324 Jun 27 05:23 repository.cfg
-rw-r----- 1 root root     129 Jun 27 05:19 repository.key
-rw-r--r-- 1 root root    4384 Jun 13 15:32 rlp_server.cfg
-rw-r----- 1 root root     606 Jun 27 05:19 servercertificate.crt
-rw-r----- 1 root root     624 Jun 27 05:19 servercertificate.key
```

Figure 3-6: List of Installed Files

Files	Description
dpsinit	An executable that must be run if you change the userid and/or password for the user connecting to the database to create a new dbup.bin.
rlp_server	This executable is the daemon process that handles the shared memory and access control information.
rlp_server_ctrl	A startup script for the rlp_server executable.
dpsadmin	A tool using which to connect and perform diagnostics on the rlp_server.
dbup.bin	A secured file containing the userid and password of the user connecting to the database.
kekup.bin	Contains PKCS#5 information.
rlp_server.cfg	A configuration file for the rlp_server executable.
libodbc.so.1	A soft link to the UNIX ODBC driver.
crt key	Necessary files for executing the rlp_server in a secure mode using SSL for communication and encryption for securing information stored in temporary areas.

3.2.1.1 Setting Up the Environment for rlp_server

In order for the *rlp_server* daemon to connect using ODBC, some ODBC settings are required.

Locate the *odbc.ini* file in the system, or create a new *odbc.ini* file. Below is a sample of this file:

```
[ODBC]
InstallDir=/opt/teradata/client/ODBC_64
Trace=0
TraceDll=/opt/teradata/client/ODBC_64/lib/odbctrac.so
TraceFile=/usr/joe/odbcusr/trace.log
TraceAutoStop=0
[ODBC Data Sources]
default=tdata.so
rulerepos=tdata.so
[rulerepos]
Driver= /opt/teradata/client/13.0/odbc_64/lib/tdata.so
Description=Row Level Protector for Teradata 14
DBCName=127.0.0.1
LastUser=
Username=
Password=
Database=
DefaultDatabase=
[default]
Driver= /opt/teradata/client/13.0/odbc_64/lib/tdata.so
Description=Default DSN
DBCName=208.199.59.208
LastUser=
Username=
Password=
Database=
DefaultDatabase=
```

Two environment variables must be defined.

- ODBCINI=/usr/joe/odbc.inip

  Export ODBCINI

- LD_LIBRARY_PATH=/opt/Protegrity/defiance_rlp/data:$LD_LIBRARY_PATH

  Export LD_LIBRARY_PATH

When you do this, you are able to start the *rlp_server*.

3.2.2 Installing the Row Level Protection Functions

Before you begin

1. Run the automated setup program for the Row Level Protection functions. Specify if the installation should be done in a directory other than the default.

```
TDExpress14.10_Sles11:/opt/protegrity # ./RLPUdf1410Setup_Linux_x64_6.5.2.25.sh
*************************************************
 Welcome to the RowColumn Level Protection UDF Setup Wizard
*************************************************

This will install the RowColumn Level Protection PEP on your computer.
Do you want to continue? [yes or no]
yes
Enter installation directory.
A new directory will be created in the installation directory.
[/opt/protegrity]:

Unpacking...
Extracting files...

RowColumn Level Protection UDF installed in /opt/protegrity/defiance_rlp.
```

Figure 3-7: RLP PEP Installation

2. After the setup is done, run the ***createobjects.sql*** script located in the *defiance_rlp/udf/sqlscripts/teradata/* directory with a SQL tool to be chosen (BTEQ and SQL Assistant, among others), and create the functions in the Teradata database.

   Make sure that you are in the correct database to create the functions.

   a.

3.3 Row Level Protection Functions Configuration and Usage

This section contains a short description of the UDF functions. There are four different ways of using the RLP. The default is what is later referred to as “Base” functionality.

Then there are three additional ways:

- Using equal comparison
- Regular expressions usage
- Customer specific implementation

The configuration for running in other than the “Base” mode, is done via a configuration parameter in the rlp_server.cfg under the section “application”:

```
[application]
# What mode to run in, depending on settings different shared memories are setup */
# useruletype = 0 (BASE_RLP_RULES) - default
# useruletype = 1 (EQUAL_COMPARISON_RULES)
# useruletype = 2 (REGEXP_COMPARISON_RULES)
# useruletype = 3 (CUST_RULES)
# useruletype = 0
```

3.3.1 Base Functions

Function	Function Type	Description
PTY_RLP_ACCESS	Table function	This is a table function that takes a rule set id (data type is integer) as parameter and returns a table with up to 5 columns of valid data for the logged on user for that specific rule set id. The function is used to limit the row set returned and only return the rows the user has access to. This can be achieved via implementing this function in a view and join it together with base table data.
PTY_RLP_HAS_ACCESS	Regular function	This regular function takes two parameters, rule set id (integer) and role name (character) and the function is used to verify if the logged on user has access to the input values. The UDF returns an integer value of 1 indicating that the user has access and a value of 0 (zero) if no access is granted.
PTY_RLP_TRUSTED_USERS	Table function	The table function takes one parameter of type integer specifying the rule set to return users that have been granted access to it. It returns a table consisting of the user names of those that have been granted access to that rule set.
PTY_RLP_GETVERSION	Regular function	This function does not take any parameters and returns a character field with information on the version of the row level access control functions installed.
PTY_RLP_WHOAMI	Regular function	The function does not take any parameters and returns the valid user name of the user executing the function.

Note: User names are case sensitive. You must use the case consistently when providing user names in a table. Mixed casing is not supported.

3.3.2 Non-Base Functions

When using other than the base functionality (where useruletype is not set to zero), the functions are read from the shared memory and not a rulecache database.

The rlp_server populates the shared memory by using these two tables in the repository:

- RLP_ROW
- RLP_COL

Table 3-1:

Function	Parameters	Description	Return Values
PTY_RLP_ROW (regular function)	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional)	Uses the SHARING_ID and ACCESSKEY to match against the RLP_ROW definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • logged-on user Depending on the value set for useruletype in the **rlp_server**	The function provides the following return values: • 1, if RLP_SHARE_IND is set to **Y**, and a match is found • 0, if no there is no match or RLP_SHARE_IND is set to **N**

Function	Parameters	Description	Return Values
		config file, the function does the following: 1. validates either with equal comparison 2. uses a regular expression comparison against ACCESSKEY in the RLP_ROW table 3. uses a customer specific comparison	
PTY_RLP_COLUMN_ VARCHARLATIN (regular function)	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional) Data value (Mandatory)	Uses the SHARING_ID and ACCESSKEY to match against the RLP_COL definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user Depending on the value set for useruletype in the rlp_server config file, the function does the following: 1. validates either with equal comparison. 2. uses a regular expression comparison against the accesskey in the RLP_ROW table. 3. uses a customer specific comparison.	The function provides the following return values: • Column value, if RLP_SHARE_IND is set to **Y**, and a match is found • NULL, if no there is no match or RLP_SHARE_IND is set to **N**
PTY_RLP_COLUMN_ INTEGER (regular function)	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional) Data value (Mandatory)	The function uses the SHARING_ID and ACCESSKEY to match against the RLP_COL definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user Depending on the value set for useruletype in the **rlp_server** config file, the function does the following: 1. validates either with equal comparison. 2. uses a regular expression comparison against the accesskey in the RLP_ROW table. 3. uses a customer specific comparison.	The function provides the following return values: • Column value, if RLP_SHARE_IND is set to **Y**, and a match is found • NULL, if no there is no match or RLP_SHARE_IND is set to **N**

Function	Parameters	Description	Return Values
PTY_RLP_COLUMN_FLOAT (regular function)	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional) Data value (Mandatory)	The function uses the SHARING_ID and ACCESSKEY to match against the RLP_COL definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user Depending on the value set for useruletype in the rlp_server config file, the function does the following: 1. validates either with equal comparison. 2. uses a regular expression comparison against the accesskey in the RLP_ROW table. 3. uses a customer specific comparison.	The function provides the following return values: • Column value, if RLP_SHARE_IND is set to **Y**, and a match is found • NULL, if no there is no match or RLP_SHARE_IND is set to **N**
PTY_RLP_COLUMN_VARBYTE (regular function)	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional) Data value (Mandatory)	The function uses the SHARING_ID and ACCESSKEY to match against the RLP_COL definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user Depending on the value set for useruletype in the rlp_server config file, the function does the following: 1. validates either with equal comparison. 2. uses a regular expression comparison against the accesskey in the RLP_ROW table. 3. uses a customer specific comparison.	The function provides the following return values: • Column value, if RLP_SHARE_IND is set to **Y**, and a match is found • NULL, if no there is no match or RLP_SHARE_IND is set to **N**
PTY_RLP_COLUMN_SMALLINT (regular function)	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional) Data value (Mandatory)	The function uses the SHARING_ID and ACCESSKEY to match against the RLP_COL definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user	The function provides the following return values: • Column value, if RLP_SHARE_IND is set to **Y**, and a match is found • NULL, if no there is no match or RLP_SHARE_IND is set to **N**

Function	Parameters	Description	Return Values
		Depending on the value set for useruletype in the **rlp_server** config file, the function does the following: 1. validates either with equal comparison. 2. uses a regular expression comparison against the accesskey in the RLP_ROW table. 3. uses a customer specific comparison.	
PTY_RLP_COLUMN_DECIMAL8 (regular function)	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional) Data value (Mandatory)	The function uses the SHARING_ID and ACCESSKEY to match against the RLP_COL definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user Depending on the value set for useruletype in the rlp_server config file, the function does the following: 1. validates either with equal comparison. 2. uses a regular expression comparison against the accesskey in the RLP_ROW table. 3. uses a customer specific comparison.	The function provides the following return values: • Column value, if RLP_SHARE_IND is set to **Y**, and a match is found • NULL, if no there is no match or RLP_SHARE_IND is set to **N**
PTY_RLP_COLUMN_DECIMAL16	SHARING_ID (Mandatory) ACCESSKEY (Mandatory) CUSTOMER (Optional) CUST_GROUP (Optional) Data value (Mandatory)	The function uses the SHARING_ID and ACCESSKEY to match against the RLP_COL definitions. If CUSTOMER or CUST_GROUP is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user Depending on the value set for useruletype in the rlp_server config file, the function does the following: 1. validates either with equal comparison. 2. uses a regular expression comparison against the	The function provides the following return values: • Column value, if RLP_SHARE_IND is set to **Y**, and a match is found • NULL, if no there is no match or RLP_SHARE_IND is set to **N**

Function	Parameters	Description	Return Values
		accesskey in the RLP_ROW table. 3. uses a customer specific comparison.	
PTY_RLP_ROW_TBL_ACCESS (table function)	CUSTOMER (Optional)	If CUSTOMER is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user It then retrieves all rows from the RLP_ROW table that has RLP_SHARE_IND set to **N** and evaluates how the CUSTOMER is granted access to this, with either of the following: • the CUSTOMER column in the RLP_ROW table is set to a “*” it matches the CUSTOMER supplied to the function or retrieved by QueryBand or logged-on user.	The function returns a table consisting of the following six columns: • SHARING_ID – the valid sharing id from RLP_ROW table • FIELD1-4, the column RLP_ACCESS_KEY in the RLP_ROW table split into 4 columns, each 4 characters wide. • CUSTOMER_ID, from the RLP_ROW table and the column RLP_MEMBER.
PTY_RLP_ROW_TBL_NOACCESS (table function)	CUSTOMER (Optional)	If CUSTOMER is not provided, then the function uses either one of these: • ProxyUser if set with QueryBand • Logged-on user It then retrieves all rows from the RLP_ROW table that has RLP_SHARE_IND set to **N** and evaluates how the CUSTOMER is granted access to this, with either of the following: • the CUSTOMER column in the RLP_ROW table is set to a “*” it matches the CUSTOMER supplied to the function or retrieved by QueryBand or logged-on user.	The function returns a table consisting of the following six columns: • SHARING_ID – the valid sharing id from RLP_ROW table • FIELD1-4, the column RLP_ACCESS_KEY in the RLP_ROW table split into 4 columns, each 4 characters wide. • CUSTOMER_ID, from the RLP_ROW table and the column RLP_MEMBER.
PTY_RLP_COMPILE (regular function)	ACCESSKEY (Mandatory)	This is for testing to compile a regular expression, it takes an ACCESSKEY (64 characters) and tries to compile it as a regular expression. A sample of a regular expression could be “AN[AB]*”, where it accepts a value to start with AN	Returns either one of these: • Message Expression compiles ok if it compiles correctly, • Error message is returned trying to explain what is wrong with the regular expression syntax.

Function	Parameters	Description	Return Values
		and the third character must be A or B, then the rest is ignored.	
PTY_RLP_TEST (regular function)	ACCESSKEY (Mandatory)	This is for testing a regular expression, it takes an ACCESSKEY (64 characters) and tries to validate if the STRINGIN matches the regular expression. A sample of a regular expression can be “AN[AB]*”, where it accepts a value to start with AN and the third character must be A or B, then the rest is ignored.	Returns either one of these: • Match Found! if it matches correctly • No Match Found! if it does not match correctly

Note: User names are case sensitive. You must use the case consistently when providing user names in a table. Mixed casing is not supported.

3.3.3 General Functions

PTY_RLP_WHOAMI() is a regular function which takes no parameters. It returns the user name of the logged on user to Teradata.

Here is a UDF invocation sample:

```
SELECT PTY_RLP_WHOAMI();
```

3.4 Rules Updater Configuration

The Rules Updater application (rlp_server) can be configured via rlp_server.cfg file, available in data sub-folder of the installation folder.

3.4.1 Rlp_server.cfg

```
# Configuration file for the rule access control server
#

# --------------------------------
# Security configuration
# --------------------------------
[security]

# path to master key (KEK) used for all repository and communication security.
#masterkeyfile = <full path>/master.key

# path to user and password file used to decrypt master key.
#kekuserpasswordfile = <full path>/kekup.bin

# Filenamne of the key to be used for encrypt/decrypt repository
#repositorykeyfile = <full path>/repository.key

# path to user and password file used to decrypt certificates private key.
#certuserpasswordfile = <full path>/certkeyup.bin


# --------------------------------
```

```
# Logging configuration,
# Write application log to file as trace
# --------------------------------
[logging]

# level must be set to one of: OFF - No logging, SEVERE, WARNING, INFO, CONFIG, ALL
#level = WARNING

# name of file to log to
#filename = <full path>/rulsynchserver.log

# Append logfile to existing logfile on startup, yes or no
#append = no


# ------------------------
# Listeners configuration
# ------------------------
[listeners]

# listener = <type>, <port>, <protocol>
# Available types:
#    tcp (plain socket)
#    ssl (encrypted communication using secure sockets layer)
# Available protocols:
#    text/xml
#    xc
# Services port
listener = ssl, 15720, text/xml


# --------------------------------
# Secure Socket Layer configuration
# --------------------------------
[ssl]

# This section is only used when listener is configured for ssl.

# One of these has to be configured, type of ssl used for the client
# 1 = SSL client with client authentication.
# 3 = SSL client with no client authentication.
#client = 1

# One of these has to be configured, type of ssl used for the server
# 2 = SSL server with client authentication.
# 4 = SSL server with no client authentication.
#server = 2

# path to private key for server certificate.
#servercertificatekey = <full path>/servercertificate.key

# path to private key for client certificate.
#clientcertificatekey = <full path>/clientcertificate.key

# path to ca certificate.
#cacertificate = <full path>/cacertificate.crt

# path to server certificate.
#servercertificate = <full path>/servercertificate.crt

# path to client certificate.
#clientcertificate = <full path>/clientcertificate.crt


# -------------------------
# Repository configuration
# -------------------------
[repository]

# Configuration file for the repository
#configfile = ./repository.cfg

# ---------------------------
```

```
# Application configuration
# ----------------------------
[application]

#The directory where the server saves its temporary files etc.
#workingdir = <working path>

#The update interval in minutes
# Valid values are in the range 15 to 1440.
updateinterval = 15

# The number of semaphore resources.
# Should be calculated like this: ps -o nlwp -p `pidof actmain`
# This setting is for 16 amps * 95 threads = 1520
semaphoreresources = 1520

# What mode to run in, depending on settings different shared memories are setup */
# useruletype = 0 (BASE_RLP_RULES) - default
# useruletype = 1 (EQUAL_COMPARISON_RULES)
# useruletype = 2 (REGEXP_COMPARISON_RULES)
# useruletype = 3 (CUST_RULES)
# useruletype = 0

# If the source database (Teradata or other) that contains the rules is not running how many
times to try to reconnect.
# retryconnecttimes = 0 (default)
# retryconnecttimes = 0

# When trying to reconnect, how many seconds to sleep in between.
# retrysleeptime = 5 (default)
# retrysleeptime = 5


# ----------------------------
# Disk space management
# ----------------------------
[diskspace]

# The alert value specifies the amount of free disk space that will trigger the
# server to take 'disk full action'.
# The threshold is a delta from the alert value, and will be used to generate
# an audit event by the server to warn when disk space is running out.
# It is also the level that will enable processing after a 'disk full action'.
#
# Values are specified in Megabytes.
# Valid values are in the range 10 to 100000.
#diskfullalert = 10
#threshold = 10

# Action to take when the alert limit is reached.
# Valid values are:
# stop  - (The default) Stop processing. processing will fail with an error.
# nolog - Stop logging. Processing will continue w/o logging.
#diskfullaction = stop

# Interval in seconds between disk full checks.
# Specify a value in the range 10 to 60 seconds (default is 10 seconds).
# Disable the disk full check by setting the value to zero.
#checkinterval = 10

# Max size for one individual log file in Megabytes.
# Valid values are in the range 1 to 100.
#maxlogfilesize = 20
If the source database (Teradata or other) that contains the rules is not running how many
times to try to reconnect.
# retryconnecttimes = 0 (default)
# retryconnecttimes = 0

# When trying to reconnect, how many seconds to sleep in between.
```

```
# retrysleeptime = 5 (default)
# retrysleeptime = 5
```

Note: The *retryconnectimes* parameter defines the number of times that the RLP server will try to connect Teradata Database Engine.

The *retrysleeptime* parameter defines the time, in seconds, for which the RLP server will wait between retries to connect Teradata Database Engine.

Note:

While configuring the RLP server to start automatically after reboot of machine, you must set the *retryconnectimes* parameter to at least 5 times in the *rlp_server.cfg* file. Otherwise, the RLP server may not be able to connect with the Teradata Database.

3.4.2 Repository.cfg

```
[repository]
sqltype=Teradata
modulename=./dbodbc.plm
dbconname=RULEREPOS
tableowner=RULEDB.
dbuserpassword=./dbup.bin
retryconnect=3


[rulecache]
sqltype=SQLite
modulename=./dbsqlite.plm
dbconname=/opt/protegrity/defiance_rlp/data/rulecache.db
#tableowner=<tableowner>
#dbuserpassword=
retryconnect=0
```

Chapter 4

Row Level Troubleshooting

Since the Row Level Protection modules have dependencies on operating system and environmental parts, there can be some specifics in installation and/or starting the application.

4.1 RLP_SERVER

The rlpserver application is dependent on some OS and environment parts being configured properly, such as:

- ODBC
- Shared memory settings in the OS

4.1.1 ODBC

In order to have the ODBC functioning properly it has to be configured. The file odbc.ini (or if using user data sources the .ODBC.INI) has to be updated with a data source:

```
[ODBC Data Sources]
default=tdata.so
mydb=tdata.so

[mydb]
Driver=/opt/teradata/client/13.0/odbc_64/lib/tdata.so
Description=My Terata 13
DBCName=127.0.0.1
LastUser=
Username=
Password=
Database=
DefaultDatabase=
```

You must set the environment variables ODBCINI (if the odbc.ini is not located in the folder) and LD_LIBRARY_PATH:

```
export ODBCINI=/etc/odbc.ini
export LD_LIBRARY_PATH=/opt/protegrity/defiance_rlp/data:$LD_LIBRARY_PATH
```

4.1.2 Shared Memory

In a normal Linux environment the maximum segment size is 32 MB, which is usually sufficient and the default maximum total size is 2097152 pages. To find out how large the page size is, you can run the OS command ***`getconf PAGE_SIZE`***, the page size is normally 4096 bytes.

However, if more applications are running and utilizing shared memory, the settings may have to be modified. You can do this by making some OS configuration changes.

For example, if you need to allow 16 GB (with a page size of 4096), you can run:

```
sysctl -w kernel.shmmax=17179869184
sysctl -w kernel.shmall=4194304
```

To preserve these settings in case of a reboot, it is recommended to store them in the file */etc/sysctl.conf*.